More pattern matching problems on compressed strings

Tim Nonner

June 1st, 2005
Let $\Sigma$ be a finite alphabet, and let $u, v \in \Sigma^*$ with $u = a_1a_2 \cdots a_n$ ($a_i \in \Sigma$).

$u$ is a **subword** of $v$ if $v \in \Sigma^*a_1\Sigma^*a_2\Sigma^* \cdots a_n\Sigma^*$.

$u$ is a **factor** of $v$ if $v = xuy$ for some $x, y \in \Sigma^*$.

Example: $aa$ is a subword of $babbbab$ but not a factor.
Let $\Sigma$ be a finite alphabet, and let $u, v \in \Sigma^*$ with $u = a_1a_2 \cdots a_n$ ($a_i \in \Sigma$).

$u$ is a subword of $v$ if $v \in \Sigma^*a_1\Sigma^*a_2\Sigma^* \cdots a_n\Sigma^*$.

$u$ is a factor of $v$ if $v = xuy$ for some $x, y \in \Sigma^*$.

Example: $aa$ is a subword of $babbbab$ but not a factor.
Two computational problems

**SUBWORD:**

INPUT: \( u, v \in \Sigma^* \)

QUESTION: Is \( u \) a subword of \( v \) ?

**FACTOR:**

INPUT: \( u, v \in \Sigma^* \)

QUESTION: Is \( u \) a factor of \( v \) ?
Two computational problems

**SUBWORD:**

**INPUT:** $u, v \in \Sigma^*$

**QUESTION:** Is $u$ a subword of $v$?

**FACTOR:**

**INPUT:** $u, v \in \Sigma^*$

**QUESTION:** Is $u$ a factor of $v$?
Compressed strings

A straight-line program over the alphabet $\Sigma$ is a context-free grammar $H = (V, \Sigma, P, S)$ in Chomsky normal form such that:

- For every $A \in V$ there exists exactly one production of the form $A \rightarrow \alpha$ in $P$.
- There exists a linear ordering $A_1, A_2, \ldots, A_n$ of $V$ such that $S = A_1$ and for every production $A_i \rightarrow A_j A_k$ we have $i < j, k$.

$\text{eval}(H)$ denotes the unique word generated by $H$.

Let $A$ be a nonterminal symbol of $H$. Then $\text{eval}(A)$ denotes the unique word generated by $A$ in $H$.

Let $|H|$ be the number of nonterminal symbols of $H$. 
A straight-line program over the alphabet $\Sigma$ is a context-free grammar $H = (V, \Sigma, P, S)$ in Chomsky normal form such that:

- For every $A \in V$ there exists exactly one production of the form $A \rightarrow \alpha$ in $P$.
- There exists a linear ordering $A_1, A_2, \ldots, A_n$ of $V$ such that $S = A_1$ and for every production $A_i \rightarrow A_j A_k$ we have $i < j, k$.

$\text{eval}(H)$ denotes the unique word generated by $H$.

Let $A$ be a nonterminal symbol of $H$. Then $\text{eval}(A)$ denotes the unique word generated by $A$ in $H$.

Let $|H|$ be the number of nonterminal symbols of $H$. 
A straight-line program over the alphabet $\Sigma$ is a context-free grammar $H = (V, \Sigma, P, S)$ in Chomsky normal form such that:

- For every $A \in V$ there exists exactly one production of the form $A \rightarrow \alpha$ in $P$.
- There exists a linear ordering $A_1, A_2, \ldots, A_n$ of $V$ such that $S = A_1$ and for every production $A_i \rightarrow A_jA_k$ we have $i < j, k$.

$\text{eval}(H)$ denotes the unique word generated by $H$.

Let $A$ be a nonterminal symbol of $H$. Then $\text{eval}(A)$ denotes the unique word generated by $A$ in $H$.

Let $|H|$ be the number of nonterminal symbols of $H$. 
A straight-line program over the alphabet $\Sigma$ is a context-free grammar $H = (V, \Sigma, P, S)$ in Chomsky normal form such that:

- For every $A \in V$ there exists exactly one production of the form $A \rightarrow \alpha$ in $P$.
- There exists a linear ordering $A_1, A_2, \ldots, A_n$ of $V$ such that $S = A_1$ and for every production $A_i \rightarrow A_jA_k$ we have $i < j, k$.

