Robustness Analysis of String Transducers

Roopsha Samanta

Joint work with Jyotirmoy V. Deshmukh and Swarat Chaudhuri

February 4, 2013

Problem Overview

- String transducers abound!
  - Compilers, image processors, computational biology
- Uncertainty is pervasive
  - Noisy images in image processing engines
  - Incomplete DNA strings in computational biology
  - Wrongly spelled inputs to text processors
  - Corrupt data from sensors in medical devices

Problem Overview

- String transducers abound!
  - Compilers, image processors, computational biology
- Uncertainty is pervasive
  - Noisy images in image processing engines
  - Incomplete DNA strings in computational biology
  - Wrongly spelled inputs to text processors
  - Corrupt data from sensors in medical devices

We need predictability in the presence of uncertainty

Problem Overview

- String transducers abound!
  - Compilers, image processors, computational biology
- Uncertainty is pervasive
  - Noisy images in image processing engines
  - Incomplete DNA strings in computational biology
  - Wrongly spelled inputs to text processors
  - Corrupt data from sensors in medical devices

Small input pertubation $\rightarrow$ Small perturbation in system output

Problem Overview

- String transducers abound!
  - Compilers, image processors, computational biology
- Uncertainty is pervasive
  - Noisy images in image processing engines
  - Incomplete DNA strings in computational biology
  - Wrongly spelled inputs to text processors
  - Corrupt data from sensors in medical devices

Robustness!

Functional transducers ($\mathcal{T}$)

- Finite-state device with two tapes
- In each step, $\mathcal{T}$
  - reads an input symbol (alphabet Σ)
  - writes a finite string (alphabet Γ)
  - nondeterministically changes state
- Output of $\mathcal{T}$: defined only if run is *accepting*
  - $\mathcal{L} \subseteq \Sigma^*$
- Functional: at most one output string for every input string
  - $s' = [\![\mathcal{T}]\!](s)$
- Mealy machines: deterministic, symbol-to-symbol functional transducer

Functional transducers ($\mathcal{T}$)

- Finite-state device with two tapes
- In each step, $\mathcal{T}$
  - reads an input symbol (alphabet Σ)
  - writes a finite string (alphabet Γ)
  - nondeterministically changes state
- Output of $\mathcal{T}$: defined only if run is *accepting*
  - $\mathcal{L} \subseteq \Sigma^*$
- Functional: at most one output string for every input string
  - $s' = [\![\mathcal{T}]\!](s)$
- Mealy machines: deterministic, symbol-to-symbol functional transducer

Functional transducers ($\mathcal{T}$)

- Finite-state device with two tapes
- In each step, $\mathcal{T}$
  - reads an input symbol (alphabet Σ)
  - writes a finite string (alphabet Γ)
  - nondeterministically changes state
- Output of $\mathcal{T}$: defined only if run is *accepting*
  - $\mathcal{L} \subseteq \Sigma^*$
- Functional: at most one output string for every input string
  - $s' = [\![\mathcal{T}]\!](s)$
- Mealy machines: deterministic, symbol-to-symbol functional transducer

Functional transducers ($\mathcal{T}$)

- Finite-state device with two tapes
- In each step, $\mathcal{T}$
  - reads an input symbol (alphabet Σ)
  - writes a finite string (alphabet Γ)
  - nondeterministically changes state
- Output of $\mathcal{T}$: defined only if run is *accepting*
  - $\mathcal{L} \subseteq \Sigma^*$
- Functional: at most one output string for every input string
  - $s' = [\![\mathcal{T}]\!](s)$
- Mealy machines: deterministic, symbol-to-symbol functional transducer

Functional transducers ($\mathcal{T}$)

- Finite-state device with two tapes
- In each step, $\mathcal{T}$
  - reads an input symbol (alphabet Σ)
  - writes a finite string (alphabet Γ)
  - nondeterministically changes state
- Output of $\mathcal{T}$: defined only if run is *accepting*
  - $\mathcal{L} \subseteq \Sigma^*$
- Functional: at most one output string for every input string
  - $s' = [\![\mathcal{T}]\!](s)$
- Mealy machines: deterministic, symbol-to-symbol functional transducer

Robust transducers

Given:

- a (functional) transducer $\mathcal{T}$, over $\mathcal{L} \subseteq \Sigma^*$,
- upper bound B on input perturbation,
- distance metric $d : \Sigma^* \times \Sigma^* \cup \Gamma^\star \times \Gamma^\star \rightarrow \mathbb{N}$,
- constant $K \in \mathbb{N}$

