Finding Short Synchronizing Words for Prefix Codes

Andrew Ryzhikov

LIGM, Université Paris-Est

17th Mons Theoretical Computer Science Days
joint work with M. Szykuła (Univ. of Wrocław)

Prefix codes

Definition

A set of words is called a *prefix code* if no word in the set is a prefix of another word.

Examples: $\{a, ba\}$, a^*ba are prefix codes, and $\{a, aba\}$ is not.

Definition

A word w is called *synchronizing* for a prefix code X if for any words u, v such that $uwv \in X^*$ both uw and wv are in X^*. A code having a synchronizing word is also called *synchronizing*.

Examples: for $\{a, ba\}$ the word ba is synchronizing.

Prefix codes

Definition

A set of words is called a *prefix code* if no word in the set is a prefix of another word.

Examples: $\{a, ba\}$, a^*ba are prefix codes, and $\{a, aba\}$ is not.

Definition

A word w is called *synchronizing* for a prefix code X if for any words u, v such that $uwv \in X^*$ both uw and wv are in X^*. A code having a synchronizing word is also called *synchronizing*.

Examples: for $\{a, ba\}$ the word ba is synchronizing.

Synchronization example

Take a code $\{a, baab\}$.

baab a a baab a a baab baab a
ba a baab a a baab a a bbaaba
ba abaabaabaa baab baab a

The word *baabbaab* is synchronizing: after reading it, only one interpretation is possible.

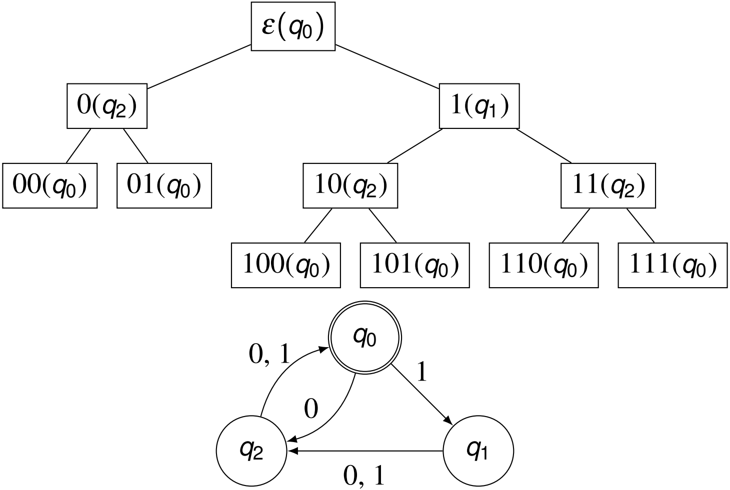
Synchronization example

Take a code $\{a, baab\}$.

baab a a baab a a baab baab a
ba a baab a a baab a a bbaaba
ba abaabaabaa baab baab a

The word *baabbaab* is synchronizing: after reading it, only one interpretation is possible.

Decoding of prefix codes



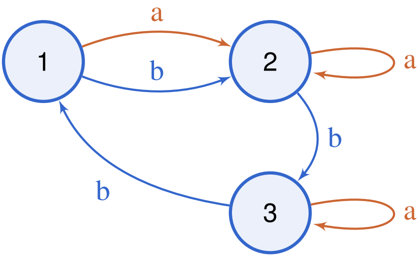
The code $0\{0,1\} \cup 1\{0,1\}^2 = \{00, 01, 100, 101, 110, 111\}$.

Synchronizing Automata

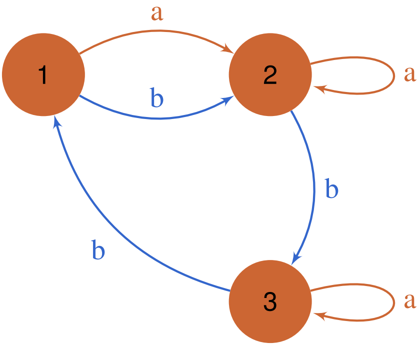
We consider deterministic finite automata without inputs and outputs.

