What do you do when a computational object fails a specification?

Source

1000
10000
100000
...

Target

1050
10050
100050
...

Worst-case cost of repairing source into target:

\[ \max_{s \in S} \min_{t \in T} \text{dist}(s, t) \]

Depends on distance (e.g., Levenshtein distance)

Can be infinite!
What do you do when a computational object fails a specification?

Worst-case cost of repairing source into target:

$$\max_{s \in S} \min_{t \in T} \text{dist}(s, t)$$
What do you do when a computational object fails a specification?

Worst-case cost of repairing source into target:

\[ \max_{s \in S} \min_{t \in T} \text{dist}(s, t) \]

 Depends on distance (e.g., Levenstein distance)
What do you do when a computational object fails a specification?

Worst-case cost of repairing source into target:

\[
\max_{s \in S} \min_{t \in T} \text{dist}(s, t)
\]

- Depends on distance (e.g., Levenstein distance)
- Can be infinite!
Plan

A. Bounded repairability of regular word languages
   1) characterization
   2) streaming setting
   3) complexity

B. Bounded repairability of regular tree languages
   1) curry encodings, stepwise automata, contexts
   2) characterization
   3) complexity
Part A. Problem setting:

- Given two languages $S \subseteq \Sigma^*$ and $T \subseteq \Delta^*$ (represented by finite state automata)

- Decide whether $\max_{s \in S} \min_{t \in T} \text{dist}(s, t)$ is finite.
Part A. Problem setting:

- Given two languages $S \subseteq \Sigma^*$ and $T \subseteq \Delta^*$ (represented by finite state automata)

- Decide whether $\max_{s \in S} \min_{t \in T} \text{dist}(s, t)$ is finite.

Examples

- $10^* \text{ is bounded repairable into } 10^*50$

- $10^* \text{ is not bounded repairable into } (10)^*$

- $(1 + 0)^* \text{ is not bounded repairable into } (1 + 0^*5)^*$
Rule of thumb: “If you need to edit, you’d better do it outside a loop!”
Rule of thumb: "If you need to edit, you’d better do it outside a loop! "

Source automaton

Target automaton
Rule of thumb: “If you need to edit, you’d better do it outside a loop!”

For any strategy that repairs traces of X into traces of Y:

1. either \( \text{traces}(X) \subseteq \text{traces}(Y) \)
2. or the strategy has unbounded cost.
Characterization of bounded repairability of word languages

$S$ is repairable into $T$ with uniformly bounded cost

\[ \uparrow \]

Given some (trimmed) automata for $S$ and $T$ and the DAGs of strongly connected components...

Source DAG

Target DAG
Characterization of bounded repairability of word languages

$S$ is repairable into $T$ with uniformly bounded cost

Given some (trimmed) automata for $S$ and $T$ and the DAGs of strongly connected components...

...every chain of components in the source is covered by a chain of components in the target.
Characterization of bounded repairability of word languages

$S$ is repairable into $T$ with uniformly bounded cost

$\uparrow$

Given some (trimmed) automata for $S$ and $T$ and the DAGs of strongly connected components...

...every chain of components in the source is covered by a chain of components in the target.
An example

All chains of source DAG are covered by chains of target DAG
⇒ $S$ is repairable into $T$ with uniformly bounded cost.

$S = 30*1^* + 30*2^*$

$T = 10*1^* + 20*2^*$
An example

All chains of source DAG are covered by chains of target DAG
⇒ $S$ is repairable into $T$ with uniformly bounded cost.

$S = 30 \cdot 1^* + 30 \cdot 2^*$

$T = 10 \cdot 1^* + 20 \cdot 2^*$
An example

All chains of source DAG are covered by chains of target DAG ⇒ \( S \) is repairable into \( T \) with uniformly bounded cost.

\[
S = 30*1^* + 30*2^* \\
T = 10*1^* + 20*2^*
\]
An example

All chains of source DAG are covered by chains of target DAG
⇒ $S$ is repairable into $T$ with uniformly bounded cost.

$S = 30*1^* + 30*2^*$

$T = 10*1^* + 20*2^*$

There is no covering relation compatible with prefixes
⇒ the repair strategy is not streaming
(i.e. implementable by a sequential transducer)
An example

All chains of source DAG are covered by chains of target DAG
⇒ \( S \) is repairable into \( T \) with uniformly bounded cost.

\[
S = 30 \cdot 1^* + 30 \cdot 2^*
\]

\[
T = 10 \cdot 1^* + 20 \cdot 2^*
\]

There is no covering relation compatible with prefixes
⇒ the repair strategy is not streaming
   (i.e. implementable by a sequential transducer)
Complexity of **non-streaming** bounded repairability problem:

