In this talk, we are interested on streaming XML documents.

```xml
<doc>
<elem>
<elem>
<child>
<child>
<child>
</child>
</child>
</elem>
</elem>
</doc>
```
Two main questions

- XML Validation with respect to a DTD:
  \[
  \begin{align*}
  r & \rightarrow a^+ \\
  a & \rightarrow a^+ | b^+ | \epsilon \\
  b & \rightarrow \epsilon 
  \end{align*}
  \]

  How much memory do we require to validate a streaming XML Document with respect to a DTD?

- XML Filtering for XPath queries:
  \[
  /\text{descendant::a[child::b]}/\text{child::c}
  \]

  How much memory do we require to evaluate an XPath query over a streaming XML Document?
First problem: XML validation

Example

\[ \begin{align*}
  & r \rightarrow a^* \\
  & \text{d}_1 : a \rightarrow b^* \\
  & \quad b \rightarrow \epsilon \\
  \end{align*} \]

\[ L(d_1) = r (a (b \bar{b})^* \bar{a})^* \bar{r} \checkmark \]

\[ \begin{align*}
  & r \rightarrow a \\
  & d_2 : a \rightarrow a | \epsilon \\
  \end{align*} \]

\[ L(d_2) = \{ r (a^n \bar{a}^n) \bar{r} \mid n \in \mathbb{N} \} \times \]
XML validation main results

Theorem [SV02]
A streaming XML Document can be validated with constant memory with respect to a DTD iff the DTD is non-recursive.

Theorem [SV02], [GKS07]
The memory required to validate a streaming XML Document $t$ with respect to a DTD is in
$$\Theta(\text{Depth}(t))$$
Second problem: XML filtering

Let $t$ be a streaming XML document and $Q$ an XPath query.

- **One scan:**
  
  $$t: \ r\ a\ b\ b\ a\ a\ a\ a\ \ldots$$
  
  (1-time) $\uparrow$

- **Multiple scans:**
  
  $$t: \ r\ a\ b\ b\ a\ a\ a\ a\ \ldots$$
  
  (k-times) $\uparrow$

- **Indexed streams:**
  
  **Indexed node**: (Begin, End, Level)

  
  $$a: (2,5,2) (6,9,2) (7,8,3) \ldots$$
  
  (1-time) $\uparrow$
XML filtering main results

Let \( t \) be a streaming XML Documents and \( Q \) a Core XPath query.

**Theorem**

- **One scan** [GKS07]:
  
  The memory required to evaluate \( Q \) over \( t \) is in \( \Theta(\text{Depth}(t)) \).

- **Multiple scans** [GKS07]:
  
  The memory required \( m \) to evaluate \( Q \) over \( t \) with \( s \) scans satisfy:

  \[
  s \cdot m \in \Omega(\text{Depth}(t))
  \]

- **Indexed streams** [SBY08]:
  
  The memory required to evaluate \( Q \) over indexed XML streams of \( t \) is in \( \Theta(\text{Depth}(t)) \).
Stream-based processing of XML documents

Cristian Riveros
M. Benedikt

Oxford University
Thurs 12 Nov 2010
Outline

Notation

XML validation

XML filtering
Some notation

- Two fixed alphabets: $\Sigma$ and $\bar{\Sigma}$.

- Tags alphabet: $\Delta = \Sigma \cup \bar{\Sigma}$.

- We consider the set of well formed XML documents:

$$\text{Docs} = \{ t \in \Delta^* | t \text{ is a well-formed XML document} \}$$

- We use the following notation:
  - $t = \text{XML document}$.
  - $d = \text{DTD}$.
  - $Q = \text{an XPath query}$.
Outline

Notation

XML validation

XML filtering
Validation with respect to a DTD (Document Type Definition)

Definition

A DTD \( d = (r, R) \) over \( \Delta^* \) is a tuple where:

- \( r \in \Sigma \) is the root label, and
- \( R = \{ a \rightarrow R_a \mid a \in \Sigma \} \) with \( R_a \) a regular expression over \( \Sigma \).

