Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access

Daniel Reidenbach, Markus L. Schmid, Loughborough University, UK

LATA 2011
Motivation

- Extended regular expressions (and pattern languages (introduced by Angluin in 1980))
- NP-complete membership problem
- Nontrivial subclasses with polynomial membership problem
- Automata as general algorithmic frameworks
Motivation

Motivation

\[ x_1 \cdot x_2 \cdot x_2 \cdot x_1 \cdot x_3 \cdot x_3 \cdot x_4 \cdot x_4 \cdot x_3 \cdot x_4 \cdot x_1 \cdot x_2 \]
Motivation

Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access
Motivation

Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access

Daniel Reidenbach, Markus L. Schmid, Loughborough University, UK
Motivation

Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access
Motivation

Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access
Motivation

Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access

\[ x_1 \cdot x_2 \cdot x_2 \cdot x_1 \cdot x_3 \cdot x_3 \cdot x_4 \cdot x_4 \cdot x_3 \cdot x_4 \cdot x_1 \cdot x_2 \]
Motivation

Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access
Motivation

\[ x_1 \cdot x_2 \cdot x_2 \cdot x_1 \cdot x_3 \cdot x_3 \cdot x_4 \cdot x_4 \cdot x_3 \cdot x_4 \cdot x_1 \cdot x_2 \]

Daniel Reidenbach, Markus L. Schmid, Loughborough University, UK
Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access
$X_1 \cdot X_2 \cdot X_2 \cdot X_1 \cdot X_3 \cdot X_3 \cdot X_4 \cdot X_4 \cdot X_3 \cdot X_4 \cdot X_1 \cdot X_2$

1 2 2 1 3 3 4 4 3 4 1 2
Motivation

\[ x_1 \cdot x_2 \cdot x_2 \cdot x_1 \cdot x_3 \cdot x_3 \cdot x_4 \cdot x_4 \cdot x_3 \cdot x_4 \cdot x_1 \cdot x_2 \]
Motivation

Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access
Motivation

Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access
Motivation

Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access

Daniel Reidenbach, Markus L. Schmid, Loughborough University, UK
Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access
Motivation

$x_1 \cdot x_2 \cdot x_2 \cdot x_1 \cdot x_3 \cdot x_3 \cdot x_4 \cdot x_4 \cdot x_3 \cdot x_4 \cdot x_1 \cdot x_2$

Daniel Reidenbach, Markus L. Schmid, Loughborough University, UK
Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access
Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access
Motivation

Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access
Motivation

Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access
Motivation

Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access
Motivation

Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access

Daniel Reidenbach, Markus L. Schmid, Loughborough University, UK
Motivation

Daniel Reidenbach, Markus L. Schmid, Loughborough University, UK

Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access
Motivation

Finding Shufle Words that Represent Optimal Scheduling of Shared Memory Access
Motivation

Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access
Motivation

Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access
Motivation

Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access
Motivation

Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access

\[ x_1 \cdot x_2 \cdot x_2 \cdot x_1 \cdot x_3 \cdot x_3 \cdot x_4 \cdot x_4 \cdot x_3 \cdot x_4 \cdot x_1 \cdot x_2 \]
Motivation

Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access
Motivation

\[ x_1 \cdot x_2 \cdot x_2 \cdot x_1 \cdot x_3 \cdot x_3 \cdot x_4 \cdot x_4 \cdot x_3 \cdot x_4 \cdot x_1 \cdot x_2 \]

Daniel Reidenbach, Markus L. Schmid, Loughborough University, UK

Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access
Motivation

Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access
Motivation

Motivation

Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access
Motivation

Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access
Motivation

Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access
Motivation

Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access

Daniel Reidenbach, Markus L. Schmid, Loughborough University, UK
Motivation

Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access
Motivation

Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access
Scheduling example

Process 1: a b a c b c

Memory: □ □
Motivation

The Problem

Approaches

Scheduling example

Process 1: a b a c b c

Memory: a b
Scheduling example

Process 1: a b a c b c

Memory: [c] [b]