<table>
<thead>
<tr>
<th></th>
<th>fixed</th>
<th>DFA</th>
<th>NFA</th>
</tr>
</thead>
<tbody>
<tr>
<td>fixed</td>
<td>CONST</td>
<td>P</td>
<td>PSPACE</td>
</tr>
<tr>
<td>DFA</td>
<td>P</td>
<td>coNP</td>
<td>PSPACE</td>
</tr>
<tr>
<td>NFA</td>
<td>PTIME</td>
<td>coNP</td>
<td>PSPACE</td>
</tr>
</tbody>
</table>
### Complexity of non-streaming bounded repairability problem:

<table>
<thead>
<tr>
<th></th>
<th>fixed</th>
<th>DFA</th>
<th>NFA</th>
</tr>
</thead>
<tbody>
<tr>
<td>fixed</td>
<td>CONST</td>
<td>P</td>
<td>PSPACE</td>
</tr>
<tr>
<td>DFA</td>
<td>P</td>
<td>coNP</td>
<td>PSPACE</td>
</tr>
<tr>
<td>NFA</td>
<td>PTIME</td>
<td>coNP</td>
<td>PSPACE</td>
</tr>
</tbody>
</table>

### Complexity of streaming bounded repairability problem:

<table>
<thead>
<tr>
<th></th>
<th>fixed</th>
<th>DFA</th>
<th>NFA</th>
</tr>
</thead>
<tbody>
<tr>
<td>fixed</td>
<td>CONST</td>
<td>P</td>
<td>PSPACE</td>
</tr>
<tr>
<td>DFA</td>
<td>P</td>
<td>P</td>
<td>PSPACE</td>
</tr>
<tr>
<td>NFA</td>
<td>≤ PSPACE ≥ P</td>
<td>≤ PSPACE ≥ P</td>
<td>≤ EXP ≥ PSPACE</td>
</tr>
</tbody>
</table>
Part B. New tools for a more general setting...

<table>
<thead>
<tr>
<th>Languages of words:</th>
<th>Languages of unranked trees:</th>
</tr>
</thead>
<tbody>
<tr>
<td>- insersions / deletions</td>
<td>- insertions / deletions</td>
</tr>
<tr>
<td>- finite state automata</td>
<td>- stepwise tree automata</td>
</tr>
<tr>
<td>- components &amp; traces</td>
<td>- components &amp; contexts</td>
</tr>
<tr>
<td>- coverability of chains</td>
<td>- coverability of synopsis trees</td>
</tr>
</tbody>
</table>
Edit operations on unranked trees: deletions
Edit operations on unranked trees: deletions
Edit operations on unranked trees: insertions
Edit operations on unranked trees: insertions
An example

Source

Target
An example

Source

Target
An example
An example
Bottom-up automata on ranked (binary) trees:

How to parse unranked trees?
Bottom-up automata on ranked (binary) trees:

How to parse unranked trees?

Encode them using binary trees!
The curry encoding
The curry encoding

\[
\begin{array}{c}
1 \rightarrow 2 \\
\end{array}
\]

\[
\begin{array}{c}
0 \rightarrow 1 \\
\end{array}
\]

\[
\begin{array}{c}
2 \\
\end{array}
\]

\[
\begin{array}{c}
0 \quad 1 \\
\end{array}
\]

\[
\begin{array}{c}
2 \\
\end{array}
\]

\[
\Rightarrow
\]

\[
\begin{array}{c}
0 \quad @ \\
\end{array}
\]

\[
\begin{array}{c}
1 \\
\end{array}
\]

\[
\begin{array}{c}
2 \\
\end{array}
\]

\[
\Rightarrow
\]

\[
\begin{array}{c}
0 \\
\end{array}
\]

\[
\begin{array}{c}
1 \\
\end{array}
\]

\[
\begin{array}{c}
2 \\
\end{array}
\]

\[
\Rightarrow
\]

\[
\begin{array}{c}
0 \\
\end{array}
\]

\[
\begin{array}{c}
1 \\
\end{array}
\]

\[
\begin{array}{c}
2 \\
\end{array}
\]

Stepwise automata = bottom-up on curry encodings
The curry encoding

\[
\begin{array}{c}
1 & 2 & \cdots & n \\
\end{array}
\]

\[\cong\]

\[
\begin{array}{c}
0 & \cdots & n \\
\end{array}
\]