$\mathcal{T}$ is defined to be (B, K)-robust if:

$$\forall \delta \leq B, \forall s, t \in \mathcal{L} : \; d(s,t) = \delta \implies d([\![\mathcal{T}]\!](s), [\![\mathcal{T}]\!](t)) \leq K\delta$$

Robust transducers

Given:

- a (functional) transducer $\mathcal{T}$, over $\mathcal{L} \subseteq \Sigma^*$,
- upper bound B on input perturbation,
- distance metric $d : \Sigma^* \times \Sigma^* \cup \Gamma^\star \times \Gamma^\star \rightarrow \mathbb{N}$,
- constant $K \in \mathbb{N}$

$\mathcal{T}$ is defined to be (B, K)-robust if:

$$\forall \delta \leq B, \forall s, t \in \mathcal{L} : \; d(s, t) = \delta \implies d([\![\mathcal{T}]\!](s), [\![\mathcal{T}]\!](t)) \leq K\delta$$

Problem definition

Given:

- a (functional) transducer $\mathcal{T}$, over $\mathcal{L} \subseteq \Sigma^*$,
- upper bound B on input perturbation,
- distance metric $d : \Sigma^* \times \Sigma^* \cup \Gamma^\star \times \Gamma^\star \rightarrow \mathbb{N}$,
- constant $K \in \mathbb{N}$

Check if $\mathcal{T}$ is (B, K)-robust.

Solution strategy

Given:

- a (functional) transducer $\mathcal{T}$, over $\mathcal{L} \subseteq \Sigma^*$,
- upper bound B on input perturbation,
- distance metric $d : \Sigma^* \times \Sigma^* \cup \Gamma^\star \times \Gamma^\star \rightarrow \mathbb{N}$,
- constant $K \in \mathbb{N}$

For each $\delta \leq B$, construct machine $\mathcal{A}^\delta$:

$\mathcal{T}$ is (B, K)-robust iff for all $\delta \leq B$, $\mathcal{L}(\mathcal{A}^\delta)$ is empty

Solution strategy

Construct $\mathcal{A}^\delta$ such that $\mathcal{A}^\delta$ accepts (s,t) iff:

- $d(s,t) = \delta$, and,
- $\exists (s', t')$:
  - $s' = [\![\mathcal{T}]\!](s)$
  - $t' = [\![\mathcal{T}]\!](t)$
  - $d(s', t') > K\delta$

Solution strategy

$\mathcal{A}^{\delta}$ is constructed from:

1. Input automaton, **Input**: accepts (s, t) iff $d(s, t) = \delta$
2. Pair-transducer, **Pair**: transforms (s, t) to (s', t') according to $\mathcal{T}$
3. Output automaton, **Output**: accepts (s', t') iff $d(s', t') > K\delta$.

Distance metric(s)

Levenshtein distance $d_L(s, t)$:
Minimum number of symbol *insertions*, *deletions* and *substitutions* to transform s into t

$d_L(baa, abca) = 2$

$$\begin{array}{cccc} \sqcup & b & a & a \\ a & b & c & a \end{array}$$

Distance metric(s)

Generalized Levenshtein distance $d_{gL}(s,t)$:

- Tracks the *degree*, not just the *number* of mismatches
- $\mathtt{diff}(a,b)$: pair-wise mismatch penalty to substitute a and b
- α: gap penalty to insert or delete symbol

Let: $\mathtt{diff}(a,b) = \mathtt{diff}(b,c) = 1, \;\; \mathtt{diff}(a,c) = 2, \;\; \alpha = 1$

$d_{gL}(baa, abca) = 3$

$$\begin{array}{cccc} \sqcup & b & a & a \\ a & b & c & a \end{array}$$

Distance metric(s)

$$\cancel{d_{gL}(s,t)}\ d(s,t)$$

Generalized Levenshtein distance computation

t

			c	a	c	a
		0	1	2	3	4
	0	0	1	2	3	4
a	1	1				
c	2	2				
c	3	3				
c	4	4				
a	5	5				

s

$$d(s[0], t[0]) = 0$$
$$d(s[0, i], t[0]) = i\alpha$$
$$d(s[0], t[0, j]) = j\alpha$$

$$\alpha = 1$$

Generalized Levenshtein distance computation

t

s

			c	a	c	a
		0	1	2	3	4
	0	0	1	2	3	4
a	1	1				
c	2	2		subs	del	
c	3	3		ins	?	
c	4	4				
a	5	5				

$$
\begin{aligned}
& d(s[0,i],\ t[0,j]) = \\
& \min(d\,(s[0,i{-}1],\ t[0,j{-}1]) + \mathtt{diff}(s[i],\ t[j]), \\
& \qquad d\,(s[0,i{-}1],\ t[0,j]) + \alpha, \\
& \qquad d\,(s[0,i],\ t[0,j{-}1]) + \alpha \\
& \quad)
\end{aligned}
$$

