Projection for Nested Word Automata
Speeds up XPath Evaluation on XML Streams

Tom Sebastian
INNOVIMAX & LINKS TEAM

Joachim Niehren
INRIA & LINKS TEAM

SOFSEM 2016
Outline

1. Introduction
2. XML Stream Processing via Nested Word Automata
   - Nested Word Automata
   - Evaluation of Nested Word Automata
3. Projection Nested Word Automata
   - Projected Nested Words
   - Projection Nested Word Automata
4. Translation from NWAs to pNWAs
5. Evaluation of Projection NWAs
6. Experiments
Projection for XML processing

Many existing works:

- Marian&Siméon 03: Projecting XML documents. In VLDB.
- Frisch 04: Regular tree language recognition with static information. In IFIP TCS.
  - projection of XML documents,
  - according to XQuery programs (and thus XPath queries),
  - implements Siméon’s algorithms
Projection for XPath queries

XPath query: `[/a/b]`

Consider XML tree:

```
  c
  |  
  a  c
  |   |  
  c   c
  |    |   
  b    c
  |     |   
  a     b
```
Projection for automata

- needed for efficient evaluation of automata over some input structure
- jumping inside of the input structure
- consumption of only part of the input structure
- input structure in-memory: indexes
- input structure streaming: parsing and running automata only for relevant parts
- many queries, same input structure
Projection for finite word automata

Example

consider word $w = cbacaabcc$

consider language of all words that contain $ab$:

![Automaton Diagram]

projected word automaton:

![Projected Automaton Diagram]

projected word: $\ldots a\ldots aab\ldots$

- Runtime of automaton:
  - depends on #a’s in $w$, not on the length of $w$
Processing of XML Streams = Nested Word

Nested Word Automata

<table>
<thead>
<tr>
<th>Nested Word Automata</th>
<th>Alur&amp;Madhudusan 05</th>
</tr>
</thead>
<tbody>
<tr>
<td>visibly pushdown automata</td>
<td>Alur&amp;Madhudusan 02</td>
</tr>
<tr>
<td>pushdown forest automata</td>
<td>Neumann&amp;Seidl 98</td>
</tr>
</tbody>
</table>

XML Streaming with Nested Word Automata

- Madhusudan&Viswanathan 09. Query automata for nested words.
Projection for Nested Words

consider XML tree:

```
  c
   /\    \
  a   c   a
   \   /   / \b
  c   c /   /a  \
  b    /    /c
```

nested word:

\[
\langle c \rangle \langle c \rangle \langle a \rangle \langle c \rangle \langle b \rangle \langle /b \rangle \langle /c \rangle \langle a \rangle \langle a \rangle \langle /a \rangle \langle /c \rangle
\]

projected nested word for XPath query [//a/b]:

```
\langle a \rangle \langle /a \rangle
\langle a \rangle \langle b \rangle
```

\[
\text{ch}^2 \quad \text{ch}^{-2}/\text{ch} \quad \langle a \rangle \langle /a \rangle \quad \langle b \rangle
\]
Contributions

Projected Nested Words

“Projection” Nested Word Automaton
- computes the projected nested word
- runs on the projected nested word

Translation to Projection Nested Word Automaton
- defines Projection Properties in the Nested Word Automaton
- Irrelevant Labels and Prefixes
Implementation

**XPath query answering on XML Streams**
- Improvement by a factor of 4 in parsing-free time

**Streaming vs In-Memory**
- 12 queries in the same time as Saxon in-memory
Outline

1 Introduction

2 XML Stream Processing via Nested Word Automata
   - Nested Word Automata
   - Evaluation of Nested Word Automata

3 Projection Nested Word Automata
   - Projected Nested Words
   - Projection Nested Word Automata

4 Translation from NWAs to pNWAs

5 Evaluation of Projection NWAs

6 Experiments
Automata for Nested Words (NWAs)

Automata with visible stacks running on XML streams

A Nested Word Automaton (NWA) $A = (\Sigma, Q, I, F, \Gamma, R)$

Open: $q \xrightarrow{\langle a \rangle \downarrow \gamma} q'$. For open tag $\langle a \rangle$ push $\gamma$, change from $q$ to $q'$.