$\text{eval}(H)$ denotes the unique word generated by $H$.

Let $A$ be a nonterminal symbol of $H$. Then $\text{eval}(A)$ denotes the unique word generated by $A$ in $H$.

Let $|H|$ be the number of nonterminal symbols of $H$. 
Example: Let $H$ be the straight-line program that consists of the following productions:

$$
S \rightarrow A_1 A_2 \\
A_1 \rightarrow A_4 A_3 \\
A_2 \rightarrow A_3 A_4 \\
A_3 \rightarrow A_6 A_5 \\
A_4 \rightarrow A_5 A_6 \\
A_5 \rightarrow a \\
A_6 \rightarrow b
$$

Then $\text{eval}(H) = abbabaab$ and $\text{eval}(A_1) = abba$. 
Example: Let $H$ be the straight-line program that consists of the following productions:

\[
S \rightarrow A_1 A_2 \\
A_1 \rightarrow A_4 A_3 \\
A_2 \rightarrow A_3 A_4 \\
A_3 \rightarrow A_6 A_5 \\
A_4 \rightarrow A_5 A_6 \\
A_5 \rightarrow a \\
A_6 \rightarrow b
\]

Then $\text{eval}(H) = abbabaab$ and $\text{eval}(A_1) = abba$. 
Two computational problems

COMPRESSED-SUBWORD:
INPUT: Straight-line programs $G$ and $H$
QUESTION: Is $\text{eval}(G)$ a subword of $\text{eval}(H)$?

COMPRESSED-FACTOR:
INPUT: Straight-line programs $G$ and $H$
QUESTION: Is $\text{eval}(G)$ a factor of $\text{eval}(H)$?

Gasieniec, Karpinski, Miyazaki, Plandowski, Rytter, Shinohara, Takeda (mid 90's): COMPRESSED-FACTOR can be solved in polynomial time.

Yury Lifshits, Markus Lohrey: COMPRESSED-SUBWORD is NP-hard, even when restricted to the alphabet $\{a, b\}$.

Tim Nonner         More pattern matching problems on compressed strings
Two computational problems

COMPRESSED-SUBWORD:
INPUT: Straight-line programs $G$ and $H$
QUESTION: Is $\text{eval}(G)$ a subword of $\text{eval}(H)$?

COMPRESSED-FACTOR:
INPUT: Straight-line programs $G$ and $H$
QUESTION: Is $\text{eval}(G)$ a factor of $\text{eval}(H)$?

Gasieniec, Karpinski, Miyazaki, Plandowski, Rytter, Shinohara, Takeda (mid 90's): COMPRESSED-FACTOR can be solved in polynomial time.

Yury Lifshits, Markus Lohrey: COMPRESSED-SUBWORD is NP-hard, even when restricted to the alphabet $\{a, b\}$. 
Two computational problems

**COMPRESSED-SUBWORD:**
INPUT: Straight-line programs $G$ and $H$
QUESTION: Is $\text{eval}(G)$ a subword of $\text{eval}(H)$?

**COMPRESSED-FACTOR:**
INPUT: Straight-line programs $G$ and $H$
QUESTION: Is $\text{eval}(G)$ a factor of $\text{eval}(H)$?

Gasieniec, Karpinski, Miyazaki, Plandowski, Rytter, Shinohara, Takeda (mid 90's): COMPRESSED-FACTOR can be solved in polynomial time.

Yury Lifshits, Markus Lohrey: COMPRESSED-SUBWORD is NP-hard, even when restricted to the alphabet $\{a, b\}$. 
Two computational problems

COMPRESSED-SUBWORD:
INPUT: Straight-line programs $G$ and $H$
QUESTION: Is $\text{eval}(G)$ a subword of $\text{eval}(H)$?

COMPRESSED-FACTOR:
INPUT: Straight-line programs $G$ and $H$
QUESTION: Is $\text{eval}(G)$ a factor of $\text{eval}(H)$?

