XPath evaluation in linear time

Paweł Parys

Common work with Mikołaj Bojańczyk

XPath is a query language:
XPath queries select nodes in a XML document tree.
We consider a fragment called FOXPath.

Input: FOXPath query Q, XML document D
Output: document tree nodes,
which satisfy the query

Contribution: We solve the above problem
in time $O(|D| \cdot 2^{|Q|})$ or $O(|D| \cdot \log|D| \cdot |Q|^2)$

Example document:

```
<html>
  <title>Nice document</title>
  <h1>Section</h1>
  <a href=”http://www.uw.edu.pl/”>University</a>
  <a href=”http://www.google.com/”>Search</a>
  <table><tr><td>
    <a href=”http://www.uw.edu.pl/”>
      A link in a table</a>
  </td></tr></table>
</html>
```

Example document:

```
<html>
  <title>Nice document</title>
  <h1>Section</h1>
  <a href="http://www.uw.edu.pl/">University</a>
  <a href="http://www.google.com/">Search</a>
  <table><tr><td>
    <a href="http://www.uw.edu.pl/">
      A link in a table</a>
  </td></tr></table>
</html>
```

Example query – navigation only (CoreXPath):

```
self::"a" and not (ancestor::"table")
```

Example document:

```
<html>
  <title>Nice document</title>
  <h1>Section</h1>
  <a href="http://www.uw.edu.pl/">University</a>
  <a href="http://www.google.com/">Search</a>
  <table><tr><td>
    <a href="http://www.uw.edu.pl/">
      A link in a table</a>
  </td></tr></table>
</html>
```

Example query – comparing data (FOXPath):

```
self::"a" and ((self/@href=following::"a"/@href)
            or (self/@href=preceding::"a"/@href))
```

Example document:

```
<html>
  <title>Nice document</title>
  <h1>Section</h1>
  <a href=”http://www.uw.edu.pl/”>University</a>
  <a href=”http://www.google.com/”>Search</a>
  <table><tr><td>
    <a href=”http://www.uw.edu.pl/”>
      A link in a table</a>
  </td></tr></table>
</html>
```

Example query – counting, positional arithmetic (full XPath 1.0):

```
root/descendant[position()=
        count(root/descendant::”a”)]
```

CoreXPath (no data)

$O(|D| \cdot |Q|)$ - Gottlob, Koch, Pichler 2002

$O(|D|^2 \cdot 2^{|Q|})$ - real world XPath engines

FOXPath (comparing data)

$O(|D|^2 \cdot |Q|)$ - previous works

$O(|D| \cdot 2^{|Q|})$, $O(|D| \cdot \log|D| \cdot |Q|^2)$ – Bojańczyk, Parys 2008

$O(|D| \cdot |Q|^c)$ – work in progress...

Full XPath (counting, node positions)

$O(|D|^4 \cdot |Q|^2)$ - Gottlob, Koch, Pichler 2003

$O(|D|^3 \cdot |Q|^c)$ – work in progress...

Tool used: Simon's theorem

(I. Simon, Factorization forests of finite height, 1990)

Two players: **D**isbeliever, **S**imon

D: regular language L

S: number K

D: word u

repeated K times:

S: factorization $u=u_1...u_n$ where all u_i are equivalent

or $u=u_1u_2$

D: chooses one of u_i as new u

....

S wins, if after K steps u has only one letter

u and v are equivalent, if for any words w_1, w_2 there is

$w_1uw_2 \in L \Leftrightarrow w_1vw_2 \in L$

repeated K times

Two players: **D**isbeliever, **S**imon

D: regular language L

S: number K

D: word u

S: factorization $u=u_1...u_n$ where all u_i are equivalent

or $u=u_1u_2$

D: chooses one of u_i as new u

....

S wins, if after K steps u has only one letter

Example

L=(a+b)*b

a a a a b

Simon's theorem

Two players: **D**isbeliever, **S**imon

D: regular language L

S: number K

D: word u

S: factorization $u=u_1...u_n$ where all u_i are equivalent

or $u=u_1u_2$

D: chooses one of u_i as new u

....

(repeated K times)

S wins, if after K steps u has only one letter

Example

L=(a+b)*b

a a a a	b

Two players: **D**isbeliever, **S**imon

D: regular language L

S: number K

D: word u

repeated K times:

S: factorization $u=u_1...u_n$ where all u_i are equivalent

or $u=u_1u_2$

D: chooses one of u_i as new u

....

S wins, if after K steps u has only one letter

Example

L=(a+b)*b

a	a	a	a	b

repeated K times

Two players: **D**isbeliever, **S**imon

D: regular language L

S: number K

D: word u

S: factorization $u=u_1...u_n$ where all u_i are equivalent

or $u=u_1u_2$

D: chooses one of u_i as new u

....

S wins, if after K steps u has only one letter

Example

L=(a+b)*b

b b a a a b a b a a a

Two players: **D**isbeliever, **S**imon

D: regular language L

S: number K

repeated K times:

D: word u

S: factorization $u=u_1...u_n$ where all u_i are equivalent

or $u=u_1u_2$

D: chooses one of u_i as new u

....

S wins, if after K steps u has only one letter

Example

L=(a+b)*b

b b a a a b a b	a a a

Two players: **D**isbeliever, **S**imon

D: regular language L

S: number K

D: word u

repeated K times:

- **S**: factorization $u=u_1...u_n$ where all u_i are equivalent or $u=u_1u_2$
- **D**: chooses one of u_i as new u
- ....

S wins, if after K steps u has only one letter

Example

L=(a+b)*b

b	b	a a a b	a b	a	a	a

Corollary from Simon's theorem

Given a regular language L and a word u.
After preprocessing in time linear in the length of u
we may in constant time answer questions:

$u_i \ldots u_j \in L$?

Thank you