Stepwise automata = bottom-up on curry encodings
The curry encoding
The curry encoding

\[ \begin{array}{c}
\text{1} \quad \text{2} \quad \ldots \quad \text{n} \\
\end{array} \]

\[ \cong \]

\[ \begin{array}{c}
\text{0} \quad @ \quad @ \quad \ldots \quad @ \\
\text{1} \quad 1 \quad 2 \\
\text{n} \\
\end{array} \]

\[ \text{Stepwise automata} = \text{bottom-up on curry encodings} \]
An example

\[
\begin{array}{c}
\begin{array}{c}
\text{r} \\
\vdots \\
\text{b} \\
\text{c}
\end{array}
\end{array}
\quad \Leftrightarrow \quad
\begin{array}{c}
\begin{array}{c}
\text{r} \\
\vdots \\
\text{b} \\
\text{c}
\end{array}
\end{array}
\end{array}
\]
An example

\begin{align*}
    r & \rightarrow q_r \\
    a & \rightarrow q_a \\
    b & \rightarrow q_b \\
    c & \rightarrow q_c \\
    q_r @ q_a & \rightarrow q_r \\
    q_b @ q_c & \rightarrow q_c \\
    q_r @ q_c & \rightarrow q_{\text{final}}
\end{align*}
An example

\[
\begin{align*}
\begin{array}{c}
r \\
\vdots \\
a \\
\vdots \\
a \\
b \\
\vdots \\
 b \\
c
\end{array}
\end{align*}
\] 
\[
\begin{array}{c}
\begin{array}{ccc}
m & n & \ldots & m \Rightarrow q_m \\
q_m & \Rightarrow (q_{m-1} @ m \\ q_{m-1}) \\
q_{m-1} & \Rightarrow m \\
\vdots & \vdots & \vdots \\
 q_0 & \Rightarrow a \\
\ldots & \ldots & \ldots \\
 q_{n-1} & \Rightarrow b \\
\ldots & \ldots & \ldots \\
 q_0 & \Rightarrow c \\
q_0 & \Rightarrow q_{final}
\end{array}
\end{array}
\]
An example
An example

\[
\begin{array}{c}
\text{r} \\
a \ldots a \quad b \\
\vdots \\
 b \\
 c
\end{array}
\quad \cong 
\begin{array}{c}
q_{\text{final}} \\
q_r \\
a \\
q_a \\
qr \\
q_r @ qa \\
q_r @ qc \\
\text{final}
\end{array}
\]

\[
\begin{array}{c}
r \rightarrow qr \\
a \rightarrow qa \\
b \rightarrow qb \\
c \rightarrow qc
\end{array}
\]

\[
\begin{array}{c}
q_r @ qa \rightarrow qr \\
q_b @ qc \rightarrow qc \\
q_r @ qc \rightarrow q_{\text{final}}
\end{array}
\]
Contexts = trees with holes

Contexts can be parsed between two states: $p \xrightarrow{\text{C}} q$ (accessibility of states and components are defined accordingly)
Contexts = trees with holes

\[
\begin{align*}
\text{Contexts} &= \text{trees with holes} \\
\end{align*}
\]
Contexts = trees with holes

Contexts can be parsed between two states: $p \xrightarrow{c} q$

(accessibility of states and components are defined accordingly)
Recall: a run of a finite state automaton induces a chain of components...

Likewise, a run of a stepwise automaton induces a synopsis tree (i.e. a tree of components).
Recall: a run of a finite state automaton induces a chain of components...

Likewise, a run of a stepwise automaton induces a synopsis tree (i.e. a tree of components).
Characterization of bounded repairability of tree languages

\( S \) is repairable into \( T \) with uniformly bounded cost

\[ \uparrow \Downarrow \]

Given some (trimmed) stepwise automata for \( S \) and \( T \), all synopsis trees of \( S \) are covered by synopsis trees of \( T \)

i.e. ...
Characterization of bounded repairability of tree languages

$S$ is repairable into $T$ with uniformly bounded cost

Given some (trimmed) stepwise automata for $S$ and $T$, all synopsis trees of $S$ are covered by synopsis trees of $T$

\[ \exists \lambda : \text{cyclic components} \rightarrow \text{cyclic components} \]
Given some (trimmed) stepwise automata for $S$ and $T$, all synopsis trees of $S$ are covered by synopsis trees of $T$

i.e. $\exists \lambda : \text{cyclic components} \rightarrow \text{cyclic components}$

1. $\lambda$ preserves contexts:
   $\text{contexts}(X) \subseteq \text{contexts}(\lambda(X))$
Given some (trimmed) stepwise automata for $S$ and $T$, all synopsis trees of $S$ are covered by synopsis trees of $T$.

i.e. $\exists \lambda : \text{cyclic components} \rightarrow \text{cyclic components}$

1. $\lambda$ preserves contexts:
   $\text{contexts}(X) \subseteq \text{contexts}(\lambda(X))$

2. $\lambda$ respects post-order of components:
   $X \leq_{\text{postorder}} Y \iff \lambda(X) \leq_{\text{postorder}} \lambda(Y)$

