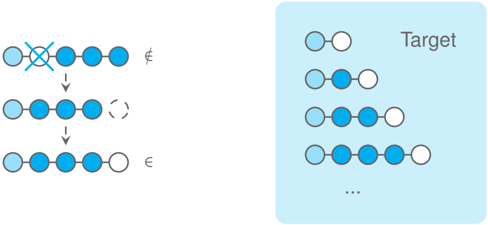
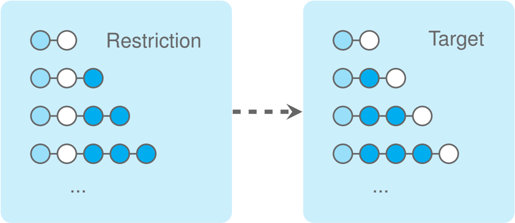
# What do you do if your data fail your specification?



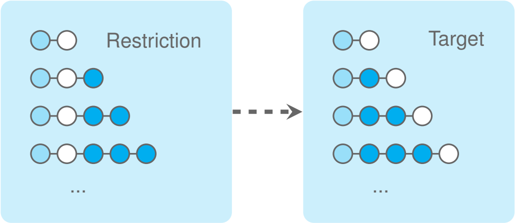

**Repair your data.**

# What do you do if your data fail your specification?

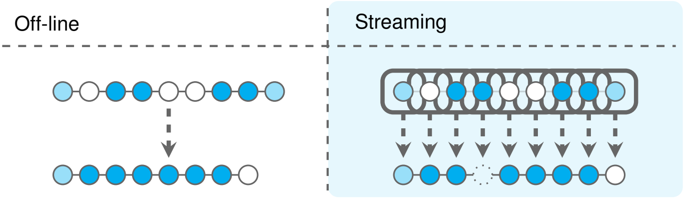



**Repair your data.**

# What do you do if your data fail your specification?




Different ways of repairing data:



# 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 $S : L(\mathcal{R}) \to L(\mathcal{T})$ and an uniform bound $N \in \mathbb{N}$:

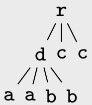
$$\mathrm{cost}(t, S) \leq N \qquad \text{for all XML documents } t \vDash \mathcal{R}.$$

Streaming bounded repair problem

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

**Streaming bounded repair problem**

**Example**

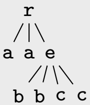


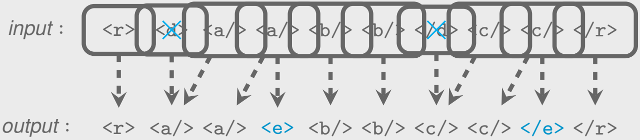

$\mathcal{R}$: $\mathtt{r} \to \mathtt{d} \cdot \mathtt{c}^*$

$\mathtt{d} \to \mathtt{a}^* \cdot \mathtt{b}^*$

$\mathcal{T}$: $\mathtt{r} \to \mathtt{a}^* \cdot \mathtt{e}$

$\mathtt{e} \to \mathtt{b}^* \cdot \mathtt{c}^*$

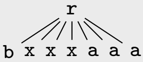


# Can we streaming-repair each XML document with an uniform number of edits?

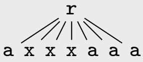
**Streaming bounded repair problem**

## Example




$\mathcal{R}_2$: $\mathtt{r} \rightarrow (\mathtt{a} + \mathtt{b}) \cdot \mathtt{x}^* \cdot (\mathtt{a}^* + \mathtt{b}^*)$

$\mathcal{T}_2$: $\mathtt{r} \rightarrow \mathtt{a} \cdot \mathtt{x}^* \cdot \mathtt{a}^* + \mathtt{b}^* \cdot \mathtt{x}^* \cdot \mathtt{b}^*$




---

*input* : <r> <a/> <x/> <x/> <x/> <a/> <a/> <a/> <a/> </r>

*output* : <r> <b/> <x/> <x/> <x/> <b/> <b/> <b/> <b/> </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

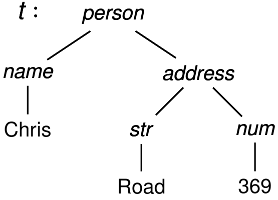
Complexity

# Outline

# Trees and their XML-encoding

## Unranked trees over $\Sigma$

$t$ :




