XPath Evaluation in Linear Time

Paweł Parys

Considered problem

Input:

XML document

XPath query

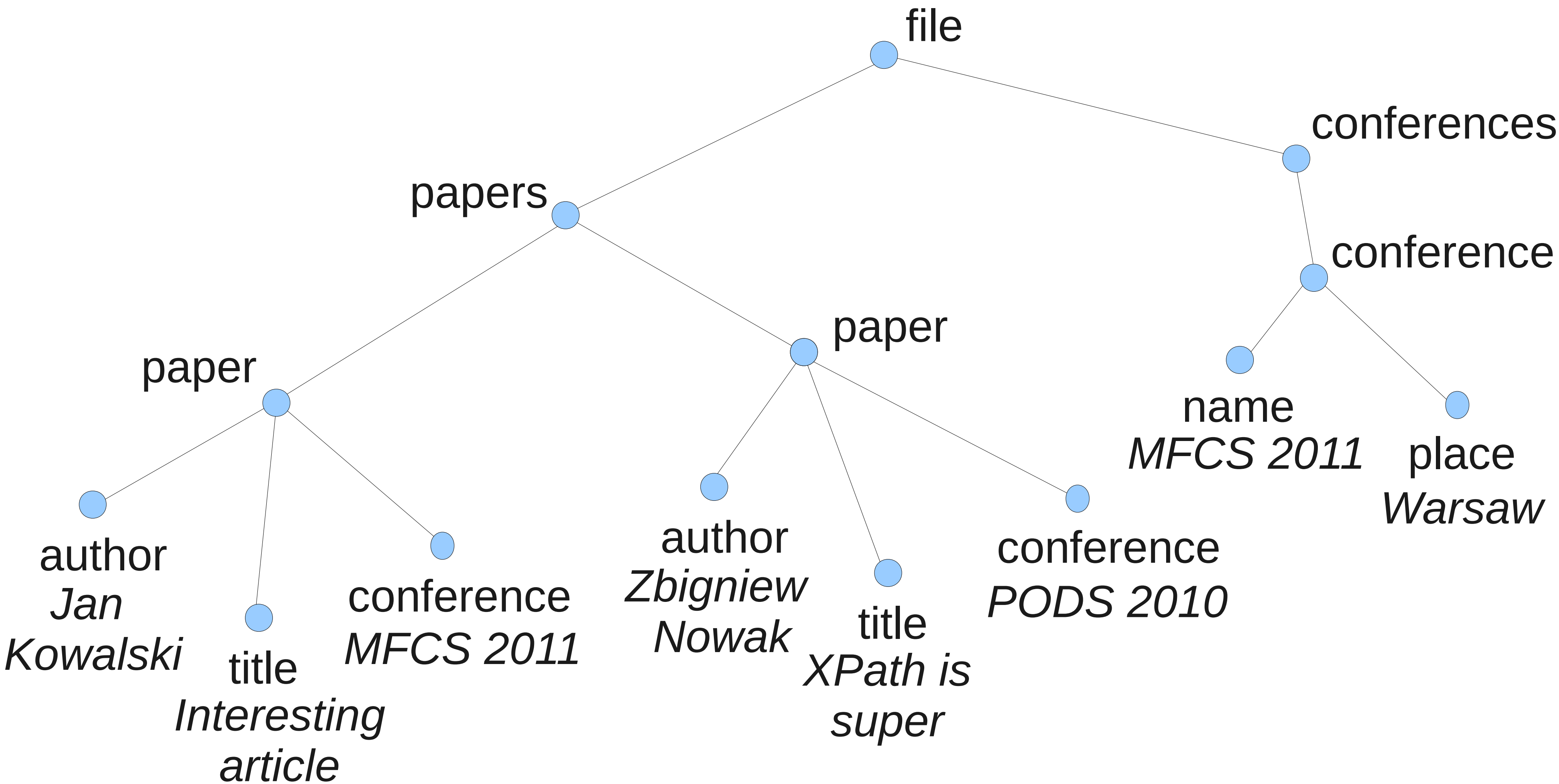
Output:

Nodes of the document satisfying the query

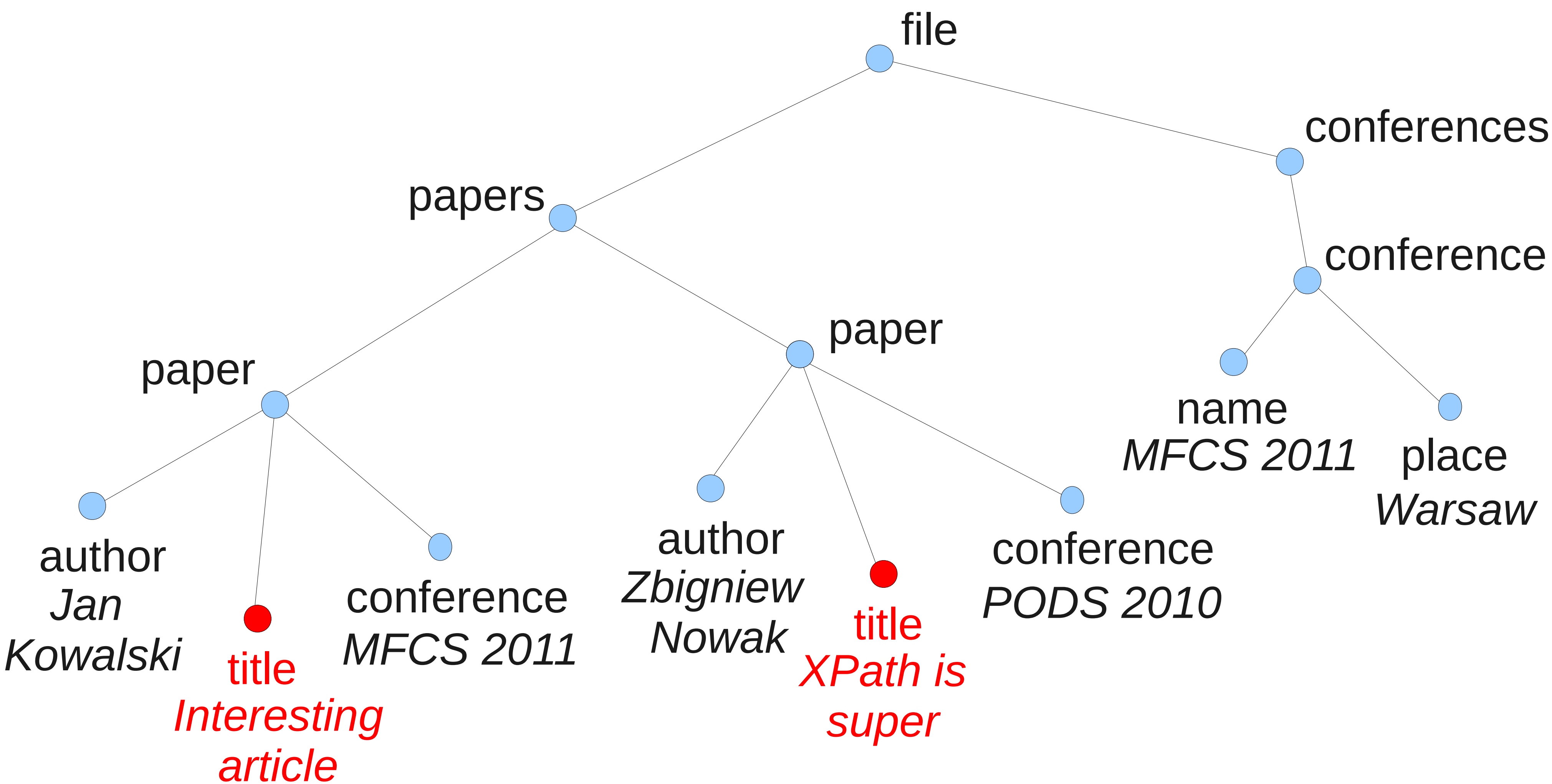
Example

```
<file>
  <papers>
    <paper>
      <author>Jan Kowalski</author>
      <title>Interesting article</title>
      <conference>MFCS 2011</conference>
    </paper>
    <paper>
      <author>Zbigniew Nowak</author>
      <title>XPath is super</title>
      <conference>PODS 2010</conference>
    </paper>
  </papers>
  <conferences>
    <conference>
      <name>MFCS 2011</name>
      <place>Warsaw</place>
    </conference>
  </conferences>
</file>
```

Example


file
papers
conferences
conference
paper
paper
name
MFCS 2011
place
Warsaw
author
Jan
Kowalski
title
Interesting
article
conference
MFCS 2011
author
Zbigniew
Nowak
title
XPath is
super
conference
PODS 2010

