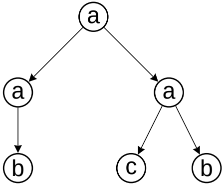
XML Documents



Document object model (DOM)

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

$$<a>\ <a>\ \ \ \ <a>\ <c>\ </c>\ <b>\ \ldots$$

$$a \quad a \quad b \quad \bar{b} \quad \bar{a} \quad a \quad c \quad \bar{c} \quad b \quad \ldots$$

Two main questions

- XML Validation with respect to a DTD:

$$\begin{array}{lcl} \mathsf{r} & \rightarrow & \mathsf{a}^{+} \\ \mathsf{a} & \rightarrow & \mathsf{a}^{+} \mid \mathsf{b}^{+} \mid \epsilon \\ \mathsf{b} & \rightarrow & \epsilon \end{array}$$

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

- XML Filtering for XPath queries:

$$/\mathit{descendant}::\mathit{a}[\mathit{child}::\mathit{b}]/\mathit{child}::\mathit{c}$$

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

First problem: XML validation

Example

- d_1 :
$$\begin{array}{lcl} \mathsf{r} & \rightarrow & \mathsf{a}^* \\ \mathsf{a} & \rightarrow & \mathsf{b}^* \\ \mathsf{b} & \rightarrow & \epsilon \end{array}$$

$$\mathcal{L}(d_1) = \mathsf{r}\ \ (\mathsf{a}\ \ (\mathsf{b}\ \ \bar{\mathsf{b}})^*\ \ \bar{\mathsf{a}})^*\ \ \bar{\mathsf{r}} \quad \checkmark$$

- d_2 :
$$\begin{array}{lcl} \mathsf{r} & \rightarrow & \mathsf{a} \\ \mathsf{a} & \rightarrow & \mathsf{a} \mid \epsilon \end{array}$$

$$\mathcal{L}(d_2) = \{\ \mathsf{r}\ \ (\mathsf{a}^n\ \ \bar{\mathsf{a}}^n)\ \ \bar{\mathsf{r}}\ \ \mid n \in \mathbb{N}\} \quad \times$$

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

$$\Theta(\mathrm{Depth}(t))$$

Second problem: XML filtering

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

- One scan:

$$t: \; r \;\; a \;\; b \;\; \overline{b} \;\; \overline{a} \;\; a \;\; a \;\; \overline{a} \;\; \overline{a} \;\; \ldots$$

(1-time) $\Uparrow$

- Multiple scans:

$$t: \; r \;\; a \;\; b \;\; \overline{b} \;\; \overline{a} \;\; a \;\; a \;\; \overline{a} \;\; \overline{a} \;\; \ldots$$

(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: Σ and $\bar{\Sigma}$.
- Tags alphabet: $\Delta = \Sigma \cup \bar{\Sigma}$.
- We consider the set of *well formed* XML documents:

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

- We use the following notation:
  - $t =$ XML document.
  - $d =$ DTD.
  - $Q =$ an XPath query.

Outline

Notation

XML validation

XML filtering

Validation with respect to a DTD (Document Type Definition)

Definition

A DTD $d = (\mathrm{r}, R)$ over Δ^* is a tuple where:

- $\mathrm{r} \in \Sigma$ is the root label, and
- $R = \{a \to R_a \mid a \in \Sigma\}$ with R_a a regular expression over Σ.

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

$$\mathcal{L}(d) = \{t \in \mathrm{Docs} \mid t \models d\}$$

Example

$$\begin{array}{lcl} \mathrm{r} & \to & \mathrm{a}^* \\ \mathrm{a} & \to & \mathrm{b}^* \\ \mathrm{b} & \to & \epsilon \end{array}$$

Two possible flavors of XML Validation

- Well-formed $\Rightarrow t \in$ Docs

Example

$$\mathrm{r\,a\,b\,\bar{b}\,\bar{a}\,a\,\bar{a}\,\bar{r}} \quad \rightarrow \quad \text{well-formed}$$
$$\mathrm{r\,a\,b\,\bar{b}\,\bar{a}\,a\,\bar{r}} \quad \rightarrow \quad \text{not well-formed}$$

- 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 Σ.

Definition

We define the implication graph $\mathcal{G}_d = (V, E)$ of d where:

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

Example

$$d: \begin{array}{lcl} \mathrm{r} & \to & \mathrm{a}^* \\ \mathrm{a} & \to & \mathrm{a} \mid \epsilon \end{array} \qquad \mathcal{G}_d: \begin{tikzcd} \mathrm{r} \arrow[r] & \mathrm{a} \arrow[loop right] \end{tikzcd}$$

d is *non-recursive* iff $\mathcal{G}_d$ 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.

($\Rightarrow$) By pumping argument.