$\alpha = 1$
$\mathtt{diff}(a,b) = \mathtt{diff}(b,c) = 1$,
$\mathtt{diff}(a,c) = 2$

Generalized Levenshtein distance computation

			c	a	c	a
		0	1	2	3	4
	0	0	1	2	3	4
a	1	1	2	1	2	3
c	2	2	1			
c	3	3	2			
c	4	4	3			
a	5	5	4			

$$d(s[0,i],\ t[0,j]) =$$
$$\min(d\,(s[0,i{-}1],\ t[0,j{-}1]) + \mathtt{diff}(s[i],\ t[j]),$$
$$d\,(s[0,i{-}1],\ t[0,j]) + \alpha,$$
$$d\,(s[0,i],\ t[0,j{-}1]) + \alpha$$
$$)$$

$\alpha = 1$
$\mathtt{diff}(a,b) = \mathtt{diff}(b,c) = 1,$
$\mathtt{diff}(a,c) = 2$

Generalized Levenshtein distance computation

t

s

			c	*a*	*c*	*a*
		0	1	2	3	4
	0	0	1	2	3	4
a	1	1	2	1	2	3
c	2	2	1	2	1	2
c	3	3	2	3	2	3
c	4	4	3	4	3	4
a	5	5	4	3	4	3

$$d(s[0,i],\ t[0,j]) =$$
$$\min(d\,(s[0,i{-}1],\ t[0,j{-}1]) + \mathtt{diff}(s[i],\ t[j]),$$
$$d\,(s[0,i{-}1],\ t[0,j]) + \alpha,$$
$$d\,(s[0,i],\ t[0,j{-}1]) + \alpha$$
$$)$$

$\alpha = 1$
$\mathtt{diff}(a,b) = \mathtt{diff}(b,c) = 1$,
$\mathtt{diff}(a,c) = 2$

Generalized Levenshtein distance computation

t

s

			c	a	c	a
		0	1	2	3	4
	0	0	1	2		
a	1	1	2	1	2	
c	2	2	1	2	1	2
c	3		2	3	2	3
c	4			4	3	4
a	5				4	3

$\delta = 2$

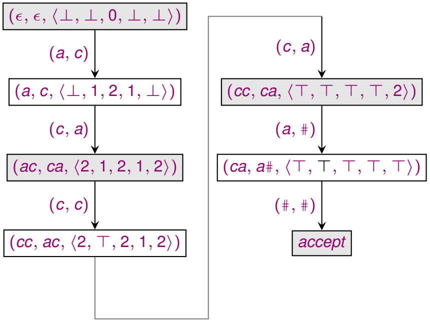
To check if $d(s,t) > \delta$ or if $d(s,t) = \delta$, focus on δ-diagonal.

Construct DFA $\mathcal{D}^{>\delta}$: runs on a string pair (s,t), and accepts iff $d(s,t) > \delta$.

Construct DFA $\mathcal{D}^{=\delta}$: runs on a string pair (s,t), and accepts iff $d(s,t) = \delta$.

Distance-tracking automaton, $\mathcal{D}^{>\delta}$

			c	*a*	*c*	*a*	#	#
		0	1	2	3	4	5	6
	0	0	1	2				
a	1	1	2	1	2			
c	2	2	1	2	1	2		
c	3		2	$\top$	2	$\top$	$\top$	
c	4			$\top$	$\top$	$\top$	$\top$	$\top$
a	5				$\top$	$\top$	$\top$	$\top$
#	6					$\top$	$\top$	$\top$



$$\delta = 2$$

Distance-tracking automata

$\mathcal{D}^{>\delta}$ accepts a pair of strings (s, t) iff $d(s, t) > \delta$.

$\mathcal{D}^{=\delta}$ accepts a pair of strings (s, t) iff $d(s, t) = \delta$.

Robustness analysis of Mealy machines

$\mathcal{A}^{\delta}$ is constructed from:

1. **Input**: accepts (s, t) iff $d(s, t) = \delta$
2. **Pair**: transforms (s, t) to (s', t') according to $\mathcal{T}$
3. **Output**: accepts (s', t') iff $d(s', t') > K\delta$.