Definition

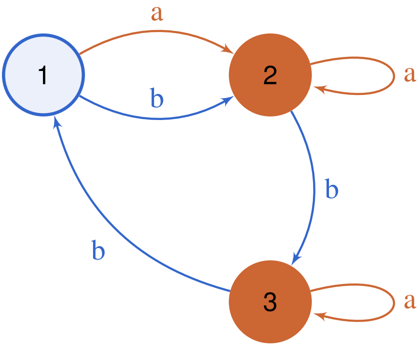
A (complete) automaton $A = (Q, \Sigma, \delta)$ is *synchronizing*, if there exists a word $w \in \Sigma^*$ such that after reading this word A is sent to some particular state regardless of its initial state. Such word is called a *synchronizing word*.



Example

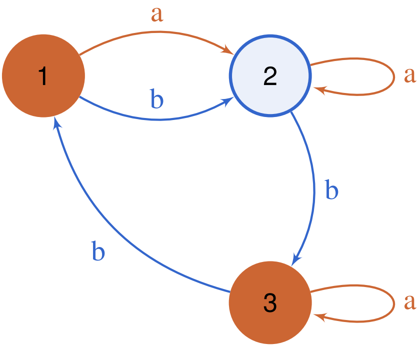


Example



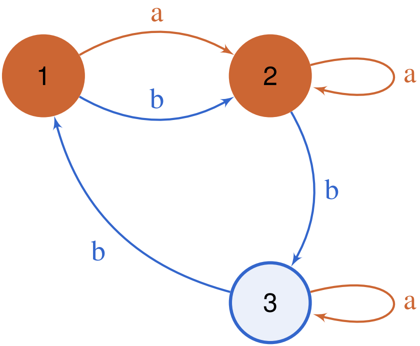
a

Example



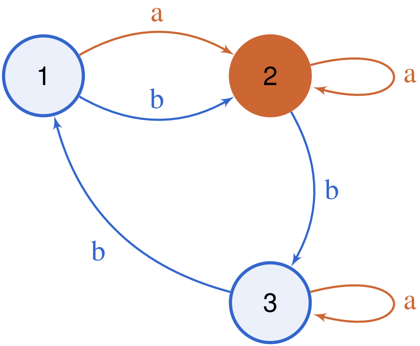
ab

Example



abb

Example



abba

Synchronizing Automata

Definition

Let $A = (Q, \Sigma, \delta)$ be a partial automaton.
A word w is called *synchronizing* for A if there exists a state $q \in Q$ such that w maps each state of A either to q or the mapping of w is undefined for this state, and there is at least one state such that the mapping of w is defined for it.

Some more definitions

Definition

A prefix code is called *maximal* if it is not contained in another prefix code.

The code $\{aa, b\}$ is not maximal, because it is contained in a (maximal) code $\{aa, ab, b\}$.

Definition

A partial automaton is called *strongly connected* if for every ordered pair q, q' of states there is a word mapping q to q'.

Some more definitions

Definition

A prefix code is called *maximal* if it is not contained in another prefix code.

The code $\{aa, b\}$ is not maximal, because it is contained in a (maximal) code $\{aa, ab, b\}$.

Definition

A partial automaton is called *strongly connected* if for every ordered pair q, q' of states there is a word mapping q to q'.

Codes and Automata

There is a strong relation between prefix codes and DFAs.

Definition

Let A be a DFA with a state r which is the initial and the only accepting state. A word is called a *first return word* if it maps r to itself such that each non-empty prefix does not map r to itself.

Theorem

The set of first return words of a strongly connected

1. partial DFA is a prefix code;
2. complete DFA is a maximal prefix code;
3. partial Huffman decoder is a finite prefix code;
4. Huffman decoder is a finite maximal prefix code.

Codes and Automata

There is a strong relation between prefix codes and DFAs.

Definition

Let A be a DFA with a state r which is the initial and the only accepting state. A word is called a *first return word* if it maps r to itself such that each non-empty prefix does not map r to itself.

Theorem

The set of first return words of a strongly connected

1. partial DFA is a prefix code;
2. complete DFA is a maximal prefix code;
3. partial Huffman decoder is a finite prefix code;
4. Huffman decoder is a finite maximal prefix code.

Codes and Automata

There is a strong relation between prefix codes and DFAs.

Definition

Let A be a DFA with a state r which is the initial and the only accepting state. A word is called a *first return word* if it maps r to itself such that each non-empty prefix does not map r to itself.

Theorem

The set of first return words of a strongly connected

1. partial DFA is a prefix code;
2. complete DFA is a maximal prefix code;
3. partial Huffman decoder is a finite prefix code;
4. Huffman decoder is a finite maximal prefix code.