Daniel Reidenbach, Markus L. Schmid, Loughborough University, UK
Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access
Scheduling example

Process 1: a b a c b c

Process 2: a b c

Memory: □ □
Scheduling example

Process 1: a b a c b c

Process 2: a b c

Memory: a b
Scheduling example

Process 1: a b a c b c

Process 2: a b c

Memory: a b c
Scheduling example

Process 1: a b a c b c

Process 2: a b c

Memory: [a] [b] [c]
Scheduling example

Process 1:  a  b  a  c  b  c

Process 2:  a  b  c

Memory:  [ ]  [ ]  [ ]
Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access
Scheduling example

Process 1: \[ \text{a} \rightarrow \text{b} \rightarrow \text{a} \rightarrow \text{c} \rightarrow \text{b} \rightarrow \text{c} \]

Process 2: \[ \text{a} \rightarrow \text{b} \rightarrow \text{c} \]

Memory: \[ \square \rightarrow \square \rightarrow \square \]
Scheduling example

Process 1:

Process 2:

Memory:
Motivation The Problem Approaches

Scheduling example

Process 1: a b a c b c

Process 2: a b c

Memory: c b

Daniel Reidenbach, Markus L. Schmid, Loughborough University, UK
Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access
The Scope Coincidence Degree

- The *scope* of symbol $a$ coincides with the scope of symbol $b$ in $w \in \Sigma^*$ iff $w = \cdots a \cdots b \cdots a \cdots$ or $w = \cdots b \cdots a \cdots b \cdots$.
- Let $w \in \Sigma^*$. $scd(w)$ denotes the maximum number of symbols in $w$, the scopes of which are pairwise coinciding.
The Scope Coincidence Degree

- The *scope* of symbol $a$ coincides with the scope of symbol $b$ in $w \in \Sigma^*$ iff $w = \cdots a \cdots b \cdots a \cdots$ or $w = \cdots b \cdots a \cdots b \cdots$.

- Let $w \in \Sigma^*$. $\text{scd}(w)$ denotes the maximum number of symbols in $w$, the scopes of which are pairwise coinciding.

$$w_1 := a \ b \ a \ c \ b \ c \ a \ b \ c$$

$$w_2 := a \ b \ a \ a \ a \ b \ c \ b \ c \ c$$
The Scope Coincidence Degree

- The *scope* of symbol $a$ coincides with the scope of symbol $b$ in $w \in \Sigma^*$ iff $w = \cdots a \cdots b \cdots a \cdots$ or $w = \cdots b \cdots a \cdots b \cdots$.
- Let $w \in \Sigma^*$. $\text{scd}(w)$ denotes the maximum number of symbols in $w$, the scopes of which are pairwise coinciding.

$$w_1 := \begin{array}{cccccccc}
a & b & a & c & b & c & a & b & c 
\end{array}$$

$$w_2 := \begin{array}{cccccccc}
a & b & a & a & a & b & c & b & c & c 
\end{array}$$
The Scope Coincidence Degree

- The *scope* of symbol $a$ coincides with the scope of symbol $b$ in $w \in \Sigma^*$ iff $w = \cdots a \cdots b \cdots a \cdots$ or $w = \cdots b \cdots a \cdots b \cdots$.
- Let $w \in \Sigma^*$. $scd(w)$ denotes the maximum number of symbols in $w$, the scopes of which are pairwise coinciding.

$$w_1 := \begin{array}{cccccccc}
a & b & a & c & b & c & a & b & c \\
\end{array}$$

$$w_2 := \begin{array}{cccccccc}
a & b & a & a & a & b & c & b & c & c \\
\end{array}$$

Daniel Reidenbach, Markus L. Schmid, Loughborough University, UK
Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access
The Scope Coincidence Degree

- The *scope* of symbol \(a\) coincides with the scope of symbol \(b\) in \(w \in \Sigma^*\) iff \(w = \cdots a \cdots b \cdots a \cdots\) or \(w = \cdots b \cdots a \cdots b \cdots\).