Gasieniec, Karpinski, Miyazaki, Plandowski, Rytter, Shinohara, Takeda (mid 90's): COMPRESSED-FACTOR can be solved in polynomial time.

Yury Lifshits, Markus Lohrey: COMPRESSED-SUBWORD is NP-hard, even when restricted to the alphabet $\{a, b\}$. 
Two easier computational problems

**PARTLY-COMPRESSED-SUBWORD:**
INPUT: A word \( g \) and a straight-line program \( H \)
QUESTION: Is \( g \) a subword of \( \text{eval}(H) \) ?

**PARTLY-COMPRESSED-FACTOR:**
INPUT: A word \( g \) and a straight-line program \( H \)
QUESTION: Is \( g \) a factor of \( \text{eval}(H) \) ?

Gasieniec, Gibbons, Rytter (1997):
PARTLY-COMPRESSED-FACTOR can be solved in \( O(\log |g| + \log |H|) \) time with \( O(|g| \cdot |H|) \) processors.

Today: PARTLY-COMPRESSED-SUBWORD can be solved in
\( O(|\Sigma| \cdot (\log |g| + \log |H|) + \log^2 |H|) \) time with \( O(|g| \cdot |H| + |H|^3) \) processors.
Two easier computational problems

PARTLY-COMPRESSED-SUBWORD:
INPUT: A word $g$ and a straight-line program $H$
QUESTION: Is $g$ a subword of $\text{eval}(H)$?

PARTLY-COMPRESSED-FACTOR:
INPUT: A word $g$ and a straight-line program $H$
QUESTION: Is $g$ a factor of $\text{eval}(H)$?

Gasieniec, Gibbons, Rytter (1997):
PARTLY-COMPRESSED-FACTOR can be solved in $O(\log |g| + \log |H|)$ time with $O(|g| \cdot |H|)$ processors.

Today: PARTLY-COMPRESSED-SUBWORD can be solved in $O(|\Sigma| \cdot (\log |g| + \log |H|) + \log^2 |H|)$ time with $O(|g| \cdot |H| + |H|^3)$ processors.
Two easier computational problems

PARTLY-COMPRESSED-SUBWORD:
INPUT: A word $g$ and a straight-line program $H$
QUESTION: Is $g$ a subword of $\text{eval}(H)$?

PARTLY-COMPRESSED-FACTOR:
INPUT: A word $g$ and a straight-line program $H$
QUESTION: Is $g$ a factor of $\text{eval}(H)$?

Gasieniec, Gibbons, Rytter (1997):
PARTLY-COMPRESSED-FACTOR can be solved in $O(\log |g| + \log |H|)$ time with $O(|g| \cdot |H|)$ processors.

Today: PARTLY-COMPRESSED-SUBWORD can be solved in $O(|\Sigma| \cdot (\log |g| + \log |H|) + \log^2 |H|)$ time with $O(|g| \cdot |H| + |H|^3)$ processors.
Two easier computational problems

**PARTLY-COMPRESSED-SUBWORD:**
**INPUT:** A word $g$ and a straight-line program $H$
**QUESTION:** Is $g$ a subword of $\text{eval}(H)$?

**PARTLY-COMPRESSED-FACTOR:**
**INPUT:** A word $g$ and a straight-line program $H$
**QUESTION:** Is $g$ a factor of $\text{eval}(H)$?

Gasieniec, Gibbons, Rytter (1997):
PARTLY-COMPRESSED-FACTOR can be solved in $O(\log |g| + \log |H|)$ time with $O(|g| \cdot |H|)$ processors.

Today:
PARTLY-COMPRESSED-SUBWORD can be solved in $O(|\Sigma| \cdot (\log |g| + \log |H|) + \log^2 |H|)$ time with $O(|g| \cdot |H| + |H|^3)$ processors.
A Parallel Random Access Machine (PRAM) is an ordinary computer, that is able to do operations in parallel.

If we want to do $n$ operations in parallel, then we need $n$ processors.

A problem can be solved in parallel, iff it can be solved on a PRAM in polylogarithmic time with a polynomial number of processors.

The class of problems, that can be solved in parallel on a PRAM, is the class $NC$. 
A Parallel Random Access Machine (PRAM) is an ordinary computer, that is able to do operations in parallel.