## XML encoding

$\hat{t}$ :

```
<person>
    <name> Chris </name>
    <address>
        <str> Road </str>
        <num> 369 </num>
    </address>
</person>
```

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

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

$$\mathit{Docs}(\mathcal{A}) = \{\hat{t} \in \text{XML} \mid t \models \mathcal{A}\}$$

# Streaming transducers for repairing XML documents

- A repair strategy is a function $f : L(\mathcal{R}) \to L(\mathcal{T})$.
- A streaming repair strategy is a function $\mathcal{S} : Docs(\mathcal{R}) \to Docs(\mathcal{T})$:
  - $\mathcal{S}$ is specified by a sequential transducer.
  - $\mathcal{S}$ could have infinite memory.
- Cost of a streaming repair strategy $\mathcal{S}$ over $\hat{t} = a_1 \ldots a_n$:

$$cost(\hat{t}, \mathcal{S}) = \sum_{i=1}^{n} \mathrm{dist}(a_i, u_i)$$

where $u_i$ is the output of $\mathcal{S}$ after reading $a_i$.

# Outline

# Streaming bounded repair problem

## Definition

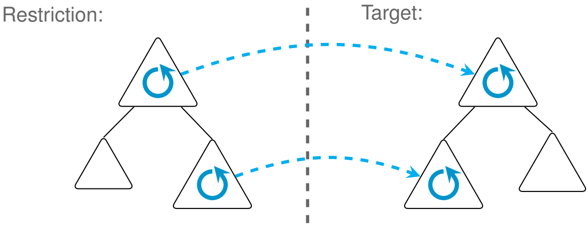
Given XML specifications $\mathcal{R}$ and $\mathcal{T}$, determine if there exists a streaming repair strategy $\mathcal{S} : \mathit{Docs}(\mathcal{R}) \to \mathit{Docs}(\mathcal{T})$ and an uniform bound $N \in \mathbb{N}$:

$$\mathrm{cost}(\hat{t}, \mathcal{S}) \leq N \qquad \forall\, \hat{t} \in \mathit{Docs}(\mathcal{R})$$

# We have studied this problem over words and (non-streaming) trees

1. "Regular repair of specifications", in LICS 2011.
2. "Bounded repairability for regular tree languages", in ICDT 2012.

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:

$$\mathcal{A} \;=\; (\Sigma, Q, \delta, q_0, F)$$

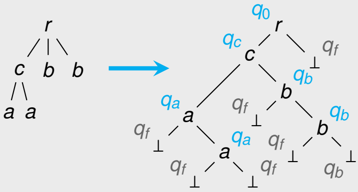
- $\delta : Q \times \Sigma \rightarrow Q \times Q$ is the transition function,
- $q_0$ is the initial state, and $F \subseteq Q$ is the final set of states.

DTT-automata over the **first-child-next-sibling** encoding.

## Example

$$R: \quad \begin{aligned} \mathtt{r} &\rightarrow \mathtt{cb}^* \\ \mathtt{c} &\rightarrow a^* \end{aligned}$$

$$\mathcal{R}: \quad \begin{aligned} \delta(q_0, r) &= (q_c, \underline{q_f}) \\ \delta(q_c, c) &= (\underline{q_a}, \underline{q_b}) \\ \delta(\underline{q_a}, a) &= (q_f, \underline{q_a}) \\ \delta(\underline{q_b}, b) &= (q_f, \underline{q_b}) \end{aligned}$$



# Deterministic top-down tree automata

## Definition

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

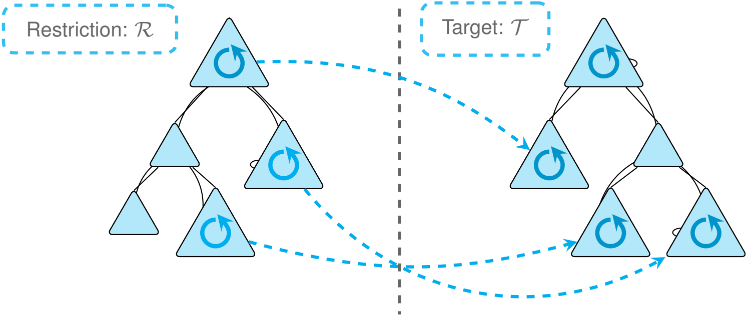
$$\mathcal{A} = (\Sigma, Q, \delta, q_0, F)$$