Close: $q \xrightarrow{\langle /a \rangle \uparrow \gamma} q'$. For close tag $\langle /a \rangle$ pop $\gamma$, change from $q$ to $q'$. 
NWA for [//a/b]

- alphabet $\Sigma = \{a, b, c\}$
- stack symbols $\Gamma = \{\alpha, \beta, \beta', \gamma, \delta\}$
- initial state $q_1$
- final state $q_4$
Run of a NWA

\[
\begin{align*}
&\langle \{b, c\} \rangle \downarrow \beta' \\
&\langle a \rangle \downarrow \beta \\
&\langle \Sigma \rangle \downarrow \alpha
\end{align*}
\]

\[
\begin{align*}
&\langle /\{b, c\} \rangle \uparrow \beta' \\
&\langle /c \rangle \uparrow \gamma \\
&\langle /a \rangle \uparrow \beta \\
&\langle \Sigma \rangle \uparrow \delta
\end{align*}
\]

\[
\begin{align*}
&\langle b \rangle \downarrow \alpha \\
&\langle \Sigma \rangle \downarrow \delta
\end{align*}
\]
Outline

1. Introduction

2. XML Stream Processing via Nested Word Automata
   - Nested Word Automata
   - Evaluation of Nested Word Automata

3. Projection Nested Word Automata
   - Projected Nested Words
   - Projection Nested Word Automata

4. Translation from NWAs to pNWAs

5. Evaluation of Projection NWAs

6. Experiments
A projected nested word is a word whose letters are

- jump symbols \( i \) where \( i \in \mathbb{Z} \),
- jump targets \( \langle a \rangle @r \) and \( \langle / a \rangle @r \), where \( a \in \Sigma \), \( r \in \{ \text{self, ch, ch}^+, \text{ch}^{-n}/\text{ch}^+, \text{ch}^{-n}/\text{ch} \} \), \( n \in \mathbb{N} \), or
- usual tags \( \langle a \rangle \) and \( \langle / a \rangle \), where \( a \in \Sigma \).

Jumps are proceeded by jump symbols: indicates depth difference

\[
\ldots \langle a \rangle @\text{ch}^+ \ldots \langle / a \rangle @\text{self} \ldots \langle a \rangle @\text{ch}^{-2}/\text{ch}^+ \ldots \langle b \rangle @\text{ch}
\]
Projection NWAs

Definition

A projection nested word automaton (PNWA) is a tuple $A = (\Sigma, Q, I, F, \Gamma, R)$ like for NWAs but different kinds of rules.

Open: $q \xrightarrow{\langle a \rangle \downarrow \gamma} q'$. Like for NWAs.

Close: $q \xrightarrow{\langle /a \rangle \uparrow \gamma} q'$. Like for NWAs.
Projection NWAs: Jumping to a child

Jump to a child: $q \xrightarrow{.\langle a\rangle@ch\downarrow0\downarrow\gamma} q'$, short $q \xrightarrow{ju(\langle a\rangle, ch, \gamma)} q'$

Example: [c/a]
Projection NWAs: Jump to a descendant

Jump to a descendant: $q \xrightarrow{z \langle a \rangle @ ch^+ \downarrow \downarrow \gamma} q'$, short $q \xrightarrow{ju(\langle a \rangle, ch^+, \gamma)} q'$

Example: [d//a/b]
Projection NWAs: Jump to Closing tag

Jump to the closing tag of current node: $q \xrightarrow{\mathit{\text{0}.\langle /a\rangle @\text{self} \uparrow \gamma}} q'$.

Example: [c/a/d]
Projection NWAs: Rejump to a child or a descendant

Rejump to another child or descendant $r \in \{ch, ch^+\}$:

$$q \xrightarrow{z.\langle a\rangle @ ch^{-}(z'+1) / r \uparrow z' \downarrow z + z' \downarrow \gamma} q', \text{ short } q \xrightarrow{reju(\langle a\rangle , r, \gamma)} q'$$

Example: [d//a/b]
Projection NWAs: Jump to the Jump’s origin

Jump back to the jump’s origin:

\[ q \xrightarrow{\frac{-z\langle/a\rangle@ch^{-}(z+1)\uparrow\uparrow\gamma}{\forall z \geq 0}} q', \text{ short } q \xrightarrow{ju\text{-}back(\langle/a\rangle, \gamma)} q' \]