If we want to do \( n \) operations in parallel, then we need \( n \) processors.

A problem can be solved in parallel, iff it can be solved on a PRAM in polylogarithmic time with a polynomial number of processors.

The class of problems, that can be solved in parallel on a PRAM, is the class \( \text{NC} \).
A Parallel Random Access Machine (PRAM) is an ordinary computer, that is able to do operations in parallel.

If we want to do $n$ operations in parallel, then we need $n$ processors.

A problem can be solved in parallel, iff it can be solved on a PRAM in polylogarithmic time with a polynomial number of processors.

The class of problems, that can be solved in parallel on a PRAM, is the class $NC$. 
A **Parallel Random Access Machine (PRAM)** is an ordinary computer, that is able to do operations in parallel.

If we want to do \( n \) operations in parallel, then we need \( n \) processors.

A problem can be solved in parallel, iff it can be solved on a PRAM in polylogarithmic time with a polynomial number of processors.

The class of problems, that can be solved in parallel on a PRAM, is the class **NC**.
A problem can be solved **optimally** in parallel, iff the total number of operations needed is the time to solve this problem sequentially.

A problem can be solved **nearly optimally** in parallel, iff the number of processors needed is the time to solve this problem sequentially.
A problem can be solved **optimally** in parallel, iff the total number of operations needed is the time to solve this problem sequentially.

A problem can be solved **nearly optimally** in parallel, iff the number of processors needed is the time to solve this problem sequentially.
A straight-line program defines a directed acyclic graph (dag).

Problems related to dags form the following hierarchy:

1. can be solved in parallel
2. can be solved optimally/nearly optimally, if the transitive hull of the dag is known
3. can be solved optimally/nearly optimally in parallel


A straight-line program defines a directed acyclic graph (dag).

Problems related to dags form the following hierarchy:

1. can be solved in parallel
2. can be solved optimally/nearly optimally, if the transitive hull of the dag is known
3. can be solved optimally/nearly optimally in parallel


A straight-line program defines a directed acyclic graph (dag).

Problems related to dags form the following hierarchy:

1. can be solved in parallel
2. can be solved optimally/nearly optimally, if the transitive hull of the dag is known
3. can be solved optimally/nearly optimally in parallel


A straight-line program defines a directed acyclic graph (dag).

Problems related to dags form the following hierarchy:

1. can be solved in parallel
2. can be solved optimally/nearly optimally, if the transitive hull of the dag is known
3. can be solved optimally/nearly optimally in parallel


A straight-line program defines a directed acyclic graph (dag).

Problems related to dags form the following hierarchy:

1. can be solved in parallel
2. can be solved optimally/nearly optimally, if the transitive hull of the dag is known
3. can be solved optimally/nearly optimally in parallel


INPUT: A word \( g \) and a \( \epsilon \)-free and context-free grammar \( K \) in Chomsky normal form

QUESTION: Does \( g \) belong to the language \( L(K) \)?

Let \( t \) be the number of nonterminal symbols and \( s \) the number of production rules of \( K \). Then this problem can be solved in \( O(\log |g|) \) time with \( O(t^3 \cdot |g|^6 + s) \) processors.

Reduction of PARTLY-COMPRESSED-SUBWORD: Generate a grammar \( K \), such that \( L(K) \) is the set of all words, that are subwords of \( \text{eval}(H) \).

PARTLY-COMPRESSED-SUBWORD can be solved in \( O(\log |g| + \log^2 |H|) \) time with \( O(|H|^3 \cdot |g|^6) \) processors.
Reduction

INPUT: A word $g$ and a $\epsilon$-free and context-free grammar $K$ in Chomsky normal form
QUESTION: Does $g$ belong to the language $L(K)$?

Let $t$ be the number of nonterminal symbols and $s$ the number of production rules of $K$. Then this problem can be solved in $O(\log |g|)$ time with $O(t^3 \cdot |g|^6 + s)$ processors.

Reduction of PARTLY-COMPRESSED-SUBWORD: Generate a grammar $K$, such that $L(K)$ is the set of all words, that are subwords of $\text{eval}(H)$.