Two main synchronization problems

There are two main questions:

1. Extremal: how long can a shortest synchronizing word be?
2. Algorithmic: how hard is it to decide synchronizability or to find a shortest synchronizing word?

Two main synchronization problems

There are two main questions:

1. Extremal: how long can a shortest synchronizing word be?
2. Algorithmic: how hard is it to decide synchronizability or to find a shortest synchronizing word?

Two main synchronization problems

There are two main questions:

1. Extremal: how long can a shortest synchronizing word be?
2. Algorithmic: how hard is it to decide synchronizability or to find a shortest synchronizing word?

Extremal

Conjecture (Černý, 1971)

For each synchronizing automaton with n states there exists a synchronizing word of length $(n-1)^2$.

Proved for several particular classes of automata.

Theorem (Pin, 1983)

For each synchronizing automaton with n states there exists a synchronizing word of length $\frac{n^3-n}{6}$.

Theorem (Szykuła, 2018)

For each synchronizing automaton with n states there exists a synchronizing word of length $\frac{15617n^3+7500n^2+9375n-31250}{93750}$.

Improvement by 4/46875.

Extremal

Conjecture (Černý, 1971)

For each synchronizing automaton with n states there exists a synchronizing word of length $(n-1)^2$.

Proved for several particular classes of automata.

Theorem (Pin, 1983)

For each synchronizing automaton with n states there exists a synchronizing word of length $\frac{n^3-n}{6}$.

Theorem (Szykuła, 2018)

For each synchronizing automaton with n states there exists a synchronizing word of length $\frac{15617n^3+7500n^2+9375n-31250}{93750}$.

Improvement by 4/46875.

Extremal

Conjecture (Černý, 1971)

For each synchronizing automaton with n states there exists a synchronizing word of length $(n-1)^2$.

Proved for several particular classes of automata.

Theorem (Pin, 1983)

For each synchronizing automaton with n states there exists a synchronizing word of length $\frac{n^3-n}{6}$.

Theorem (Szykuła, 2018)

For each synchronizing automaton with n states there exists a synchronizing word of length $\frac{15617n^3+7500n^2+9375n-31250}{93750}$.

Improvement by 4/46875.

Algorithmic

Theorem

It can be checked in polynomial time whether a partial deterministic finite automaton is synchronizing.

> SHORT SYNC WORD
> *Input*: A synchronizing partial automaton A;
> *Output*: The length of a shortest synchronizing word for A.

The problem is studied for different classes of automata.

Algorithmic

Theorem

It can be checked in polynomial time whether a partial deterministic finite automaton is synchronizing.

> SHORT SYNC WORD
> *Input*: A synchronizing partial automaton A;
> *Output*: The length of a shortest synchronizing word for A.

The problem is studied for different classes of automata.

Algorithmic

Theorem

It can be checked in polynomial time whether a partial deterministic finite automaton is synchronizing.

> SHORT SYNC WORD
> *Input*: A synchronizing partial automaton A;
> *Output*: The length of a shortest synchronizing word for A.

The problem is studied for different classes of automata.

Approximability

An algorithm is called r-approximation for a minimization problem if it outputs a solution which is at most r times larger the size of the optimal solution.

Theorem

SHORT SYNC WORD is in NP.

Theorem

There exists an $O(n)$-approximation polynomial time algorithm for SHORT SYNC WORD for partial automata.

Approximability

An algorithm is called r-approximation for a minimization problem if it outputs a solution which is at most r times larger the size of the optimal solution.

Theorem

SHORT SYNC WORD is in NP.

Theorem

There exists an $O(n)$-approximation polynomial time algorithm for SHORT SYNC WORD for partial automata.

Approximability

An algorithm is called r-approximation for a minimization problem if it outputs a solution which is at most r times larger the size of the optimal solution.

Theorem

SHORT SYNC WORD is in NP.

Theorem

There exists an $O(n)$-approximation polynomial time algorithm for SHORT SYNC WORD for partial automata.

Existing results

Theorem (Eppstein, 1990)

SHORT SYNC WORDS is NP-hard.

Theorem (Berlinkov, 2014)

The SHORT SYNC WORD problem cannot be approximated in polynomial time within a factor of $c \log n$ for some $c > 0$ for n-state automata over an alphabet of size $n^{1+\gamma}$ for every $\gamma > 0$ unless $P = NP$.