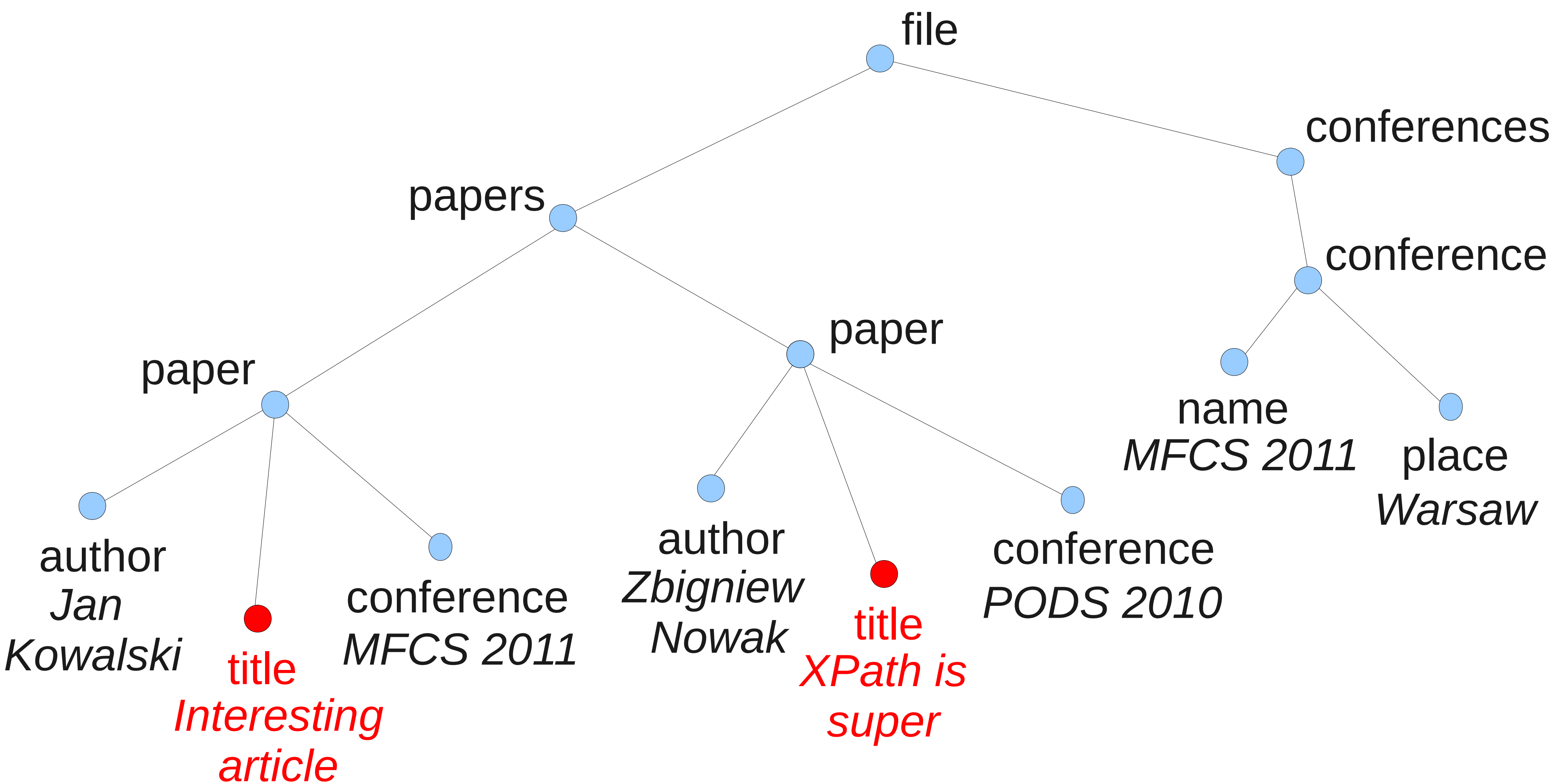
Example



Query:

file/papers/paper/title

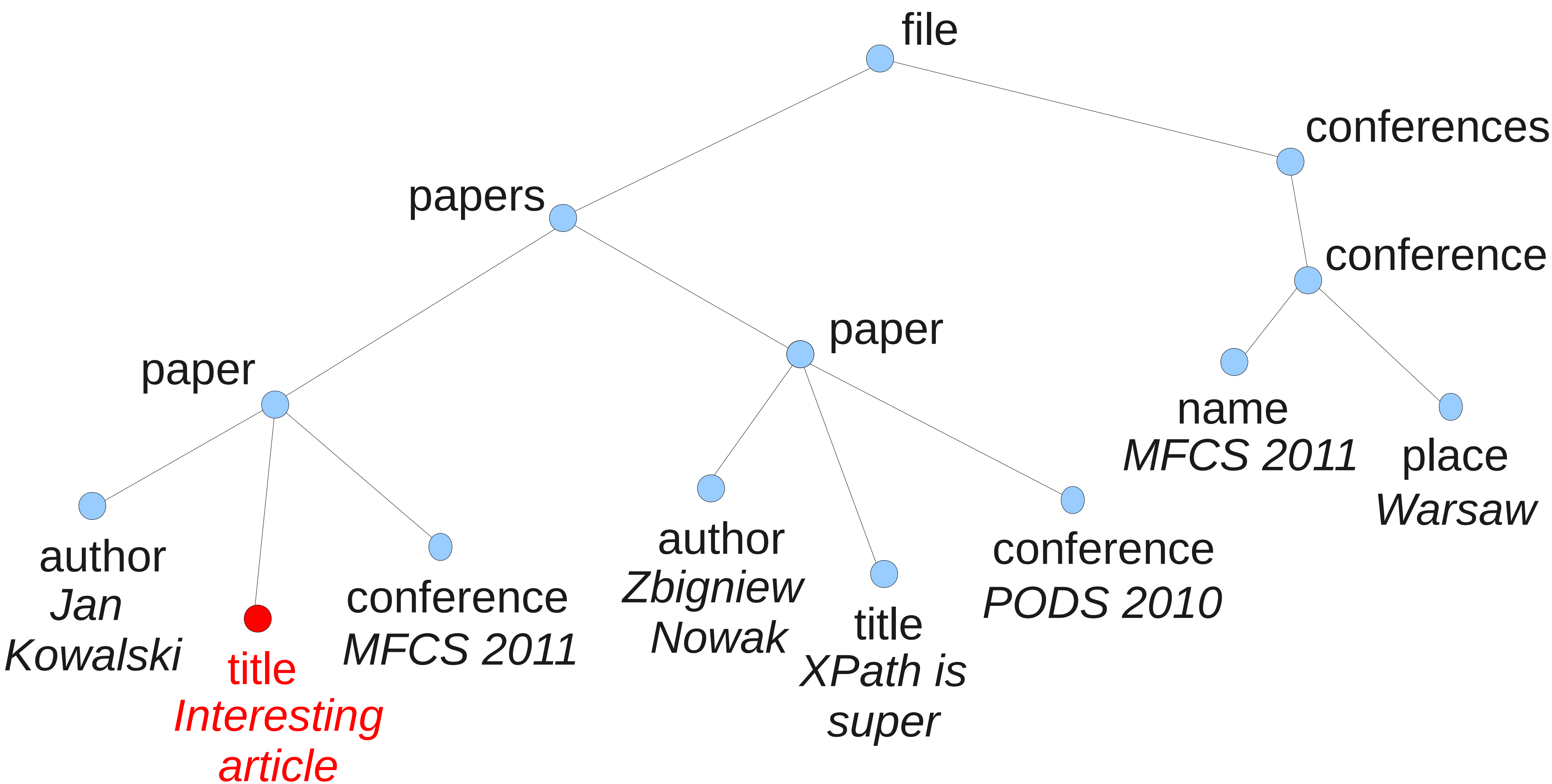
Example



Query:

file/papers/paper/../paper/../paper/../paper/../paper/
../paper/../paper/../paper/../paper/../paper/title