PARTLY-COMPRESSED-SUBWORD can be solved in $O(\log |g| + \log^2 |H|)$ time with $O(|H|^3 \cdot |g|^6)$ processors.
INPUT: A word $g$ and a $\epsilon$-free and context-free grammar $K$ in Chomsky normal form

QUESTION: Does $g$ belong to the language $L(K)$?

Let $t$ be the number of nonterminal symbols and $s$ the number of production rules of $K$. Then this problem can be solved in $O(\log |g|)$ time with $O(t^3 \cdot |g|^6 + s)$ processors.

Reduction of PARTLY-COMPRESSED-SUBWORD: Generate a grammar $K$, such that $L(K)$ is the set of all words, that are subwords of $\text{eval}(H)$.

PARTLY-COMPRESSED-SUBWORD can be solved in $O(\log |g| + \log^2 |H|)$ time with $O(|H|^3 \cdot |g|^6)$ processors.
INPUT: A word $g$ and a $\epsilon$-free and context-free grammar $K$ in chomsky normal form

QUESTION: Does $g$ belong to the language $L(K)$?

Let $t$ be the number of nonterminal symbols and $s$ the number of production rules of $K$. Then this problem can be solved in $O(\log |g|)$ time with $O(t^3 \cdot |g|^6 + s)$ processors.

Reduction of PARTLY-COMPRESSED-SUBWORD: Generate a grammar $K$, such that $L(K)$ is the set of all words, that are subwords of $\text{eval}(H)$.

PARTLY-COMPRESSED-SUBWORD can be solved in $O(\log |g| + \log^2 |H|)$ time with $O(|H|^3 \cdot |g|^6)$ processors.
The number of processors is quite big.

Constraint: Assume, that the size of the alphabet is quite small / the size of the alphabet is polylogarithmically bounded.
The number of processors is quite big.

Constraint: Assume, that the size of the alphabet is quite small / the size of the alphabet is polylogarithmically bounded.
PARTLY-COMPRESSED-SUBWORD is a language recognition problem for a compressed word.

Let $m := |g|$, and let for $i \in \{0, \ldots, m-1\}$ the number $\text{IN}(g, A, i)$ be the maximal number, such that $g[i+1..\text{IN}(g, A, i)]$ is a subword of $\text{eval}(A)$. Let $\text{IN}(g, A, m) := m$ and let $\text{IN}(g, A) : \{0, \ldots, m\} \rightarrow \{0, \ldots, m\} : i \mapsto \text{IN}(g, A, i)$.

CIRCUIT EVALUATION PROBLEM:
INPUT: A circuit $C$ over a Monoid $M$ and a $m \in M$.
QUESTION: Is $\text{eval}(C) = m$?

PARTLY-COMPRESSED-SUBWORD is the circuit evaluation problem for the circuit $H_L$.

PARTLY-COMPRESSED-SUBWORD can be solved in $O(|H|)$ time with $O(|g|)$ processors.
PARTLY-COMPRESSED-SUBWORD is a language recognition problem for a compressed word.

Let $m := |g|$, and let for $i \in \{0, \ldots, m - 1\}$ the number $IN(g, A, i)$ be the maximal number, such that $g[i + 1..IN(g, A, i)]$ is a subword of $\text{eval}(A)$. Let $IN(g, A, m) := m$ and let $IN(g, A) : \{0, \ldots, m\} \rightarrow \{0, \ldots, m\} : i \mapsto IN(g, A, i)$.

**CIRCUIT EVALUATION PROBLEM:**
**INPUT:** A circuit $C$ over a Monoid $M$ and a $m \in M$.
**QUESTION:** Is $\text{eval}(C) = m$?

PARTLY-COMPRESSED-SUBWORD is the circuit evaluation problem for the circuit $H_L$.

PARTLY-COMPRESSED-SUBWORD can be solved in $O(|H|)$ time with $O(|g|)$ processors.
PARTLY-COMPRESSED-SUBWORD is a language recognition problem for a compressed word.

Let $m := |g|$, and let for $i \in \{0, \ldots, m - 1\}$ the number $IN(g, A, i)$ be the maximal number, such that $g[i + 1..IN(g, A, i)]$ is a subword of $\text{eval}(A)$. Let $IN(g, A, m) := m$ and let

$IN(g, A) : \{0, \ldots, m\} \rightarrow \{0, \ldots, m\} : i \mapsto IN(g, A, i)$.