Theorem (Gawrychowski, Straszak, 2015)

The SHORT SYNC WORD problem cannot be approximated in polynomial time within a factor of $n^{1-\varepsilon}$ for every $\varepsilon > 0$ for n-state binary automata unless P = NP.

Literal Decoders

Definition

Given a finite maximal prefix code X over an alphabet Σ, the *literal Huffman decoder* recognizing X^* is an automaton $A = (Q, \Sigma, \delta)$ defined as follows. The states of A correspond to all proper prefixes of the words in X, and the transition function is defined as

$$\delta(q, x) = \begin{cases} qx & \text{if } qx \notin X, \\ \varepsilon & \text{if } qx \in X \end{cases}$$

Our results

Approximability of SHORT SYNC WORD:

class	lower bound	upper bound
strongly connected	$n^{1-\varepsilon}$	$O(n)$
partial Huffman	$n^{\frac{1}{2}-\varepsilon}$	$O(n)$
Huffman	$c \log n$	$O(n)$
literal Huffman	1	$O(\log n)$

All lower bounds are for binary and all upper bound are for general automata.

Idea of the proof: Huffman decoders

1. Prove $c \log n$-inapproximability for strongly acyclic automata over an alphabet of size $n^{1+\gamma}$ (via a reduction from SET COVER).
Strongly acyclic – no cycles but loops in the sink state.

2. Transform a strongly acyclic automaton over k letters into a Huffman decoder over $k+2$ letters with the same length of a shortest synchronizing word.

3. Make the automaton binary using a composition with a Wielandt automaton.

Idea of the proof: Huffman decoders

1. Prove $c \log n$-inapproximability for strongly acyclic automata over an alphabet of size $n^{1+\gamma}$ (via a reduction from SET COVER). Strongly acyclic – no cycles but loops in the sink state.

2. Transform a strongly acyclic automaton over k letters into a Huffman decoder over $k+2$ letters with the same length of a shortest synchronizing word.

3. Make the automaton binary using a composition with a Wielandt automaton.

Idea of the proof: Huffman decoders

1. Prove $c \log n$-inapproximability for strongly acyclic automata over an alphabet of size $n^{1+\gamma}$ (via a reduction from SET COVER). Strongly acyclic – no cycles but loops in the sink state.

2. Transform a strongly acyclic automaton over k letters into a Huffman decoder over $k+2$ letters with the same length of a shortest synchronizing word.

3. Make the automaton binary using a composition with a Wielandt automaton.

Open problems

Improve lower and upper inapproximability bounds.

Conjecture (R., Szykuła, 2018)

There exists an exact polynomial time algorithm for the SHORT SYNC WORD problem for literal Huffman decoders.

Mortal Words

Definition

A word w is called *mortal* for a partial automaton if its action is undefined for each state of this automaton.

> SHORT MORTAL WORD
> *Input*: A partial automaton A with at least one undefined transition;
> *Output*: The length of a shortest mortal word for A.

Mortal Words

Definition

A word w is called *mortal* for a partial automaton if its action is undefined for each state of this automaton.

> SHORT MORTAL WORD
> *Input*: A partial automaton A with at least one undefined transition;
> *Output*: The length of a shortest mortal word for A.

Mortal Words

Theorem (R., Szykuła, 2018)

There exists a $O(\log n)$-approximation polynomial time algorithm for the SHORT MORTAL WORD problem for n-state literal Huffman decoders. This algorithm always finds a mortal word of length $O(n \log n)$.

Theorem (R., Szykuła, 2018)

Unless P = NP, the SHORT MORTAL WORD problem cannot be approximated in polynomial time within a factor of
(i) $n^{1-\varepsilon}$ for every $\varepsilon > 0$ for n-state binary strongly connected partial automata;
(ii) $c \log n$ for some $c > 0$ for n-state binary partial Huffman decoders.

Mortal Words

Theorem (R., Szykuła, 2018)

There exists a $O(\log n)$-approximation polynomial time algorithm for the SHORT MORTAL WORD problem for n-state literal Huffman decoders. This algorithm always finds a mortal word of length $O(n \log n)$.

Theorem (R., Szykuła, 2018)

Unless P = NP, the SHORT MORTAL WORD problem cannot be approximated in polynomial time within a factor of
(i) $n^{1-\varepsilon}$ for every $\varepsilon > 0$ for n-state binary strongly connected partial automata;
(ii) $c \log n$ for some $c > 0$ for n-state binary partial Huffman decoders.

Thank you! Any questions?