Complexity of Certain Query Answering on Hyperstreams

Momar Sakho\textsuperscript{1} - Iovka Boneva\textsuperscript{2} - Joachim Niehren\textsuperscript{1}

1: Inria Lille - Nord Europe
2: Université de Lille
Processing streams with semi-structured data

Complex event processing [Mozafari 12, Luckham 98]

XML stream processing [Suciu 04, Olteanu 06, Kay 10, Koch 07, Gauwin 09, Sebastian 15]

RDF streams [Shinavier 10]

Data-centric workflows [Abiteboul et al. 13]
From Streams to Hyperstreams

Pipelines of stream transformations

produce and consume multiple streams

composition of transformations

sharing input streams

=> Streams with reference to others : Hyperstreams

[Maneth & Seidl 15, Labath & Niehren 13]
Pipelines of streams
Pipelines of streams

Streams
y y’ y” open ends
b letters ...

y
F

b y’
E

y”
G

y

Streams
y y’ y” open ends
b letters ...

b y’
E

y”
G

y

b y’
E

y”
G

y

P

M

O
Pipelines of streams

Streams
y y' y” open ends
b letters ...

References
F E G P M O

String patterns
O = P M = F E E G = y b y’ b y’ y”
Certain Query Answers

String patterns
\[ O = y b y' b y' y '' \]

\[ b = \text{book(title:"Tom Sawyer", author:"Mark Twain")} \]
Certain Query Answers

String patterns
\[ O = y b y' b y' y'' \]

\[ b = \text{book(title:"Tom Sawyer", author:"Mark Twain")} \]

Query 1
all the books of Mark Twain
Answers on O
first b certain
second b certain
...

5
Certain Query Answers

Query 1
all the books of Mark Twain

Answers on O
  first β certain
  second β certain
  ...

Query 2
last book of Mark Twain

Answers on O
  none certain
  second β alive candidate
  ...

Non-answers on O
  first β certain
  ...

String patterns
\[ O = y \beta y' \beta y'' \]

\[ \beta = \text{book(title:"Tom Sawyer", author:"Mark Twain")} \]
Question

How difficult it is to decide given

• a query Q
• a hyperstream D
• a position $\pi$ of pattern of D,

wether $\pi$ is

• certain for selection by Q?
• certain for rejection by Q?
Restrictions

Hyperstreams

describe words (not trees, nor graphs)

no data values (finite alphabet)

Queries defined by finite automata

obtained from logical queries

motivated by XPath but on words
Known results for streams

- **Selection**
  - NFA queries: \(\text{PSPACE-c}\)
  - DFA queries: \(\text{PTIME}\)

- **Rejection**
  - NFA queries: \(\text{PTIME}\)
  - DFA queries: \(\text{PTIME}\)

[Gauwin 09]
Contribution for hyperstreams

<table>
<thead>
<tr>
<th>Selection</th>
<th>NFA queries</th>
<th>DFA queries</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSPACE-c</td>
<td>PSPACE-c</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rejection</th>
<th>NFA queries</th>
<th>DFA queries</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSPACE-c</td>
<td>PSPACE-c</td>
</tr>
</tbody>
</table>
Contribution for hyperstreams

<table>
<thead>
<tr>
<th></th>
<th>NFA queries</th>
<th>DFA queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selection</td>
<td>PSPACE-c</td>
<td>PSPACE-c</td>
</tr>
<tr>
<td>Rejection</td>
<td>PSPACE-c</td>
<td>PSPACE-c</td>
</tr>
</tbody>
</table>

But not everything is lost...
Contributions for Linear Hyperstreams

<table>
<thead>
<tr>
<th></th>
<th>NFA queries</th>
<th>DFA queries</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Selection</strong></td>
<td>PSPACE-c</td>
<td>PTIME</td>
</tr>
<tr>
<td><strong>Rejection</strong></td>
<td>PTIME</td>
<td>PTIME</td>
</tr>
</tbody>
</table>
Queries on strings

Finite alphabet

\[ a, b, c \in \Sigma \]

Strings

\[ w \in \Sigma^* \]

Boolean queries

\[ Q \subseteq \Sigma^* \]

Queries can be defined by NFAs or DFAs \( A \)

\[ Q = L(A) \]
String patterns

String variables

\[ y \in Y \]

String Patterns

\[ p \in (\Sigma \cup Y)^* \]

Instance

\[ \text{Inst}(p) = \{p\sigma | \sigma: Y \rightarrow \Sigma^*\} \]
String patterns

String variables

\[ y \in Y \]

String Patterns

\[ p \in (\Sigma \cup Y)^* \]

Instance

\[ \text{Inst}(p) = \{ p\sigma \mid \sigma: Y\rightarrow\Sigma^* \} \]

\[ \text{aby}_1\text{bay}_2\text{bby}_1 \]

\[ \text{abaababbbbaa} \]
String patterns

String variables

\[ y \in Y \]

String Patterns

\[ p \in (\Sigma \cup Y)^* \]

Instance

\[ \text{Inst}(p) = \{ p\sigma \mid \sigma : Y \rightarrow \Sigma^* \} \]
String patterns

String variables

\[ y \in Y \]

String Patterns

\[ p \in (\Sigma \cup Y)^* \]

Instance

\[ \text{Inst}(p) = \{ p\sigma \mid \sigma: Y \rightarrow \Sigma^* \} \]

Diagram:

- \( \text{aby}_1 \text{bay}_2 \text{bby}_1 \)
- \( \text{abaababbbbaa} \)
- \( Y_1 \rightarrow \text{aa} \)
- \( Y_2 \rightarrow \text{bb} \)
String patterns

String variables

\[ y \in Y \]

String Patterns

\[ p \in (\Sigma \cup Y)^* \]

Instance

\[ \text{Inst}(p) = \{ p\sigma \mid \sigma : Y \rightarrow \Sigma^* \} \]

Theorem (folklore)

String pattern matching is NP-complete, i.e. whether \( w \in \text{Inst}(p) \)
Hyperstreams

DAGs

leaves are streams

outgoing edges are ordered

inner nodes are called references

String pattern

\[ D = \]

pat(D) = bbayayby'
Certainty

Given a boolean query $Q$, a hyperstream $D$ is called:

- **certain for selection** on $D$ if $\text{Inst}(\text{pat}(D)) \subseteq Q$

- **certain for rejection** on $D$ if $\text{Inst}(\text{pat}(D)) \cap Q = \emptyset$
Certainty

Given a boolean query $Q$, a hyperstream $D$ is called:

- **certain for selection** on $D$ if $\text{Inst}(\text{pat}(D)) \subseteq Q$

- **certain for rejection** on $D$ if $\text{Inst}(\text{pat}(D)) \cap Q = \emptyset$

**Theorem**

Certainty for selection (resp. rejection) on hyperstreams (resp. string patterns) is PSPACE-complete for queries defined by NFA (resp. DFA).
**PSPACE hardness**

**Proof**
- for string patterns (without compression),
- for DFAs (without nondeterminism)

**Thm** (folklore)
- Emptiness of intersection of a list of DFAs is PSPACE-complete.
PSPACE hardness

Proof
for string patterns (without compression),
for DFAs (without nondeterminism)

Thm (folklore)
Emptiness of intersection of a list of DFAs is PSPACE-complete.

Reduction
DFAs $A_1, \ldots, A_n$
$L(A_1) \cap \cdots \cap L(A_n) = \emptyset$
iff
$\text{Inst}(y\# \cdots \# y) \cap L(A_1)\# \cdots \# L(A_n) = \emptyset$ Rejection
PSPACE completeness

Let D be a hyperstream and A an NFA

Hyperstream D is certain for selection for query L(A)

iff for all substitutions $\sigma$ from string variables to A-inhabited transitions: $\text{eval}^A(D, \sigma) \cap (Q_{\text{init}} \times Q_{\text{final}}) \neq \emptyset$

All these substitutions can be enumerated and tested in PSPACE.
An efficient fragment

**Theorem**
Certainty for selection (resp. rejection) on *linear* hyperstreams (resp. on string patterns) for DFA queries is in PTIME.
An efficient fragment

**Theorem**
Certainty for selection (resp. rejection) on **linear** hyperstreams (resp. on string patterns) for DFA queries is in PTIME.

**Proof:**
- Inst(pat(D)) can be recognized by an NFA $A'$ of size linear in $|D|$
- Inclusion $L(A') \subseteq L(A)$ can be decided in time $O(|A'||A|)$ since $A$ is deterministic
- Non empty intersection $L(A') \cap L(A) = \emptyset$ is in PTIME (even for NFAs)
Non boolean queries

- All results remain true in the general case
- One more difficulty in the proof of the efficient fragment
Lifting Efficiency Result for Linear Hyperstreams

**Difficulty**
Given a hyperstream $D$ and a position $\pi$ of $\text{pat}(D)$, can one uncompress $D$ in PTIME, so that the node of $D$ that represents $\pi$ is not shared.

Yes, indeed
Lifting Efficiency Result for Linear Hyperstreams

Difficulty
Given a hyperstream $D$ and a position $\pi$ of $\text{pat}(D)$, can one uncompress $D$ in PTIME, so that the node of $D$ that represents $\pi$ is not shared.

Yes, indeed

pattern: aabyaaby
Lifting Efficiency Result for Linear Hyperstreams

Difficulty

Given a hyperstream $D$ and a position $\pi$ of $\text{pat}(D)$, can one uncompress $D$ in PTIME, so that the node of $D$ that represents $\pi$ is not shared.

Yes, indeed

Unshare position $\pi=2$ with linear size increase

pattern: aabyaaby
Lifting Efficiency Result for Linear Hyperstreams

**Difficulty**
Given a hyperstream $D$ and a position $\pi$ of $\text{pat}(D)$, can one uncompress $D$ in PTIME, so that the node of $D$ that represents $\pi$ is not shared.

**Yes, indeed**

Unshare position $\pi=2$ with linear size increase

pattern: aabyaaby
Lifting Efficiency Result for Linear Hyperstreams

Difficulty
Given a hyperstream $D$ and a position $\pi$ of $\text{pat}(D)$, can one uncompress $D$ in PTIME, so that the node of $D$ that represents $\pi$ is not shared.

Yes, indeed

pattern: aabaaby
Conclusions

Complexity of certain query answering increases with hyperstreams.

Still, efficient query answering on hyperstreams may be possible in practice.

- needs incremental algorithm maintaining alive answer candidates
- hope for lowest latency for DFA queries on linear hyperstreams

Extensions needed for hyperstreams with semi-structured data:

- JSON
- RDF

Extensions needed for hyperstreaming query-based programs.
Questions ?