CIRCUIT EVALUATION PROBLEM:
INPUT: A circuit $C$ over a Monoid $M$ and a $m \in M$.
QUESTION: Is $\text{eval}(C) = m$?

PARTLY-COMPRESSED-SUBWORD is the circuit evaluation problem for the circuit $H_L$.

PARTLY-COMPRESSED-SUBWORD can be solved in $O(|H|)$ time with $O(|g|)$ processors.
PARTLY-COMPRESSED-SUBWORD is a language recognition problem for a compressed word.

Let $m := |g|$, and let for $i \in \{0, \ldots, m - 1\}$ the number $\text{IN}(g, A, i)$ be the maximal number, such that $g[i + 1..\text{IN}(g, A, i)]$ is a subword of $\text{eval}(A)$. Let $\text{IN}(g, A, m) := m$ and let $\text{IN}(g, A) : \{0, \ldots, m\} \rightarrow \{0, \ldots, m\} : i \mapsto \text{IN}(g, A, i)$.

**CIRCUIT EVALUATION PROBLEM:**
INPUT: A circuit $C$ over a Monoid $M$ and a $m \in M$.
QUESTION: Is $\text{eval}(C) = m$?

PARTLY-COMPRESSED-SUBWORD is the circuit evaluation problem for the circuit $H_L$.

PARTLY-COMPRESSED-SUBWORD can be solved in $O(|H|)$ time with $O(|g|)$ processors.
PARTLY-COMPRESSED-SUBWORD is a language recognition problem for a compressed word.

Let \( m := |g| \), and let for \( i \in \{0, \ldots, m - 1\} \) the number \( \text{IN}(g, A, i) \) be the maximal number, such that \( g[i + 1..\text{IN}(g, A, i)] \) is a subword of \( \text{eval}(A) \). Let \( \text{IN}(g, A, m) := m \) and let \( \text{IN}(g, A) : \{0, \ldots, m\} \rightarrow \{0, \ldots, m\} : i \mapsto \text{IN}(g, A, i) \).

**CIRCUIT EVALUATION PROBLEM:**

**INPUT:** A circuit \( C \) over a Monoid \( M \) and a \( m \in M \).

**QUESTION:** Is \( \text{eval}(C) = m \)?

PARTLY-COMPRESSED-SUBWORD is the circuit evaluation problem for the circuit \( H_L \).

PARTLY-COMPRESSED-SUBWORD can be solved in \( O(|H|) \) time with \( O(|g|) \) processors.
The circuit $H_L$ can be evaluated in $O(\log \text{treesize}(H_L))$ time with $O(|H_L| \cdot |g|)$ processors.

General Idea: Calculate the output of enough gates of $H_L$, such that the treesize of $H_L$ can be reduced to a polynomial size.
The circuit $H_L$ can be evaluated in $O(\log \text{treesize}(H_L))$ time with $O(|H_L| \cdot |g|)$ processors.

General Idea: Calculate the output of enough gates of $H_L$, such that the treesize of $H_L$ can be reduced to a polynomial size.
Let $A$ be a nonterminal symbol of $H$, and let $U$ be the set of all characters, that appear in $\text{eval}(A)$.

There is a maximal natural number $\text{scat}(A)$, such that all words $w \in U^*$ with $w \leq \text{scat}(A)$ are subwords of $\text{eval}(A)$.

If $\text{scat}(A) \geq |g|$, then $\text{IN}(g, A)$ can be calculated in $O(\log |g|)$ time with $O(|g|)$ processors.

Goal: find a lower bound for $\text{scat}(A)$. 
Let $A$ be a nonterminal symbol of $H$, and let $U$ be the set of all characters, that appear in $\text{eval}(A)$.

There is a maximal natural number $\text{scat}(A)$, such that all words $w \in U^*$ with $w \leq \text{scat}(A)$ are subwords of $\text{eval}(A)$.

If $\text{scat}(A) \geq |g|$, then $\text{IN}(g, A)$ can be calculated in $O(\log |g|)$ time with $O(|g|)$ processors.