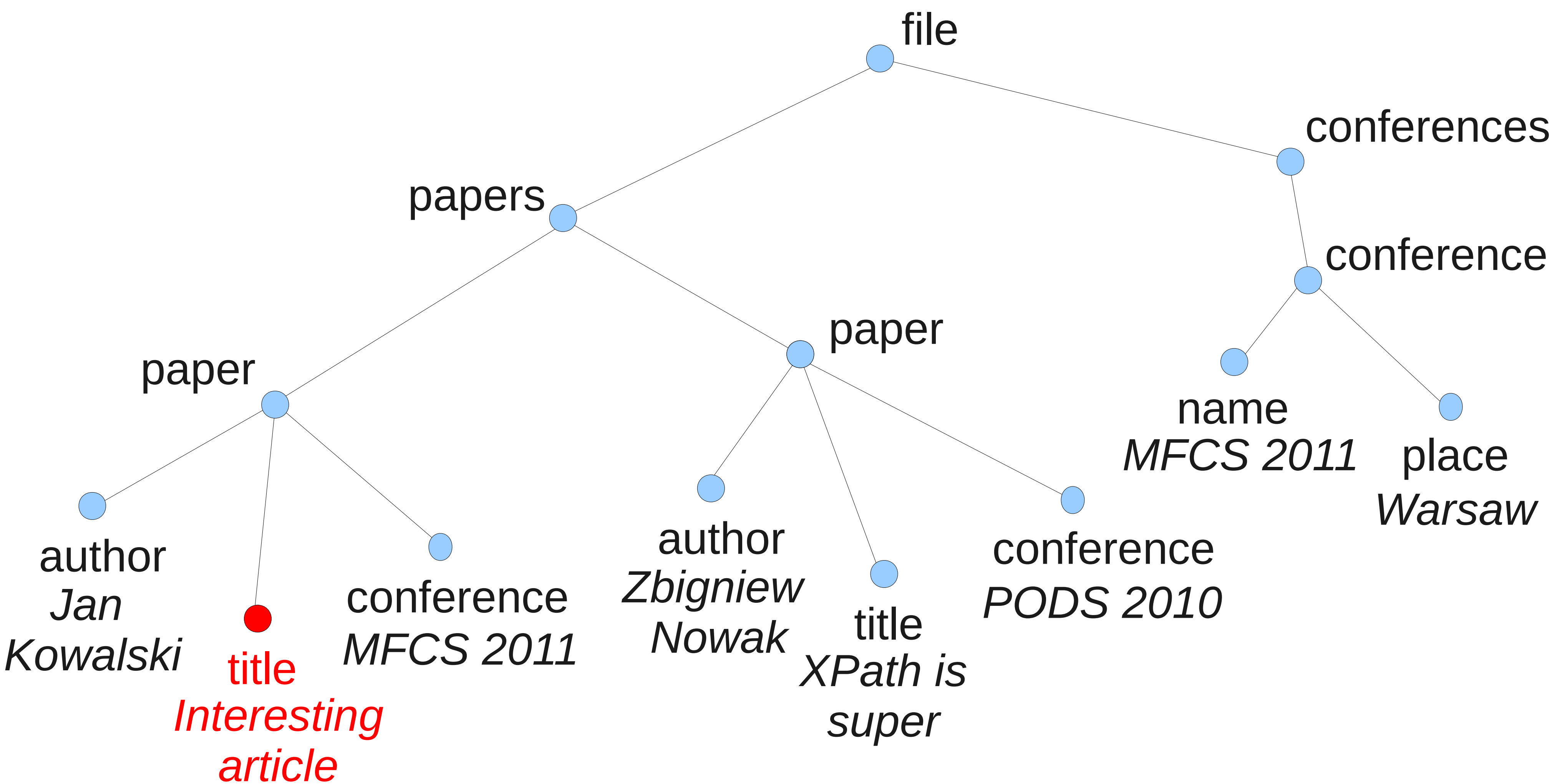
Example



Query:

file/papers/paper[author='Jan Kowalski']/title

Example



Query:

file/papers/paper[conference=
../../conferences/conference[place='Warsaw']/name]/title

Results summary

XML document
XPath query

→ Nodes of the document
Satisfying the query

XPath not refering to data

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

XPath with data (but without counting)

$O(D^2 \cdot Q)$ - Gottlob, Koch, Pichler 2002

$O(D \cdot Q^3)$ - our contribution

Where: D - document size
Q - query size

Subproblem:

Fix a regular language L. A word $u = a_1 \ldots a_n$ is given.

First, in time linear in n, we can prepare ourselves.