Example: [d//a/d]
Projection NWAs: Close last jump step

Close last jump step: $q \xrightarrow{\langle/a\rangle \uparrow z \downarrow z-1, \forall z>0} q'$.
Projection NWA: [//a/b]
Outline

1. Introduction

2. XML Stream Processing via Nested Word Automata
   - Nested Word Automata
   - Evaluation of Nested Word Automata

3. Projection Nested Word Automata
   - Projected Nested Words
   - Projection Nested Word Automata

4. Translation from NWAs to pNWAs

5. Evaluation of Projection NWAs

6. Experiments
Irrelevant Label and Prefixes

\[
\begin{align*}
\langle L \rangle & \downarrow \gamma \\
\langle \!\!\! \langle L \rangle \!\!\! \rangle & \uparrow \gamma
\end{align*}
\]

\(q\) jump to a descendant

\[
\begin{align*}
\langle L \rangle & \downarrow \gamma \\
\langle \!\!\! \langle L \rangle \!\!\! \rangle & \uparrow \gamma
\end{align*}
\]

\(q\) jump to a child

\[
\begin{align*}
\langle L \rangle & \downarrow \gamma \\
\langle \!\!\! \langle L \rangle \!\!\! \rangle & \uparrow \gamma
\end{align*}
\]

\(q\) jump to a descendant

\[
\begin{align*}
\langle L \rangle & \downarrow \gamma \\
\langle \!\!\! \langle L \rangle \!\!\! \rangle & \uparrow \gamma
\end{align*}
\]

\(q\) jump to a child

\[
\begin{align*}
\langle L \rangle & \downarrow \gamma \\
\langle \!\!\! \langle L \rangle \!\!\! \rangle & \uparrow \gamma
\end{align*}
\]
Jumping and Rejumping to Descendants

\[
\begin{align*}
&\langle \{b, c\} \rangle \downarrow \beta' \\
&\langle \Sigma \rangle \downarrow \alpha \\
&\langle /\{b, c\} \rangle \uparrow \beta' \\
&\langle /a \rangle \uparrow \gamma \\
&\langle a \rangle \downarrow \alpha \\
&\langle /a \rangle \uparrow \alpha \\
&\langle /c \rangle \downarrow \gamma \\
&\langle b \rangle \downarrow \alpha \\
&\langle \Sigma \rangle \downarrow \alpha \\
&\langle /\Sigma \rangle \uparrow \delta \\
&\langle /\Sigma \rangle \uparrow \alpha \\
&\langle /\Sigma \rangle \uparrow \beta \\
&\langle /\Sigma \rangle \uparrow \beta' \\
\end{align*}
\]
Outline

1. Introduction
2. XML Stream Processing via Nested Word Automata
   - Nested Word Automata
   - Evaluation of Nested Word Automata
3. Projection Nested Word Automata
   - Projected Nested Words
   - Projection Nested Word Automata
4. Translation from NWAs to pNWAs
5. Evaluation of Projection NWAs
6. Experiments
Evaluation of pNWAs

- Determinization of the NWA
- Computation of Projection Properties
- Creation of the Projection NWA on-the-fly
- Computes next relevant events for all states of candidates
Experiments: Setting

We implemented the presented algorithms in our QuiXPath system.

We tested against the revised version of the XPathMark query set.

We measure Parsing-free Times: The evaluation time for a query excluding the parsing time.
Experiments: pNWA vs NWAs

Gain

factor 4.3 in Parsing-free time
Experiments

Streaming vs In-Memory

Streaming or In-Memory?

12 queries in Streaming in the same time as Saxon in-memory
Thank you!
Node Selection

- Debarbieux & Gauwin & Niehren & Sebastian & Zergaoui. 15. Early Nested Word Automata for XPath Query Answering on XML Streams

- query: //a[b]
- annotated alphabet $\Sigma_{anno} = \{ a, a^x \mid a \in \Sigma \}$
- NWAs test for acceptance of trees with an by $x$ annotated node
- candidate generation
- buffer management