Regular temporal cost functions

Denis Kuperberg\textsuperscript{1}
Joint work with Thomas Colcombet\textsuperscript{1} and Sylvain Lombardy\textsuperscript{2}

Liafa/CNRS/Université Paris 7, Denis Diderot, France
Ligm - Université Paris-Est Marne-la-Vallée, France

ICALP 2010
Introduction

- Cost function: counting extension of languages
Introduction

- Cost function: counting extension of languages
- Temporal class: measuring time
Introduction

- Cost function: counting extension of languages
- Temporal class: measuring time
- Simplify constructions and lower complexity of algorithms for this class
Introduction

- Cost function: counting extension of languages
- Temporal class: measuring time
- Simplify constructions and lower complexity of algorithms for this class
- Algebraic characterization of temporal cost functions
Outline

Introduction

Counting events in words
  Cost automata
  Cost functions
  Temporal automata
  Clock-languages

Algebraic characterization
  Stabilization semigroups
  Temporal semigroups

Conclusion
Cost automata

**Aim** : To represent functions $\mathbb{A}^* \rightarrow \mathbb{N} \cup \{\infty\}$ with automata.

**Cost automaton** :
- nondeterministic finite-state
- finite set of counters, ranging over $\mathbb{N}$, initial value 0
- each transition perform actions on each counter

**Atomic actions** : increment ($i$), reset ($r$), check ($c$).
Semantics of cost automata

Two semantics:

\[ [\mathcal{A}]_B(u) = \inf \{ n / \text{there is a run with maximal check value } n \} , \]
and  \[ [\mathcal{A}]_S(u) = \sup \{ n / \text{there is a run with minimal check value } n \} . \]

with \( \inf \emptyset = \infty \) and \( \sup \emptyset = 0 \).

Example

\[ [\mathcal{A}]_B = | \cdot |_a \quad \text{and} \quad [\mathcal{A}']_S = \maxblock_a : u \mapsto \max \{ n / a^n \ \text{factor of } u \} \]

\[ a : ic \quad a, b : \varepsilon \quad a : i \quad \varepsilon : \varepsilon \quad \varepsilon : cr \quad a, b : \varepsilon \]

\[ 0 \quad 1 \quad 2 \quad 3 \]
More on cost automata

**Remark**: A standard automaton $A$ computing $L$ can be viewed as a cost automaton without any counter.
Then $\llbracket A \rrbracket_B = \chi_L$ and $\llbracket A \rrbracket_S = \chi_{A^* \setminus L}$ with

$$\chi_L(u) = \begin{cases} 
0 & \text{if } u \in L \\
\infty & \text{if } u \notin L 
\end{cases}$$

**Theorem ([Krob 94])**

*The equivalence of two cost automata is undecidable.*

**Solution**: Loosing some precision on the counting, but keeping information about bounds.
Cost functions

If \( f, g : \mathbb{A}^* \rightarrow \mathbb{N} \cup \{\infty\} \), then

\[
f \approx g \text{ if } \forall X \subseteq \mathbb{A}^*, f|_X \text{ bounded } \iff g|_X \text{ bounded.}
\]

iff \( \exists \alpha : \mathbb{N} \rightarrow \mathbb{N} \) such as \( f \leq \alpha \circ g \) and \( g \leq \alpha \circ f \) (with \( \alpha(\infty) = \infty \))

Cost function: equivalence class for \( \approx \) relation.

Example

For \( \mathbb{A} = \{a, b, c\} \),

\[
\max(| \cdot |_a, | \cdot |_b) \approx | \cdot |_a + | \cdot |_b \quad \text{but} \quad | \cdot |_a \not\approx \maxblock_a
\]
Known results on cost automata

Extension of the notion of language via \( \chi_L : L \neq L' \implies \chi_L \not\approx \chi_{L'} \).

Theorem (Colcombet 09)

\( B\) - and \( S\) - automata have the same expressive power (modulo \( \approx \)), and the translations are effective.

Theorem (Colcombet 09)

It is decidable whether two cost automata compute the same cost function (modulo \( \approx \)).

Automata-computable cost functions are called regular.
Temporal automata

Intuitive idea: Measuring the time, counting consecutive events.

Temporal automata: Only actions: \( \{ic, r\} \) for B-automata
(Kirsten and Bala’s desert automata) and \( \{i, cr\} \) for S-automata.

Example

For \( \mathbb{A} = \{a, b\} \), \text{maxblock}_a \text{ is temporal, but } \cdot \cdot |_a \text{ is not.}

\[
\begin{align*}
  a &: ic \\
  b &: r \\
\end{align*}
\]

\[
\begin{align*}
  a &: ic \\
  b &: \varepsilon \\
\end{align*}
\]
Temporal automata

Intuitive idea: Measuring the time, counting consecutive events.

