What do you do if your data fail your specification?

Repair your data.
What do you do if your data fail your specification?

Repair your data.
What do you do if your data fail your specification?

Different ways of repairing data:

Off-line

Streaming
Can we **streaming-repair** each XML document with an **uniform** number of edits?

**Definition (informal)**

Given XML specifications $\mathcal{R}$ (restriction) and $\mathcal{T}$ (target), determine if there exist a **streaming repair process** $S : L(\mathcal{R}) \rightarrow L(\mathcal{T})$ and an **uniform bound** $N \in \mathbb{N}$:

$$\text{cost}(t, S) \leq N \quad \text{for all XML documents } t \models \mathcal{R}.$$
Can we **streaming-repair** each XML document with an **uniform** number of edits?

**Streaming bounded repair problem**

**Example**

\[
\begin{align*}
\mathcal{R}: & \quad r \rightarrow d \cdot c^* \\
& \quad d \rightarrow a^* \cdot b^* \\
& \quad T: & \quad r \rightarrow a^* \cdot e \\
& \quad e \rightarrow b^* \cdot c^*
\end{align*}
\]

**input**: \(<r>\ <d>\ <a/>\ <a/>\ <b/>\ <b/>\ <c/>\ <c/>\</r>\)

**output**: \(<r>\ <a/>\ <a/>\ <e>\ <b/>\ <b/>\ <c/>\ <c/>\ <e/>\</r>\)
Can we **streaming-repair** each XML document with an **uniform** number of edits?

**Streaming bounded repair problem**

**Example**

\[
R_2: \quad r \rightarrow (a + b) \cdot x^* \cdot (a^* + b^*)
\]

\[
T_2: \quad r \rightarrow a \cdot x^* \cdot a^* + b \cdot x^* \cdot b^*
\]

**input:**

\[
<r> \quad <a/> \quad <x/> \quad <x/> \quad <x/> \quad <a/> \quad <a/> \quad <a/> \quad <a/> \quad <a/> \quad </r>
\]

**output:**

\[
<r> \quad <b/> \quad <x/> \quad <x/> \quad <x/> \quad <b/> \quad <b/> \quad <b/> \quad <b/> \quad <b/> \quad </r>
\]
Summary of main results in the paper

- **Effective characterization** for the streaming bounded repair problem.
  - For DTDs and XML Schemas (deterministic top-down tree automata).
  - Based on a stack game between two players.

- **Precise complexity** of the streaming bounded repair problem.
  - EXPTIME-complete.
  - An exponential gap between the word and tree case.
Which DTDs are streaming bounded repairable?

Cristian Riveros
University of Oxford

Pierre Bourhis
University of Oxford

Gabriele Puppis
CNRS/LaBRI Bordeaux

ICDT 2013
Outline

Setting

Streaming problem

Main characterization

Complexity
Outline

Setting

Streaming problem

Main characterization

Complexity
Trees and their XML-encoding

Unranked trees over $\Sigma$

```
\begin{align*}
    & t : \text{person} \\
    & \quad \text{name} \\
    & \quad \quad \text{Chris} \\
    & \quad \text{address} \\
    & \quad \quad \text{str} \\
    & \quad \quad \quad \text{Road} \\
    & \quad \quad \quad \text{num} \\
    & \quad \quad \quad \quad \text{369}
\end{align*}
```

XML encoding

```
\hat{t} : \langle \text{person} \rangle \\
\quad \langle \text{name} \rangle \text{Chris} \langle \text{name} \rangle \\
\quad \langle \text{address} \rangle \\
\quad \quad \langle \text{str} \rangle \text{Road} \langle \text{str} \rangle \\
\quad \quad \langle \text{num} \rangle \text{369} \langle \text{num} \rangle \\
\quad \langle \text{address} \rangle \\
\quad \langle \text{person} \rangle
```

- XML specification $\mathcal{A}$ (e.g. XML Schema or unranked tree automata)

$$L(\mathcal{A}) = \{ t \in \text{Trees} \mid t \models \mathcal{A} \}$$

$$Docs(\mathcal{A}) = \{ \hat{t} \in \text{XML} \mid t \models \mathcal{A} \}$$
Streaming transducers for repairing XML documents

■ A repair strategy is a function \( f : L(\mathcal{R}) \rightarrow L(\mathcal{T}) \).