We define \( \mathcal{L}(d) \) the set of all XML documents that satisfies \( d \):

\[
\mathcal{L}(d) = \{ t \in \text{Docs} \mid t \models d \}
\]

Example

\[
\begin{align*}
    r & \rightarrow a^* \\
    a & \rightarrow b^* \\
    b & \rightarrow \epsilon
\end{align*}
\]
Two possible flavors of XML Validation

- **Well-formed** \( \Rightarrow t \in \text{Docs} \)

**Example**

- \( r \ a \ b \ b \ a \ a \ r \) \( \rightarrow \) well-formed
- \( r \ a \ b \ b \ a \ a \ r \) \( \rightarrow \) not well-formed

- **Valid with respect to a DTD** \( d \) \( \Rightarrow t \in \mathcal{L}(d) \)

**Definition**

- **strong-validation** = well-formed + valid
- **weak-validation** = valid
A restrictive subset of DTDs: non-recursive DTDs

Let $d = (r, R)$ be a DTD over $\Sigma$.

**Definition**

We define the implication graph $G_d = (V, E)$ of $d$ where:

- $V = \Sigma$ is the set of nodes, and
- $(a, b) \in E$ if $b$ occurs in $R_a$ for $a \rightarrow R_a$ a rule in $R$.

**Example**

$d :$

\[
\begin{align*}
    r & \rightarrow a^* \\
    a & \rightarrow a | \epsilon
\end{align*}
\]

$G_d :$

```
  r  ---->  a
   \\
```

$d$ is non-recursive iff $G_d$ is acyclic.
Non-recursive DTDs characterize strong-validation

Theorem [SV02]

A streaming XML Document can be strongly validated with constant memory with respect to a DTD iff the DTD is non-recursive.

Proof idea.

(⇒) By pumping argument.

(⇐)

- For each $b \rightarrow R_b$ construct the automaton $A_b$ such that:

$$\mathcal{L}(A_b) = \mathcal{L}(b' \cdot R_b \cdot \bar{b}')$$

- Construct $A_0 = A_r, \ldots, A_i$, inductively.

Since $d$ is non recursive, this process is sure to terminate.
Weak-validation

Definition

d can be weakly validated with constant memory if there exists some regular language R such that:

\[ \mathcal{L}(d) = \text{Docs} \cap \mathcal{L}(R) \]

Example

d : 
\[
\begin{align*}
    r & \rightarrow a^* \\
    a & \rightarrow a | \epsilon
\end{align*}
\]

\[ \mathcal{L}(d) = \text{Docs} \cap \mathcal{L}(r a^* \bar{a}^* \bar{r}) \]
Not all XML documents can be weakly validated with constant memory

Example

\[ r \rightarrow a \cdot b \cdot a \]

\[ d_2 : \]

\[ a \rightarrow a | \epsilon \]

\[ b \rightarrow \epsilon \]

\[ \mathcal{L}(d_2) = \{ r \, (a^n \, \bar{a}^n) \, b \, \bar{b} \, (a^m \, \bar{a}^m) \, \bar{r} \mid n, m \in \mathbb{N} \} \]

\[ d_2 \text{ cannot be weakly validated with constant memory.} \]
Weak-validation with constant memory is an open problem

- A characterization for *fully recursive DTDs* was proved in [SV02].

  
  fully recursive DTD $\subsetneq$ DTD

- Progress has been made in [SS07].

A general characterization for weak-validation with constant memory is still open.
Formal memory model

Let $s : \Delta^* \to \mathbb{N}$ (scan) and $m : \Delta^* \to \mathbb{N}$ (memory).

Definition

A language $L \subseteq \Delta^*$ is in the class $ST(s, m)$, or $L \in ST(s, m)$, if there exists a streaming algorithm that decides $L$ such that for every $w \in \Delta^*$:

- the number of scans is less than $s(w)$, and
- the memory used is in $O(m(w))$.