- Let \(w \in \Sigma^*.\) \(scd(w)\) denotes the maximum number of symbols in \(w\), the scopes of which are pairwise coinciding.

\[
\begin{align*}
    w_1 & := a b a c b c a b c \\
    w_2 & := a b a a b c b c c
\end{align*}
\]
The Scope Coincidence Degree

- The *scope* of symbol $a$ coincides with the scope of symbol $b$ in $w \in \Sigma^*$ iff $w = \cdots a \cdots b \cdots a \cdots$ or $w = \cdots b \cdots a \cdots b \cdots$.
- Let $w \in \Sigma^*$. $\text{scd}(w)$ denotes the maximum number of symbols in $w$, the scopes of which are pairwise coinciding.

$w_1 := a \ b \ a \ c \ b \ c \ a \ b \ c \ \text{scd}(w_1) = 3$

$w_2 := a \ b \ a \ a \ b \ c \ b \ c \ c$
The Scope Coincidence Degree

- The *scope* of symbol \( a \) coincides with the scope of symbol \( b \) in \( w \in \Sigma^* \) iff \( w = \cdots a \cdots b \cdots a \cdots \) or \( w = \cdots b \cdots a \cdots b \cdots \).
- Let \( w \in \Sigma^* \). \( \text{scd}(w) \) denotes the maximum number of symbols in \( w \), the scopes of which are pairwise coinciding.

\[
\begin{align*}
    w_1 := & \quad a \quad b \quad a \quad c \quad b \quad c \quad a \quad b \quad c \\
    w_2 := & \quad a \quad b \quad a \quad a \quad b \quad c \quad b \quad c \quad c \\
\end{align*}
\]

\( \text{scd}(w_1) = 3 \)
The Scope Coincidence Degree

- The *scope* of symbol $a$ coincides with the scope of symbol $b$ in $w \in \Sigma^*$ iff $w = \cdots a \cdots b \cdots a \cdots$ or $w = \cdots b \cdots a \cdots b \cdots$.
- Let $w \in \Sigma^*$. $\text{scd}(w)$ denotes the maximum number of symbols in $w$, the scopes of which are pairwise coinciding.

$$w_1 := \begin{array}{ccccccc} a & b & a & c & b & c & a \\ b & c & a & b & c & a & b \\ c & a & b & c & a & b & c \end{array} \quad \text{scd}(w_1) = 3$$

$$w_2 := \begin{array}{cccccccc} a & b & a & a & a & b & c & b \\ b & c & b & c & b & c & c \end{array}$$
The Scope Coincidence Degree

- The *scope* of symbol $a$ coincides with the scope of symbol $b$ in $w \in \Sigma^*$ iff $w = \cdots a \cdots b \cdots a \cdots$ or $w = \cdots b \cdots a \cdots b \cdots$.

- Let $w \in \Sigma^*$. $\text{scd}(w)$ denotes the maximum number of symbols in $w$, the scopes of which are pairwise coinciding.

$$w_1 := \begin{array}{cccccc}
    a & b & a & c & b & c \\
    a & b & c & a & b & c
  \end{array} \quad \text{scd}(w_1) = 3$$

$$w_2 := \begin{array}{cccccc}
    a & b & a & a & b & c \\
    c & b & c & c
  \end{array}$$
The Scope Coincidence Degree

- The *scope* of symbol $a$ coincides with the scope of symbol $b$ in $w \in \Sigma^*$ iff $w = \cdots a \cdots b \cdots a \cdots$ or $w = \cdots b \cdots a \cdots b \cdots$.
- Let $w \in \Sigma^*$. $scd(w)$ denotes the maximum number of symbols in $w$, the scopes of which are pairwise coinciding.

$$w_1 := \begin{array}{cccccccc}
  a & b & a & c & b & c & a & b & c
\end{array} \quad scd(w_1) = 3
$$

$$w_2 := \begin{array}{cccccccc}
  a & b & a & a & a & b & c & b & c & c
\end{array} \quad scd(w_2) = 2$$
Generalised Problem on Shuffle Words