■ A streaming repair strategy is a function \( S : \text{Docs}(\mathcal{R}) \rightarrow \text{Docs}(\mathcal{T}) \):
  
  ▶ \( S \) is specified by a sequential transducer.
  ▶ \( S \) could have infinite memory.

■ Cost of a streaming repair strategy \( S \) over \( \hat{t} = a_1 \ldots a_n \):

\[
\text{cost}(\hat{t}, S) = \sum_{i=1}^{n} \text{dist}(a_i, u_i)
\]

where \( u_i \) is the output of \( S \) after reading \( a_i \).
Outline

Setting

Streaming problem

Main characterization

Complexity
Streaming bounded repair problem

Definition

Given XML specifications $\mathcal{R}$ and $\mathcal{T}$, determine if there exists a streaming repair strategy $S : Docs(\mathcal{R}) \rightarrow Docs(\mathcal{T})$ and an uniform bound $N \in \mathbb{N}$:

$$cost(\hat{t}, S) \leq N \quad \forall \hat{t} \in Docs(\mathcal{R})$$
We have studied this problem over words and (non-streaming) trees


Main ideas previous papers:

Similar approach does NOT work for the streaming case in general!
Deterministic top-down tree automata

Definition

A deterministic top-down tree automaton (DTT-automata) is a tuple:

\[ \mathcal{A} = (\Sigma, Q, \delta, q_0, F) \]

- \( \delta : Q \times \Sigma \rightarrow Q \times Q \) is the transition function,
- \( q_0 \) is the initial state, and \( F \subseteq Q \) is the final set of states.

Example

\[ R : \begin{align*}
    r &\rightarrow cb^* \\
    c &\rightarrow a^*
\end{align*} \]

\[ \mathcal{R} : \begin{align*}
    \delta(q_0, r) &= (q_c, q_f) \\
    \delta(q_c, c) &= (q_a, q_b) \\
    \delta(q_a, a) &= (q_f, q_a) \\
    \delta(q_b, b) &= (q_f, q_b)
\end{align*} \]
Deterministic top-down tree automata

Definition

A deterministic top-down tree automaton (DTT-automata) is a tuple:

\[ \mathcal{A} = (\Sigma, Q, \delta, q_0, F) \]

- \( \delta : Q \times \Sigma \rightarrow Q \times Q \) is the transition function,
- \( q_0 \) is the initial state, and \( F \subseteq Q \) is the final set of states.

DTT-automata are more expressive than DTDs or XML Schema.
Outline

Setting

Streaming problem

Main characterization

Complexity
Main ideas of the characterization

1. Transition graph of $\mathcal{R}$ and $\mathcal{T}$.

2. Cyclic behavior: Strongly connected components.

3. Stack game between Generator and Repairer.
   - Following the preorder traversal of the graph (stacks are needed).
Cyclic behavior of DTT-automata (components)

Definition

Given $\mathcal{A} = (\Sigma, Q, \delta, q_0, F)$, the transition graph of $\mathcal{A}$ is the graph $G_{\mathcal{A}} = (Q, E_h \cup E_v)$ such that for every $\delta(q, a) = (q_1, q_2)$:

- SCC$(\mathcal{A})$ is the set of strongly connected component $X$ of $G_{\mathcal{A}}$.
- $L(\mathcal{A} \mid X) = \{ C \in \text{Context}_\Sigma \mid \exists p, q \in X : \delta(p, C) = q \}$