Example

For a non-recursive DTD $d$:

$L(d) \in ST(1, 1)$
The memory required to validate a DTD is in $\Theta(\text{Depth}(t))$

Let Depth($t$) be the document depth of $t$.

Theorem [SV02, GKS07]

- For every DTD $d$:
  \[ \mathcal{L}(d) \in \text{ST}(1, \text{Depth}) \]

- There exists a DTD $d$, such that for every $m \in o(\text{Depth}(t))$:
  \[ \mathcal{L}(d) \notin \text{ST}(1, m) \]
Proof: $\mathcal{L}(d) \in \text{ST}(1, \text{Depth})$

Proof idea (Upper bound)

- Let $k$ be a stack and $t$ an XML document.
- For each $a \rightarrow R_a$, let $A_a = (Q_a, \Sigma, \delta_a, i_a, F_a)$ be a FSA.

```
if $t.\text{NextTag} = r$ then
    $k.\text{push}([r, i_r])$
else
    return false
end if
```

```
for $g \leftarrow t.\text{NextTag}$ do
    $[b, q] \rightarrow k.\text{pop}$
    if $g \in \Sigma$ then
        $k.\text{push}([b, \delta_b(q, a)])$
        $k.\text{push}([a, i_a])$
    else if $q \notin F_b$ then
        return false
    end if
end for
return true
```
Outline

Notation

XML validation

XML filtering
We consider (Core) XPath as the query language

Example

$Q_1 = /{\text{descendant :: } a[\text{child :: } b]} /{\text{child :: } c}$
$= //a[b]/c$

$Q_2 = /{\text{descendant :: } a[{\text{descendant :: } c}]}$
$= //a[}//c]$
XML filtering definition

We define a boolean XPath query $Q_B$:

$$Q_B(t) = 1 \text{ iff } Q(t) \neq \emptyset$$

Definition

Given a boolean XPath query $Q$, XML filtering is the problem to evaluate $Q(t)$.

$$\mathcal{L}(Q) = \{ t \in \text{Docs} \mid Q(t) = 1 \}$$

We only need to find one node that satisfies $Q$. 
The memory required to evaluate an XPath Query is in \( \Theta(\text{Depth}(t)) \)

**Theorem [GKS07]**

- For every XPath query \( Q \):
  \[
  \mathcal{L}(Q) \in \text{ST}(1, \text{Depth})
  \]

- There exists an XPath query \( Q \), such that for every \( m \in o(\text{Depth}(t)) \):
  \[
  \mathcal{L}(Q) \notin \text{ST}(1, m)
  \]

**Proof idea (Upper bound)**

- Every Core XPath query is equivalent to a unary MSO query.
- Every MSO query is recognizable by a unranked tree automaton.
- Use a stack based algorithm.
Theorem [GKS07]

There exists an XPath query $Q$ such that for every functions $s$ and $m$:

$$\mathcal{L}(Q) \notin \text{ST}(s, m) \quad \text{if} \quad s(t) \cdot m(t) \in o(\text{Depth}(t))$$

Proof idea.
We use communication complexity.
Communication complexity strategy

Proof idea.

- By contradiction, suppose that $L(Q) \in ST(s, m)$ for every $Q$.
- Let $N = \{1, \ldots, n\}$ and $F : 2^N \times 2^N \to \{0, 1\}$ such that:

  $$\text{com-complex}(F) = \Omega(n).$$

- We define $Q_F$ and $t_{xy}$ with $\text{Depth}(t_{xy}) \in \Theta(n)$ such that:

  $$Q_F(t_{xy}) = 1 \iff F(x, y) = 1$$

  $$t_{xy} = r \ a \ b \ \bar{b} \ \cdots \ a \ \bar{a} \ b \ \bar{b} \ a \ \bar{a} \ \cdots \ b \ \bar{b} \ \bar{a} \ \bar{r}$$

  $$\text{com-complex}(F) \leq s(t_{xy}) \cdot m(t_{xy}) \in o(n) \ \Rightarrow \Leftarrow$$
Proof idea of XML filtering lower bound