- $\delta : Q \times \Sigma \rightarrow Q \times Q$ 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

# Main ideas of the characterization

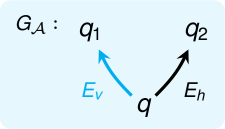



1. Transition graph of $\mathcal{R}$ and $\mathcal{T}$.
2. Cyclic behavior: Strongly connected components.
3. Stack game between Generator and Repairer.
   - Following the preorder traversal of the graph (stacks are needed).

# Cyclic behavior of DTT-automata (components)

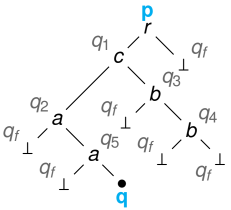
## Definition

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 $\delta(q, a) = (q_1, q_2)$:




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

$\delta(p, C) = q$ iff




$\Rightarrow$ Context $C$

$L(\mathcal{A} \mid X_1) \subseteq L(\mathcal{A} \mid X_2)$, 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: $\mathrm{SCC}(\mathcal{R})$ and $\mathrm{SCC}(\mathcal{T})$.
- Rules of the form:

$$
\begin{array}{llcl}
\textbf{push:} & X \mapsto X_1 X_2 & \dashrightarrow & X \cdot w \overset{\mathcal{A}}{\Rightarrow} X_1 \cdot X_2 \cdot w \\
\textbf{pop:} & X \mapsto \epsilon & \dashrightarrow & X \cdot w \overset{\mathcal{A}}{\Rightarrow} w
\end{array}
$$

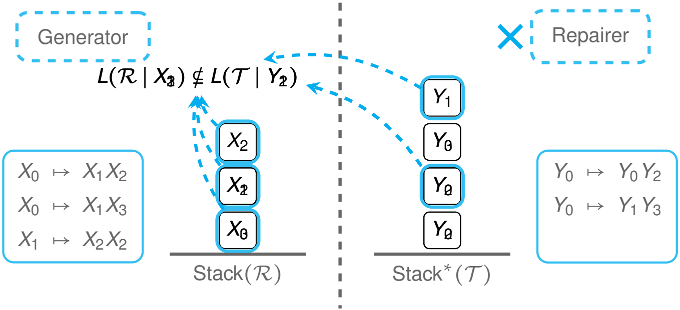
- Two prefix-rewriting systems: $\mathrm{Stack}(\mathcal{R})$ and $\mathrm{Stack}^*(\mathcal{T})$

$$
\begin{array}{rll}
X \mapsto X_1 X_2 \in \mathrm{Stack}(\mathcal{R}) & \text{iff} & \delta(p,a) = (p_1, p_2)\ \exists p \in X, p_1 \in X_1, p_2 \in X_2 \\
 & & X_1 \neq X \ \wedge \ X_2 \neq X \\
X \mapsto \epsilon \in \mathrm{Stack}(\mathcal{R}) & \text{always} & \\
\hline
Y \mapsto Y_1 Y_2 \in \mathrm{Stack}^*(\mathcal{T}) & \text{iff} & \delta'(q,a) = (q_1, q_2)\ \exists q \in Y, q_1 \in Y_1, q_2 \in Y_2 \\
Y \mapsto \epsilon \in \mathrm{Stack}^*(\mathcal{T}) & \text{always} &
\end{array}
$$

where $X, X_1, X_2 \in \mathrm{SCC}(\mathcal{R})$ and $Y, Y_1, Y_2 \in \mathrm{SCC}(\mathcal{T})$.

# 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$^*(\mathcal{T})$.



# 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

# 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:

| | **non-streaming** | **streaming** |
|---|---|---|
| **words** | coNP | PTIME |
| **trees** | coNEXPTIME | EXPTIME |

# 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