The cyclic behavior of $X_1$ is contained in the cyclic behavior of $X_2$ if $L(\mathcal{A} \mid X_1) \subseteq L(\mathcal{A} \mid X_2)$, then the cyclic behaviour of $X_1$ is contained in the cyclic behaviour of $X_2$. 
Stacks over strongly connected components

- (Prefix rewriting systems).
- Stack alphabets: SCC(\(R\)) and SCC(\(T\)).
- Rules of the form:

  push: \(X \mapsto X_1 X_2 \quad \Rightarrow \quad X \cdot w \Rightarrow^A X_1 \cdot X_2 \cdot w\)

  pop: \(X \mapsto \epsilon \quad \Rightarrow \quad X \cdot w \Rightarrow^A w\)

- Two prefix-rewriting systems: Stack(\(R\)) and Stack\(^*\)(\(T\))

\[
X \mapsto X_1 X_2 \in \text{Stack}(R) \quad \text{iff} \quad \delta(p, a) = (p_1, p_2) \quad \exists p \in X, p_1 \in X_1, p_2 \in X_2
X_1 \neq X \land X_2 \neq X
\]

X \mapsto \epsilon \in \text{Stack}(R) \quad \text{always}

\[
Y \mapsto Y_1 Y_2 \in \text{Stack}^*(T) \quad \text{iff} \quad \delta'(q, a) = (q_1, q_2) \quad \exists q \in Y, q_1 \in Y_1, q_2 \in Y_2
\]

Y \mapsto \epsilon \in \text{Stack}^*(T) \quad \text{always}

where \(X, X_1, X_2 \in \text{SCC}(R)\) and \(Y, Y_1, Y_2 \in \text{SCC}(T)\).
Stack-game between Generator and Repairer

- Given $\mathcal{R}$ and $\mathcal{T}$ we define a turn-based game $\mathcal{M}(\mathcal{R}, \mathcal{T})$.

- Two players: Generator and Repairer.
  - Generator plays over Stack($\mathcal{R}$).
  - Repairer plays over Stack($\mathcal{T}$).

\[
L(\mathcal{R} | X_0) \notin L(\mathcal{T} | Y_0) \\
L(\mathcal{R} | X_2) \notin L(\mathcal{T} | Y_2) \\
L(\mathcal{R} | X_3) \notin L(\mathcal{T} | Y_2)
\]
Main characterization

Theorem

\( L(\mathcal{R}) \) is streaming bounded repairable into \( L(\mathcal{T}) \) iff

Repairer has a winning strategy in \( \mathcal{M}(\mathcal{R}, \mathcal{T}) \).

Details of the proof: read the paper.
Outline

Setting

Streaming problem

Main characterization

Complexity
Complexity of the streaming bounded repair problem

Stack(\mathcal{R}):
- Non-recursive.
- Stacks are of polynomial size.

Stack* (\mathcal{T}):
- Stacks are of unbounded size (can be bounded by a polynomial).

Theorem
The streaming bounded repair problem for DTT-automata is EXPTIME-complete.

For deterministic word and tree automata:

<table>
<thead>
<tr>
<th></th>
<th>non-streaming</th>
<th>streaming</th>
</tr>
</thead>
<tbody>
<tr>
<td>words</td>
<td>coNP</td>
<td>PTIME</td>
</tr>
<tr>
<td>trees</td>
<td>coNEXPTIME</td>
<td>EXPTIME</td>
</tr>
</tbody>
</table>
Concluding remarks

**Effective characterization** for the streaming bounded repair problem.

- Only for DTT-automata (e.g. DTDs and XML Schemas).
- EXPTIME-complete for DTT-automata.

Open problems:

- Characterization in the general case (regular tree languages).
- Amount of memory needed for the streaming strategy.
Which DTDs are streaming bounded repairable?

Cristian Riveros  
University of Oxford

Pierre Bourhis  
University of Oxford

Gabriele Puppis  
CNRS/LaBRI Bordeaux

ICDT 2013