Let $F_{NonDisj} : 2^N \times 2^N \rightarrow \{0, 1\}$ such that

$$F_{NonDisj}(X, Y) = 1 \iff X \cap Y \neq \emptyset$$

Lemma

\[ \text{com-complex}(F_{NonDisj}) \in \Omega(n) \]

Let $\{x_i\}_{i \leq n}$ and $\{y_i\}_{i \leq n}$ be boolean variables such that:

$$x_i = 1 \quad \rightarrow \quad i \in X$$
$$y_i = 1 \quad \rightarrow \quad i \in Y$$

Given $X, Y \subseteq \{1, \ldots, n\}$, we define $t_{xy}$. 
Proof idea of XML filtering lower bound

We define:

\[ Q_{NonDisj} = \mathop{/ /}_{center[right/1]/left/1} \]

Notice that:

\[ Q_{NonDisj}(t_{xy}) = 1 \text{ iff } F_{NonDisj}(x, y) = 1 \]

Thus, if \( s(t_{xy}) \cdot m(t_{xy}) \in o(\text{Depth}(t_{xy})) \) then:

\[ \text{com-complex}(F_{NonDisj}) \in o(n) \Rightarrow \Leftarrow \]
More comments about XML filtering

Theorem [BYFJ07]

For every Redundancy-free XPath query $Q$ and for every function $m \in o(\log(Depth(t)))$:

$$\mathcal{L}(Q) \not\in ST(1, m)$$

A Redundancy-free XPath query is:
- star-restricted,
- conjunctive,
- univariate,
- leaf-only-value-restricted, and
- strongly subsumption-free.
Indexed XML streams

- One stream for each label.
- Index for each node:

\[ \text{Index} = (\text{Begin}, \text{End}, \text{Level}) \]

**Example**

- \( \text{left} = (2, 4, 2) (6, 8, 3) (10, 12, 4) \ldots \)
- \( \text{center} = (1, 8n, 1) (5, 8n - 4, 2) (9, 8n - 8, 3) \ldots \)
- \( \text{right} = (4n + 1, 4n + 3, n + 1) (4n + 5, 4n + 7, n) \ldots \)

**Motivation:**

- create an index over the XML document in order to reduce the cost of query evaluation.
For indexed XML streams, \( \Omega(Depth(t)) \) memory is still required.

**Theorem [SBY08]**

There is an XPath query \( Q \) such that every XML filtering algorithm over multiple indexed XML streams of \( t \) needs \( \Omega(Depth(t)) \) of memory.

**Proof idea.**
- Same principles of communication complexity.
- Other communication model is needed.
  - Token-based mesh communication (TMC)
Proof idea of XML filtering lower bound for indexed XML streams

Let $F_R: \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}$:

$$F_R(x, y) = 1 \text{ iff } x_i = (y^R)_i = 1 \text{ for some } i$$

Where $y^R$ is the reverse of $y$.

Lemma

$F_R$ cannot be computed by a deterministic algorithm that performs one pass over each stream and that uses less than $n - \log(n + 1) - 3$. 
Proof idea of XML filtering lower bound for indexed XML streams

For $x, y \in \{0, 1\}^n$, let $u_i \in \{a, c\}$ and $v_i \in \{b, c\}$:

$u_i = a \iff x_i = 1$
$\quad$ $v_i = b \iff y_i = 1$

Define an indexed XML document $t_{xy}$ and query $Q_R$:

$Q_R = //a/b$

Notice that:

$Q_R(t_{xy}) = 1 \iff F_R(x, y) = 1$
Conclusions

- Strongly validation with constant memory is only possible for non-recursive DTDs.

- A characterization for weak-validation with constant memory is an open problem.

- The memory needed for streaming XML validation and filtering is in $\Theta(\text{Depth}(t))$. 
Bibliography