Temporal automata: Only actions: \{ic, r\} for B-automata (Kirsten and Bala’s desert automata) and \{i, cr\} for S-automata.

Example

For \(A = \{a, b\}\), maxblock\(_a\) is temporal, but \(|\cdot|_a\) is not.

Theorem

For a cost function, it is equivalent (modulo \(\approx\)) to be recognized by

- temporal B-automaton
- temporal B-automaton with 1 counter
- temporal S-automaton
- temporal S-automaton with 1 counter

We say then that the cost function is temporal.
A ticking clock to measure time

**Problem**: Two different formalisms, inefficient constructions in the general case

**Idea**: Using an automaton reading simultaneously a word and a "clock" to measure the time

**Clock on** $u$ : Word on $\{w, t\}$ (for "wait" and "tick") of length $|u|$. 
Example of $\text{maxblock}_a$

Call a clock **good for** $u$ if each block of $a$’s in $u$ contains at most one tick of $c$.

**Principle** Minimal maxperiod of all **good clocks for** $u$ and maximal minperiod of all **bad clocks for** $u$ are both equivalent (modulo $\approx$) to $\text{maxblock}_a(u)$.

\[ t \quad t \quad t \quad t \quad t \quad \leftarrow \rightarrow \quad t \quad t \quad t \quad t \]

\[ b \quad a \quad b \quad b \quad a \quad a \quad a \quad a \quad a \quad a \quad b \quad a \quad a \quad b \]

\[ t \quad \leftarrow \rightarrow \quad t \]
Representing temporal cost functions with regular languages

Definition

$L$ regular on alphabet $\mathbb{A} \times \{w, t\}$ is a clock-language for $f$ iff

\[
\langle L \rangle_B : u \mapsto \inf \{\text{maxperiod}(c) : \langle u, c \rangle \in L\}
\]
\[
\langle L \rangle_S : u \mapsto \sup \{\text{minperiod}(c) : \langle u, c \rangle \not\in L\}
\]

are both equivalent to $f$ modulo $\equiv$.

Theorem

A cost function $f$ is temporal iff there exists a clock-language for $f$.

This give us easily closure of temporal class under min, max, projections,...
Introduction

Counting events in words
Cost automata
Cost functions
Temporal automata
Clock-languages

Algebraic characterization
Stabilization semigroups
Temporal semigroups

Conclusion
Algebraic characterization of regular cost functions

Reminder: regular language $\iff$ finite semigroup (Myhill)

Stabilization semigroup: $S = \langle S, \cdot, \leq, \# \rangle$, ordered semigroup with a $\#$-operator: stabilization over idempotents ($e = e \cdot e$).

$e^\#$ means "$e$ repeated a lot of times".

Theorem (Colcombet 09)

Stabilization semigroups recognize exactly the set of regular cost functions, and translations to or from cost automata are effective.
Temporal semigroups

**Reminder** : Star-free language $\Leftrightarrow$ group-trivial semigroup (Schützenberger)

**Theorem**

A cost function is temporal iff it is recognizable by a **temporal stabilization semigroup** (effective structural condition).
Temporal semigroups

**Reminder**: Star-free language $\Leftrightarrow$ group-trivial semigroup (Schützenberger)

**Theorem**

*A cost function is temporal iff it is recognizable by a temporal stabilization semigroup* (effective structural condition).

**Theorem**

*Let $f$ be a regular cost function,*

- There exists a (quotient-wise) minimal stabilization semigroup $S$ recognizing $f$
- $S$ is computable effectively
- $f$ is temporal iff $S$ is temporal
Temporal semigroups

**Reminder** : Star-free language $\Leftrightarrow$ group-trivial semigroup (Schützenberger)

**Theorem**

A cost function is temporal iff it is recognizable by a **temporal** stabilization semigroup (effective structural condition).

**Theorem**

Let $f$ be a regular cost function,

- There exists a (quotient-wise) minimal stabilization semigroup $S$ recognizing $f$
- $S$ is computable effectively
- $f$ is temporal iff $S$ is temporal

**Corollary**

*It is decidable whether a regular cost function is temporal.*
Conclusion

Summary:
- Temporal class defined via cost automata
- Simplifications of constructions in this class, via clock-languages
- Characterization by stabilization semigroups
- Decidability of the temporal class

Further work:
- Extension to infinite words, trees
- Other classes of regular cost functions