Robustness analysis of Mealy machines

$\mathcal{A}^{\delta}$ is constructed as a *synchronized product* of:

1. **Input**: $\mathcal{D}^{=\delta}$
2. **Pair**: transforms (s, t) to (s', t') according to $\mathcal{T}$
3. **Output**: $\mathcal{D}^{>K\delta}$

Robustness analysis of functional transducers

- In each transition, **Pair** can generate a string pair
- In each transition, **Output** can only read a symbol pair

- **Output** is tricky — needs to remember substrings of leading string
- $\mathcal{A}^{\delta}$ is not a simple synchronized product

Robustness analysis of functional transducers

- In each transition, **Pair** can generate a string pair
- In each transition, **Output** can only read a symbol pair

- **Output** is tricky — needs to remember substrings of leading string
- $\mathcal{A}^\delta$ is not a simple synchronized product

Let's have a look at "**MeFirst**"

- **MeFirst** $= \texttt{Trim}(\textbf{Input} \otimes \textbf{Pair})$
- Pairwise-delay (pd) of path π of **MeFirst**: $\mathrm{abs}\,(|w'| - |v'|)$

Let's have a look at "**MeFirst**"

- **MeFirst** $= \mathtt{Trim}(\textbf{Input} \otimes \textbf{Pair})$
- Pairwise-delay (pd) of path π of **MeFirst**: $\mathrm{abs}\left(|w'| - |v'|\right)$

MeFirst has bounded pd iff pd of all cyclic paths in **MeFirst** is 0.

MeFirst has bounded pd iff all simple cycles have equal length output strings.

Let's have a look at "**MeFirst**"

- **MeFirst** $= \texttt{Trim}(\textbf{Input} \otimes \textbf{Pair})$
- Pairwise-delay (pd) of path π of **MeFirst**: $\text{abs}\,(|w'| - |v'|)$

If **MeFirst** has bounded pd, then maximum pd over all paths $<$ **Delay**, where **Delay** $= |Q|^2|Q_{\mathcal{I}}|\ell_{max}$.

- $Q, Q_{\mathcal{I}}$: states of $\mathcal{T}$, **Input**
- ℓ_{max}: length of longest output string in $\mathcal{T}$'s transitions

Let's have a look at "**MeFirst**"

- **MeFirst** $= \texttt{Trim}(\textbf{Input} \otimes \textbf{Pair})$
- Pairwise-delay (pd) of path π of **MeFirst**: $\mathsf{abs}\,(|w'| - |v'|)$

If **MeFirst** does not have bounded pd, $\mathcal{T}$ is non-robust.

Robustness analysis of functional transducers

1. Check if **MeFirst** has bounded pd
2. If not, declare $\mathcal{T}$ as non-robust
3. If yes, *carefully construct* $\mathcal{A}^{\delta}$ from:
   1. **Input**: $\mathcal{D}^{=\delta}$
   2. **Pair**: transforms (s, t) to (s', t') according to $\mathcal{T}$
   3. **Output**:
      - similar to $\mathcal{D}^{>K\delta}$
      - remembers **Delay** $+ K\delta$ symbols in state

   and proceed as before

Results

Robustness verification w.r.t. generalized Levenshtein distance:

- Mealy machine — can be done in PSPACE in B and K
- Functional transducer — can be done in EXPSPACE in B

Robustness verification w.r.t. generalized Manhattan distance:

- Mealy machine — can be done in NLOGSPACE in $size(\mathcal{T})$, B, K, $|\Sigma|$, $|\Gamma|$ and maximum mismatch penalty
- Functional transducer — can be done in PSPACE in B and K

Results

Robustness verification w.r.t. generalized Levenshtein distance:

- Mealy machine — can be done in PSPACE in B and K
- Functional transducer — can be done in EXPSPACE in B

Robustness verification w.r.t. generalized Manhattan distance:

- Mealy machine — can be done in NLOGSPACE in $size(\mathcal{T})$, B, K, $|\Sigma|$, $|\Gamma|$ and maximum mismatch penalty
- Functional transducer — can be done in PSPACE in B and K

Related Work

- Sequential programs with perturbed inputs [MS09, CGLN11]
- Input-output stability of finite-state transducers [TBCSM12]
- Sequential circuits, *common suffix distance metric* [DHLN10]
- Robustness analysis of networked systems [SDC13]
- Reactive systems with ω-regular spec. in uncertain environment [MRT11, CHR10, BGHJ09]

Future Work

- Understand robustness of transducers better
- Generalize error model - channel error, modeling error, process failure
- Generalize system model - weighted transducers?

Thank you.