Streaming Property Testing of Visibly Pushdown languages

M. de Rougemont

University Paris II & LIAFA-CNRS

Joint work with N. François, F. Magniez, O. Serre
1. **Streaming and Property Testing**

   Streaming Property Testing of VPL languages (XML valid words)

   Main result: Tester with $O\left(\frac{(\log n)^p}{\varepsilon^q}\right)$ memory

2. **Tools**

   3.1 Tester for weighted words (Edit distance)
   
   3.2 Relationship between VPA and its slicing automaton

3. **New techniques**

   4.1 Online Dichotomic sampling
   
   4.2 Online Compression
1. Streaming Property Testing

1. Streaming and Property Testing

Streaming: small memory to decide a property
Property testing: small number of samples to approximately decide a property

2. Visibly Pushdown Languages (include XML valid words)

Streaming, \( \Omega(n) \) memory
Property Testing, \( \Omega(n^\alpha) \) samples


Definition: A Streaming \( \varepsilon \)-Tester with memory \( s(n) \) for \( L \) is a randomized Algorithm \( A \) such that:
- \( w \) in \( L \) then \( A \) accepts with probability 1
- \( w \) \( \varepsilon \)-far from \( L \) then \( A \) rejects with probability \( 2/3 \)
- \( A \) uses \( s(n) \) memory in one pass

Result: Streaming Tester for VPL with \( O\left(\frac{(\log n)^p}{\varepsilon^q}\right) \) memory
Tool: Tester for regular properties of weighted words

Weighted words, weighted edit distance

\[ W = \quad a \ a \ a \ b \ b \ a \ a \ a \ b \ b \ b \ b \ a \ b \ b \ b \ b \ b \ b \ b \ b \ b \ b \ \quad 3/n \ close \ to \ a^*b^* \]

\[
\begin{align*}
1 & \quad 1 & \quad 1 & \quad 5 & \quad 5 & \quad 1 & \quad 1 & \quad 1 & \quad 5 & \quad 5 & \quad 1 & \quad 1 & \quad 5 & \quad 5 & \quad 1 & \quad 1 & \quad 1 \\
\end{align*}
\]

weights

\[ W \quad \text{is} \quad 8/n \ \text{close to} \ a^*b^* \ \text{for the} \ \text{weighted edit distance} \]

Tester generalizes Alon’s tester for regular languages (for the Hamming distance)
VPA and General approach

**Ordered unranked tree** represented as a balanced word

```
a b b b a a a b a
```

VPA may accept `a b b a`

**Tree automaton vs. VPA**

1. Non alternating sequence: `a b a a b a`
   
   \[ w \in L(A) \text{ iff } \widehat{w} \in L(\widehat{A}) \text{ and } \widehat{A} \text{ is a finite automaton (slicing)} \]

2. General sequence:
   
   Construct \( \widehat{w} \) by compression of the peaks
2. Tool: Tester for regular properties of weighted words

Weighted words, weighted edit distance

Automaton: Connected components $C_1 C_2 \ldots C_k$

Accepting Paths: $C_1 . C_2 , \ C_1 . C_3 . C_4 , \ldots$

Tester: $m$ samples (factors of length $k$) with the weight distribution

Are the samples compatible for an accepting path?

Example: Tester for $a^*b^*$

2 samples of length 1: Accepts $a.a$, $b.b$, $a.b$

Rejects $b.a$
Tester for regular properties of weighted words

**Key lemma:** If \( w \) is \( \varepsilon \)-far for \( C_1 \cdot C_2 \), Tester rejects with h.p.

**Saturation analysis** on \( a^*b^* \): \( \varepsilon \cdot n/2 \) cuts for \( a^* \) then \( \varepsilon \cdot n/2 \) cuts for \( b^* \)

\[
\varepsilon \cdot n/2 = 3 \quad a \quad a \quad a \quad b \quad b \quad a \quad a \quad a \quad b \quad b \quad b \quad a \quad b \quad b \quad b \quad b
\]

If \( w \) is \( \varepsilon \)-far, it saturates \( C_1 \) and \( C_2 \): \( \text{Prob}[" b-a"] \) large

Tester Accepts/ Rejects

Modified Tester returns \( R \) subset of \( Q \cdot Q \)
Tester as an approximate compression

Let $R_w$ be the set of $(p,q)$ such that $p \rightarrow^w q$

$w = a \ a \ a \ b \ b \ a \ a \ a \ b \ b \ b \ b \ a \ b \ b \ b \ b$

$R_w = \{(1,3),(2,3),(3,3)\}$

Definition: $R \subseteq Q.Q$ $\varepsilon$-approximates a word $w$ for an automaton $A$ if:

1. $p \rightarrow^w q$ implies $(p,q) \in R$
2. $(p,q) \in R$ implies $\exists v$, $\varepsilon$-close to $w$, $p \rightarrow^v q$

Tester returns $R$ which $\varepsilon$-approximates $w$ on $A$.