**SWminSCD\(_\Sigma\)**

**INPUT:** \((w_1, \ldots, w_k) \in (\Sigma^+)^k\), for some \(k \in \mathbb{N}\).

**OUTPUT:** \(w \in w_1 \sqcup \ldots \sqcup w_k\) such that there exists no \(w' \in w_1 \sqcup \ldots \sqcup w_k\) with \(\text{scd}(w') < \text{scd}(w)\).

\(w_1 \sqcup w_2\) denotes the *shuffle* of \(w_1, w_2 \in \Sigma^*\).
Naive approach to SWminSCD$_\Sigma$

Let $w_1, w_2, \ldots, w_k \in \Sigma^*$, where $n := |w_1 \cdot w_2 \cdot \ldots \cdot w_k|$.

\[ |w_1 \sqcup w_2 \sqcup \ldots \sqcup w_k| \leq \binom{n}{|w_1|, \ldots, |w_k|} = \frac{n!}{|w_1|! \times \ldots \times |w_k|!} \cdot \]
Computing Shuffle Words Step by Step

Let $w_1, w_2, \ldots, w_k \in \Sigma^*$ be an input $SWminSCD_{\Sigma}$.

- $u_1 := SWminSCD_{\Sigma}(w_1, w_2)$,
- $u_2 := SWminSCD_{\Sigma}(u_1, w_3)$,
- $u_3 := SWminSCD_{\Sigma}(u_2, w_4)$,
- $\ldots$
Computing Shuffle Words Step by Step

Let $w_1 := aa$, $w_2 := bb$ and $w_3 := ba$.

- $aabb \in w_1 \sqcup w_2$ is optimal.
- For every $u \in aabb \sqcup w_3$, $scd(u) = 1$.
- $bbbaaa \in w_1 \sqcup w_2 \sqcup w_3$ with $scd(bbbaaa) = 0$. 
The Intrinsic Complexity

Let $w_1, w_2, w_3 \in \Sigma^*$ be arbitrary words.
Let $w_1, w_2, w_3 \in \Sigma^*$ be arbitrary words.
The Intrinsic Complexity

Let $w_1, w_2, w_3 \in \Sigma^*$ be arbitrary words.
Let $w_1, w_2, w_3 \in \Sigma^*$ be arbitrary words.
Let $w_1, w_2, w_3 \in \Sigma^*$ be arbitrary words.
Let $w_1, w_2, w_3 \in \Sigma^*$ be arbitrary words.
Let $w_1, w_2, w_3 \in \Sigma^*$ be arbitrary words.
The Intrinsic Complexity

Let $w_1, w_2, w_3 \in \Sigma^*$ be arbitrary words.
Let \( w_1, w_2, w_3 \in \Sigma^* \) be arbitrary words.
Our Algorithm

Main Result

$SW\text{min}SCD_\Sigma$ on input $(w_1, w_2, \ldots, w_k)$ can be solved in time $O(|w_1 \cdot w_2 \cdots w_k| \times k|\Sigma|)$. 
Our Algorithm

The idea of the algorithm:
Our Algorithm

The idea of the algorithm:

\[ u \]
\[ v \]
\[ w \]
Our Algorithm

The idea of the algorithm:

\[
\begin{align*}
\text{a} & \quad u \\
\text{} & \quad v \\
\text{} & \quad w
\end{align*}
\]
Our Algorithm

The idea of the algorithm:

\[
\begin{align*}
\text{u} \\
\text{v} \\
\text{w} \\
\text{a}
\end{align*}
\]
Our Algorithm

The idea of the algorithm:

\[ u_1 \quad u \]
\[ v_1 \quad v \]
\[ w_1 \quad w \]
Our Algorithm

The idea of the algorithm:

\[ u \]

\[ v_1 \]

\[ w_1 \]

\[ a \quad u_1 \]