3. $\lambda$ preserves ancestorship of vertical components:
   $X/_{\text{vertical}} \text{ancestor} Y \iff \lambda(X)/_{\text{vertical}} \text{ancestor} \lambda(Y)$ whenever $\text{vertical-contexts}(X) \neq \emptyset$
Given some (trimmed) stepwise automata for $S$ and $T$, all synopsis trees of $S$ are covered by synopsis trees of $T$, i.e. $\exists \lambda$ : cyclic components $\rightarrow$ cyclic components

1. $\lambda$ preserves contexts:
   $\text{contexts}(X) \subseteq \text{contexts}(\lambda(X))$

2. $\lambda$ respects post-order of components:
   $X \leq_{\text{postorder}} Y \iff \lambda(X) \leq_{\text{postorder}} \lambda(Y)$

3. $\lambda$ preserves ancestorship of vertical components:
   $X \leq_{\text{ancestor}} Y \iff \lambda(X) \leq_{\text{ancestor}} \lambda(Y)$
   whenever $\text{vertical-contexts}(X) \neq \emptyset$
delete $d$
delete $d$
delete $d$
Complexity of **non-streaming** bounded repairability problem:

<table>
<thead>
<tr>
<th></th>
<th>det. DTD</th>
<th>DTD</th>
<th>stepwise</th>
</tr>
</thead>
<tbody>
<tr>
<td>universal</td>
<td><strong>P</strong></td>
<td><strong>PSPACE</strong></td>
<td><strong>EXP</strong></td>
</tr>
<tr>
<td>fixed alphabet</td>
<td><strong>coNP</strong></td>
<td><strong>PSPACE</strong></td>
<td><strong>PSPACE</strong></td>
</tr>
<tr>
<td>det. DTD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>non recursive</td>
<td><strong>coNEXP</strong></td>
<td><strong>coNEXP</strong></td>
<td><strong>coNEXP</strong></td>
</tr>
<tr>
<td>det. DTD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>stepwise</td>
<td><strong>coNEXP</strong></td>
<td><strong>coNEXP</strong></td>
<td><strong>coNEXP</strong></td>
</tr>
</tbody>
</table>
## Complexity of non-streaming bounded repairability problem:

<table>
<thead>
<tr>
<th></th>
<th>det. DTD</th>
<th>DTD</th>
<th>stepwise</th>
</tr>
</thead>
<tbody>
<tr>
<td>universal</td>
<td>$P$</td>
<td>PSPACE</td>
<td>EXP</td>
</tr>
<tr>
<td>fixed alphabet</td>
<td>coNP</td>
<td>PSPACE</td>
<td>PSPACE</td>
</tr>
<tr>
<td>det. DTD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>non recursive</td>
<td>coNEXP</td>
<td>coNEXP</td>
<td>coNEXP</td>
</tr>
<tr>
<td>det. DTD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>stepwise</td>
<td>coNEXP</td>
<td>coNEXP</td>
<td>coNEXP</td>
</tr>
</tbody>
</table>

## Complexity of streaming bounded repairability problem:

<table>
<thead>
<tr>
<th></th>
<th>det. DTD</th>
<th>DTD</th>
</tr>
</thead>
<tbody>
<tr>
<td>universal</td>
<td>$P$</td>
<td>PSPACE</td>
</tr>
<tr>
<td>DTD</td>
<td>EXP</td>
<td>EXP</td>
</tr>
</tbody>
</table>
Some references...

- **Regular Repair of Specifications**
  Benedikt, Riveros, P. – LICS 2011

- **The cost of traveling between languages**
  Benedikt, Riveros, P. – ICALP 2011

- **Bounded repairability for regular tree languages**
  Riveros, Staworko, P. – ICDT 2012

- **Which DTDs are streaming bounded repairable?**
  Bourhis, Riveros, Staworko, P. – ICDT 2013

...and other related topics

- normalized edit cost: $\sup_{s \in S} \min_{t \in T} \frac{\text{dist}(s, t)}{|s|}$
- distance automata and limitedness problem
- energy games with perfect/imperfect information
Some references...

- **Regular Repair of Specifications**
  Benedikt, Riveros, P. – LICS 2011

- **The cost of traveling between languages**
  Benedikt, Riveros, P. – ICALP 2011

- **Bounded repairability for regular tree languages**
  Riveros, Staworko, P. – ICDT 2012

- **Which DTDs are streaming bounded repairable?**
  Bourhis, Riveros, Staworko, P. – ICDT 2013

...and other related topics

- normalized edit cost
  \[
  \sup_{s \in S} \min_{t \in T} \frac{\text{dist}(s, t)}{|s|}
  \]

- distance automata and limitedness problem

- energy games with perfect/imperfect information