Goal: find a lower bound for $\text{scat}(A)$. 
Let $A$ be a nonterminal symbol of $H$, and let $U$ be the set of all characters, that appear in $\text{eval}(A)$.

There is a maximal natural number $\text{scat}(A)$, such that all words $w \in U^*$ with $w \leq \text{scat}(A)$ are subwords of $\text{eval}(A)$.

If $\text{scat}(A) \geq |g|$, then $\text{IN}(g, A)$ can be calculated in $O(\log |g|)$ time with $O(|g|)$ processors.

Goal: find a lower bound for $\text{scat}(A)$. 
Let $A$ be a nonterminal symbol of $H$, and let $U$ be the set of all characters, that appear in $\text{eval}(A)$.

There is a maximal natural number $\text{scat}(A)$, such that all words $w \in U^*$ with $w \leq \text{scat}(A)$ are subwords of $\text{eval}(A)$.

If $\text{scat}(A) \geq |g|$, then $\text{IN}(g, A)$ can be calculated in $O(\log |g|)$ time with $O(|g|)$ processors.

Goal: find a lower bound for $\text{scat}(A)$. 
PATHS(A) is the set of all paths $A_1A_2 \ldots A_n$ in the graph $H$ with the following properties:

- $A_1 = A$
- All characters in $U$ appear in every word $\text{eval}(A_1), \ldots, \text{eval}(A_{n-1})$.
- There is at least one character in $U$, that does not appear in the word $\text{eval}(A_n)$.

$$\text{scat}(A) \geq |\text{PATHS}(A)|/|H|$$

If $|\text{PATHS}(A)| \geq |H| \cdot |g|$ then $\text{IN}(p, A)$ can be calculated in $O(\log |g|)$ time with $O(|g|)$ processors.
PATHS(A) is the set of all paths $A_1A_2 \ldots A_n$ in the graph $H$ with the following properties:

- $A_1 = A$
- All characters in $U$ appear in every word $\text{eval}(A_1), \ldots, \text{eval}(A_{n-1})$.
- There is at least one character in $U$, that does not appear in the word $\text{eval}(A_n)$.

$\text{scat}(A) \geq |\text{PATHS}(A)|/|H|$

If $|\text{PATHS}(A)| \geq |H| \cdot |g|$ then $\text{IN}(p, A)$ can be calculated in $O(\log |g|)$ time with $O(|g|)$ processors.
PATHS(A) is the set of all paths $A_1A_2\ldots A_n$ in the graph $H$ with the following properties:

- $A_1 = A$
- All characters in $U$ appear in every word $\text{eval}(A_1), \ldots, \text{eval}(A_{n-1})$.
- There is at least one character in $U$, that does not appear in the word $\text{eval}(A_n)$.

$$\text{scat}(A) \geq |\text{PATHS}(A)|/|H|$$

If $|\text{PATHS}(A)| \geq |H| \cdot |g|$ then $\text{IN}(p, A)$ can be calculated in $O(\log |g|)$ time with $O(|g|)$ processors.
The algorithm

Calculate the transitive hull of $H$ in $O(\log^2 |H|)$ time with $O(|H|^3)$ processors.

Let $V_i$ be the set of all nonterminal symbols $A$ of $H$, such that $i$ is the number of characters, that appear in $\text{eval}(A)$. Determine the sets $\{V_i | i \in \{1, \ldots, |\Sigma|\}\}$ in $O(|\Sigma|)$ time with $O(|H|)$ processors.

Determine the set of nonterminal symbols $O := \{A \in V_H | \text{PATHS}(A) \geq |H| \cdot |g|\}$ in $O(\log |g| + \log |H|)$ time with $O(|H|)$ processors.

Calculate the monoidelements $\{\text{IN}(g, A) | A \in O \cup V_1\}$ in $O(\log |g|)$ time with $O(|g| \cdot |H|)$ processors.

Shorten $H_L$. 
The algorithm

Calculate the transitive hull of $H$ in $O(\log^2 |H|)$ time with $O(|H|^3)$ processors.