Daniel Reidenbach, Markus L. Schmid, Loughborough University, UK
Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access
Our Algorithm

The idea of the algorithm:

$$a \quad u_1 \quad v_1$$

$$w_1$$

$$w$$

$$v$$

$$u$$
Our Algorithm

The idea of the algorithm:

\[
\begin{array}{c}
\text{a} & u_1 & v_1 & w_1 \\
\end{array}
\]

\[
\begin{array}{c}
u \\
v \\
w \\
\end{array}
\]
Our Algorithm

The idea of the algorithm:

\[
\begin{array}{c}
\text{a} & u_1 & v_1 & w_1 \\
\end{array}
\]
Our Algorithm

The idea of the algorithm:

\[ a \quad u_1 \quad v_1 \quad w_1 \quad b \]
Our Algorithm

The idea of the algorithm:

\[
\begin{align*}
    u_2 & \quad u \\
    v_2 & \quad v \\
    w_2 & \quad w \\
    a & \quad u_1 \quad v_1 \quad w_1 \quad b
\end{align*}
\]
Our Algorithm

The idea of the algorithm:

\[
\begin{align*}
 & u \\
 & v_2 \\
 & w_2 \\
 & a \hspace{0.5cm} u_1 \hspace{0.5cm} v_1 \hspace{0.5cm} w_1 \hspace{0.5cm} b \hspace{0.5cm} u_2
\end{align*}
\]
Our Algorithm

The idea of the algorithm:

\[
\begin{array}{c}
\text{u} \\
\text{v} \\
\text{w}_2 \\
\text{w} \\
\end{array}
\begin{array}{c}
a \\
u_1 \\
v_1 \\
w_1 \\
b \\
u_2 \\
v_2 \\
\end{array}
\]
Our Algorithm

The idea of the algorithm:

\[
\begin{array}{cccccccc}
 a & u_1 & v_1 & w_1 & b & u_2 & v_2 & w_2 \\
\end{array}
\]

Daniel Reidenbach, Markus L. Schmid, Loughborough University, UK
Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access
Our Algorithm

The idea of the algorithm:

\[ \begin{align*}
  &\text{u} \\
  &\text{c} \\
  &\text{w} \\
  &\text{a} \quad \text{u}_1 \quad \text{v}_1 \quad \text{w}_1 \quad \text{b} \quad \text{u}_2 \quad \text{v}_2 \quad \text{w}_2
\end{align*} \]
Our Algorithm

The idea of the algorithm:

\[
\begin{array}{c}
\text{a} & u_1 & v_1 & w_1 & b & u_2 & v_2 & w_2 & c \\
\end{array}
\]

\[ u \]
\[ v \]
\[ w \]
The idea of the algorithm:

\[
\begin{array}{c}
\text{u}_3 & \text{u} \\
\text{v}_3 & \text{v} \\
\text{w}_3 & \text{w}
\end{array}
\]
Our Algorithm

The idea of the algorithm:

\[
\begin{array}{ccc}
\text{a} & u_1 & v_1 & w_1 & b & u_2 & v_2 & w_2 & c & u_3 \\
\end{array}
\]

\[
\begin{array}{cccc}
\text{u} & & & \\
\text{v} & & & \\
\text{w} & & & \\
\end{array}
\]
Our Algorithm

The idea of the algorithm:

\[ a \quad u_1 \quad v_1 \quad w_1 \quad b \quad u_2 \quad v_2 \quad w_2 \quad c \quad u_3 \quad v_3 \]

\[ u \quad w_3 \quad v \]

Daniel Reidenbach, Markus L. Schmid, Loughborough University, UK
Finding Shuffle Words that Represent Optimal Scheduling of Shared Memory Access
Our Algorithm

The idea of the algorithm:

\[ \text{a} u_1 v_1 w_1 b u_2 v_2 w_2 c u_3 v_3 w_3 \]
Thank you for your attention.