($\Leftarrow$)

- For each $\mathrm{b} \to R_{\mathrm{b}}$ construct the automaton $\mathcal{A}_{\mathrm{b}}$ such that:

$$\mathcal{L}(\mathcal{A}_{\mathrm{b}}) = \mathcal{L}(\mathrm{b}' \cdot R_{\mathrm{b}} \cdot \overline{\mathrm{b}'})$$

- Construct $\mathcal{A}_0 = \mathcal{A}_r, \ldots, \mathcal{A}_i$, 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

$$d: \begin{array}{lcl} \text{r} & \to & \text{a}^* \\ \text{a} & \to & \text{a} \mid \epsilon \end{array}$$

$$\mathcal{L}(d) = \text{Docs} \ \cap \ \mathcal{L}(\text{r a}^* \overline{\text{a}}^* \ \overline{\text{r}})$$

Not all XML documents can be weakly validated with constant memory

Example

$$d_2 : \begin{array}{lcl} \mathsf{r} & \to & \mathsf{a} \cdot \mathsf{b} \cdot \mathsf{a} \\ \mathsf{a} & \to & \mathsf{a} \mid \epsilon \\ \mathsf{b} & \to & \epsilon \end{array}$$

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

d_2 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].

$$\text{fully recursive DTD} \subsetneq \text{DTD}$$

- 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 $\mathrm{ST}(s, m)$, or $L \in \mathrm{ST}(s, m)$, 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:

$$\mathcal{L}(d) \in \mathrm{ST}(1, 1)$$

The memory required to validate a DTD is in $\Theta(\text{Depth}(t))$

Let $\text{Depth}(t)$ 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: $\mathcal{L}(d) \in \mathrm{ST}(1, \mathrm{Depth})$

Proof idea (Upper bound)

- Let k be a stack and t an XML document.
- For each $\mathrm{a} \to R_{\mathrm{a}}$, let $\mathcal{A}_{\mathrm{a}} = (Q_{\mathrm{a}}, \Sigma, \delta_{\mathrm{a}}, i_{\mathrm{a}}, F_{\mathrm{a}})$ be a FSA.

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

```
for g ← t.NextTag do
    [b, q] → k.pop
    if g ∈ Σ then
        k.push([b, δ_b(q, a)])
        k.push([a, i_a])
    else if q ∉ F_b then
        return false
    end if
end for
return true
```

□

Outline

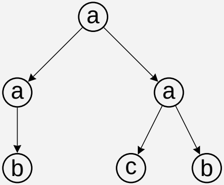
Notation

XML validation

XML filtering

We consider (Core) XPath as the query language

Example



$$\begin{aligned} Q_1 &= /descendant :: a[child :: b]/child :: c \\ &= //a[b]/c \end{aligned}$$

$$\begin{aligned} Q_2 &= /descendant :: a[descendant :: c] \\ &= //a[//c] \end{aligned}$$

XML filtering definition

We define a boolean XPath query Q_B:

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

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.

□

XML filtering with multiple scans

Theorem [GKS07]

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

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

Proof idea.

We use communication complexity.

Communication complexity strategy

Proof idea.

- By contradiction, suppose that $\mathcal{L}(Q) \in \mathrm{ST}(s, m)$ 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 $\mathrm{Depth}(t_{xy}) \in \Theta(n)$ such that:

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

$$t_{xy} = \mathrm{r}\ \overbrace{\mathrm{a}\ \mathrm{b}\ \overline{\mathrm{b}}\ \cdots\ \mathrm{a}\ \overline{\mathrm{a}}\ \mathrm{b}}^{\text{x (Alice)}}\ \overbrace{\overline{\mathrm{b}}\ \mathrm{a}\ \overline{\mathrm{a}}\ \cdots\ \mathrm{b}\ \overline{\mathrm{b}}\ \overline{\mathrm{a}}}^{\text{y (Bob)}}\ \overline{\mathrm{r}}$$

$$\text{com-complex}(F) \quad \leq \quad s(t_{xy}) \cdot m(t_{xy}) \quad \in \quad o(n) \quad \Rightarrow\Leftarrow$$

Proof idea of XML filtering lower bound

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

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

Lemma

com-complex($F_{NonDisj}$) $\in \Omega(n)$

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

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

Given $X, Y \subseteq \{1, \ldots, n\}$, we define t_{xy}.

Proof idea of XML filtering lower bound

We define:

$$Q_{NonDisj} = //center[right/1]/left/1$$

Notice that:

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

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

$$\text{com-complex}(F_{NonDisj}) \in o(n) \quad \Rightarrow\Leftarrow$$

$\square$

More comments about XML filtering

Theorem [BYFJ07]

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

$$\mathcal{L}(Q) \notin \mathrm{ST}(1, m)$$

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:

$$Index = (Begin, End, Level)$$

Example

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

Motivation:

- create an index over the XML document in order to reduce the cost of query evaluation.

For indexed XML streams, $\Omega(\text{Depth}(t))$ 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 $\Omega(\text{Depth}(t))$ 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 $F_R : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$:

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

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 $u_i \in \{\mathrm{a}, \mathrm{c}\}$ and $v_i \in \{\mathrm{b}, \mathrm{c}\}$:

$$u_i = \mathrm{a} \quad \text{iff} \quad x_i = 1$$
$$v_i = \mathrm{b} \quad \text{iff} \quad y_i = 1$$

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

$$Q_R = //a/b$$

Notice that:

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

$\square$

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

- Ziv Bar-Yossef, Marcus Fontoura, and Vanja Josifovski.
  On the memory requirements of xpath evaluation over xml streams.
  J. Comput. Syst. Sci., 73(3):391–441, 2007.
- Martin Grohe, Christoph Koch, and Nicole Schweikardt.
  Tight lower bounds for query processing on streaming and external memory data.
  Theor. Comput. Sci., 380(1-2):199–217, 2007.
- Mirit Shalem and Ziv Bar-Yossef.
  The space complexity of processing xml twig queries over indexed documents.
  In *ICDE*, pages 824–832, 2008.
- Luc Segoufin and Cristina Sirangelo.
  Constant-memory validation of streaming xml documents against dtds.
  In *ICDT*, pages 299–313, 2007.
- Luc Segoufin and Victor Vianu.
  Validating streaming xml documents.
  In *PODS*, pages 53–64, 2002.