Let $V_i$ be the set of all nonterminal symbols $A$ of $H$, such that $i$ is the number of characters, that appear in $\text{eval}(A)$. Determine the sets $\{V_i| i \in \{1, \ldots, |\Sigma|\}\}$ in $O(|\Sigma|)$ time with $O(|H|)$ processors.

Determine the set of nonterminal symbols $O := \{A \in V_H| \text{PATHS}(A) \geq |H| \cdot |g|\}$ in $O(\log |g| + \log |H|)$ time with $O(|H|)$ processors.

Calculate the monoidelements $\{\text{IN}(g, A)| A \in O \cup V_1\}$ in $O(\log |g|)$ time with $O(|g| \cdot |H|)$ processors.

Shorten $H_L$. 
The algorithm

Calculate the transitive hull of $H$ in $O(\log^2 |H|)$ time with $O(|H|^3)$ processors.

Let $V_i$ be the set of all nonterminal symbols $A$ of $H$, such that $i$ is the number of characters, that appear in $\text{eval}(A)$. Determine the sets $\{V_i| i \in \{1, \ldots, |\Sigma|\}\}$ in $O(|\Sigma|)$ time with $O(|H|)$ processors.

Determine the set of nonterminal symbols $O := \{A \in V_H | \text{PATHS}(A) \geq |H| \cdot |g|\}$ in $O(\log |g| + \log |H|)$ time with $O(|H|)$ processors.

Calculate the monoidelements $\{\text{IN}(g, A)|A \in O \cup V_1\}$ in $O(\log |g|)$ time with $O(|g| \cdot |H|)$ processors.

Shorten $H_L$. 

Tim Nonner | More pattern matching problems on compressed strings
The algorithm

Calculate the transitive hull of $H$ in $O(\log^2 |H|)$ time with $O(|H|^3)$ processors.

Let $V_i$ be the set of all nonterminal symbols $A$ of $H$, such that $i$ is the number of characters, that appear in $\text{eval}(A)$. Determine the sets $\{V_i | i \in \{1, \ldots, |\Sigma|\}\}$ in $O(|\Sigma|)$ time with $O(|H|)$ processors.

Determine the set of nonterminal symbols

$O := \{A \in V_H | \text{PATHS}(A) \geq |H| \cdot |g|\}$ in $O(\log |g| + \log |H|)$ time with $O(|H|)$ processors.

Calculate the monoidelements $\{\text{IN}(g, A) | A \in O \cup V_1\}$ in $O(\log |g|)$ time with $O(|g| \cdot |H|)$ processors.

Shorten $H_L$. 

Tim Nonner More pattern matching problems on compressed strings
The algorithm

Calculate the transitive hull of $H$ in $O(\log^2 |H|)$ time with $O(|H|^3)$ processors.

Let $V_i$ be the set of all nonterminal symbols $A$ of $H$, such that $i$ is the number of characters, that appear in $\text{eval}(A)$. Determine the sets $\{V_i | i \in \{1, \ldots, |\Sigma|\}\}$ in $O(|\Sigma|)$ time with $O(|H|)$ processors.

Determine the set of nonterminal symbols $O := \{A \in \mathcal{V}_H | \text{PATHS}(A) \geq |H| \cdot |g|\}$ in $O(\log |g| + \log |H|)$ time with $O(|H|)$ processors.

Calculate the monoid elements $\{\text{IN}(g, A) | A \in O \cup V_1\}$ in $O(\log |g|)$ time with $O(|g| \cdot |H|)$ processors.

Shorten $H_L$. 
treesize\((H_L) < (|g| \cdot |H|)^{|\Sigma|}

The circuit \(H_L\) can then be evaluated in \(O(|\Sigma| \cdot (\log |g| + \log |H|))\) time with \(O(|H| \cdot |g|)\) processors.
treesize($H_L$) < ($|g| \cdot |H|$)$^{|\Sigma|}$

The circuit $H_L$ can then be evaluated in $O(|\Sigma| \cdot (\log |g| + \log |H|))$ time with $O(|H| \cdot |g|)$ processors.
If the size of alphabet is polylogarithmically bounded and the transitive hull of $H$ is known, then \textsc{Partly-compressed-subword} can be solved nearly optimally.
Any questions?