$R'_w = \{(1,3),(2,3),(3,3),(1,2)\}$
**VPL: Alternating sequences**

**Balanced edit distance** (insertion/deletion of \(a, \_\))

\[
\begin{align*}
\text{b b b a b b a a a a a a b b b a b b b} & \quad \text{balanced word} \\
W=b-----b b a b b a---- a a a a a b b a ---- -,b b b \quad \text{word with neutral -}
\end{align*}
\]
VPL: Non-Alternating sequences

General situation: compression of the peaks
3. Exact Tester

Principle of an exact tester:
*Detect peaks and replace them by R*

1. New peak: push on the stack

2. Balanced peak: compress and merge

3. Large peak: \((|u| > |\text{previous balanced peak}|/2)\) compress the top of the stack and merge

4. Output R
1. New Peak pushed on the stack
2. Balanced Peak Compression and Pop-Merge
Exact Tester: case 3

3. $|u_0| > |v_2|/2$  compression of the top of the stack and Merge
Example

First cut  \( u_0 = v_1 \cdot v_2 \)

\[ |u_0| > |v_2| / 2 \]
Example

New partial peak:

Balanced peak compressed
New partial peak: Balanced peak compressed

Example
1. **Height of the Stack** = \( \log n \)

Worst-case: sequence of small peaks, size decreases geometrically

2. **Nesting of the R’s**

\[ \text{Depth}(R_w) = \text{Depth}(w) + 1 \]

Size \((R_w)\) decreases geometrically with the nesting

**maximum nesting** = \( O(\log n) \)
Samplings on words

1. Reservoir sampling: t elements maintained with the uniform distribution

2. Weighted Reservoir sampling: t elements with the distribution of the weights

3. Suffix sampling: $\alpha > 1$ : $\alpha$-decomposition $|u_{i+1}| < |u_i|/\alpha$

4. $(\alpha, t)$ sampling: t samples on each $u_i$ of an $\alpha$-Suffix-sampling
Largest suffix $u_i$:
t positions and weight k on the push symbols, matching positions on the pop

Call the Tester on weighted words

Same procedure for each $u$ of the decomposition
Approximate Tester

Principle of an approximate tester:
*Detect peaks (use heights) and replace them by approximate R*

1. New peak: push the samples

2. Balanced peak: compress by an approximate R (merge the samples)

3. Large peak: compress the top of the stack by an approximate R (merge the samples)

4. Output an approximate R
**Analysis of the Approximate Tester**

1. **Given**: a balanced word with all possible R as neutral letters, T samples, ε
Output: an approximate R

   **Definition**: $R \subseteq Q.Q$ ε-approximates a word w for an automaton A if:
   1. $p \xrightarrow{w} q$ implies $(p, q) \in R$
   2. $(p, q) \in R$ implies $\exists v$, ε-close to w, $p \xrightarrow{v} q$

2. **Robustness**: $\text{bdist}(w_i, v_i) > 3. \varepsilon.n$ then Tester Rejects w.h.p.

   Approximate Tester outputs $R_{\text{final}}$ for $\alpha=1+\varepsilon/\log n$

   Construct $w, w_1, w_2, \ldots, w_h = R_{\text{final}}$ for the different nestings, $h=O(\log n)$

   $v, v_2, v_1, \ldots, v_h$ are in L

   By induction on l, $\text{bdist}(w_i, v_i) < 3.h. |w_i| \cdot \varepsilon/\log n$ for some $v_i$ in L.

   **Conclusion**: If $R_{\text{final}}$ in L, then , $\text{bdist}(w_i, v_i) < 3. \varepsilon.n$ for some $v_i$ in L.
Main Result

Theorem: Let A be a VPA for L. There is an $\varepsilon$-streaming tester for L
and Memory $O(c. \frac{(\log n)^6}{\varepsilon^4})$.

Analysis:

$$ t = O(c_1. \frac{(\log n)^2}{\varepsilon^2}) \text{ samples of weight } k=O(c_2. \frac{(\log n)}{\varepsilon}) $$

For each $u$, $O(c_3. \frac{(\log n)^2}{\varepsilon})$ decomposition, Stack $O(\log n)$.

Hence Memory $O(c. \frac{(\log n)^6}{\varepsilon^4})$. 
Example: Testing a regular tree

Testing \( w \in a^*b^* \) requires two samples.

Need \( O(n^4) \) samples to test \( T \in c^*[b^*a^*]^* \): birthday's paradox.
Streaming Property Testing

Compression step

Diagram showing the streaming property testing process with labels a*, b*, and c*.
Streaming Property Testing

Iterated Compression: final slicing word
Conclusion

1. Streaming versus Property Tester

2. Streaming Property Tester for VPL $O(\text{poly} \ (\log n))$

3. New techniques
   - Dichotomic sampling
   - Online approximative compression