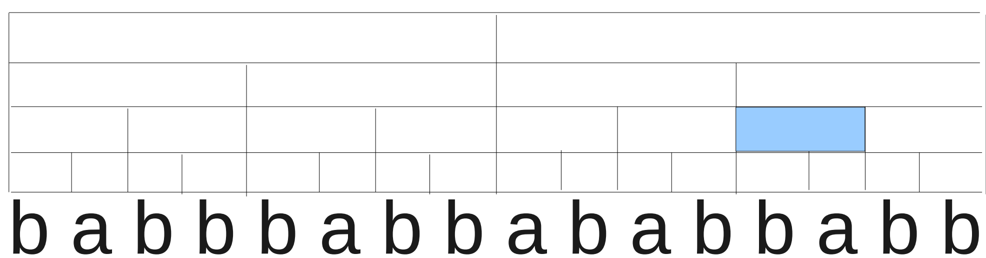
Then, in constant time we want to aswer queries:

$$a_i \ldots a_j \in L\,?$$

Subproblem:

Fix a regular language L. A word $u = a_1 \dots a_n$ is given.

Preprocessing: divide and conquer



For each subword remeber all possible automaton transitions: pairs of states p, q such that

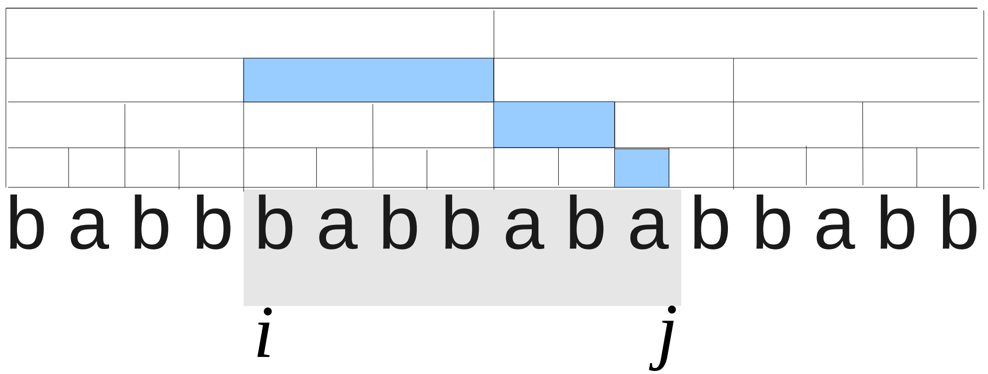
$$p \xrightarrow{a_i \dots a_j} q$$

time: $O(n)$

Subproblem:

Fix a regular language L. A word $u = a_1 \ldots a_n$ is given.

Given: i, j



Does $a_i \ldots a_j \in L$?

It is enough to compose remembered transitions!

time: $O(\log n)$

Subproblem:

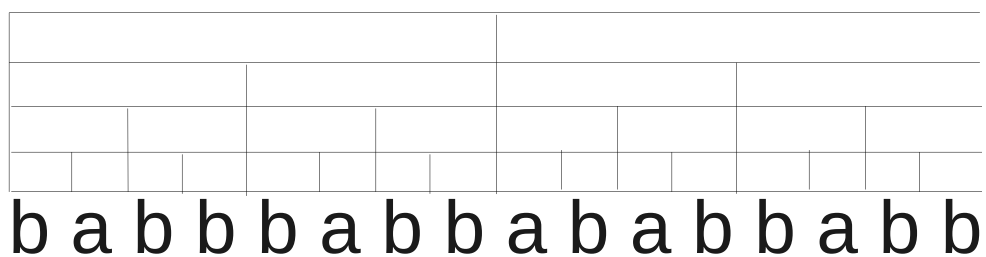
A tool used: Simon's theorem

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

Subproblem:

Fix a regular language L. A word $u = a_1 \ldots a_n$ is given.

In the „logarithmic” decomposition we always split into 2 parts



To achieve a constant height of the decomposition tree
we have to allow splits into arbitrarly many parts
- but then all parts have to be very similar

Simon's decomposition:

Every word u in the decomposition tree we split into

- 2 (arbitrary) parts $u = u_1 u_2$, or
- arbitrarly many parts $u = u_1 \ldots u_k$, where all $u_i \ldots u_j$ are equivalent.

Simon's Theorem:
For every word there exists such a decomposition tree of the same height.

u and v are equivalent, if for any words w_1, w_2 it holds
$w_1 u w_2 \in L \Leftrightarrow w_1 v w_2 \in L$

Simon's decomposition:

Every word u in the decomposition tree we split into
- 2 (arbitrary) parts $u = u_1 u_2$, or
- arbitrarly many parts $u = u_1 \ldots u_k$, where all $u_i \ldots u_j$ are equivalent.

Example

L=(a+b)*b

a a a a b

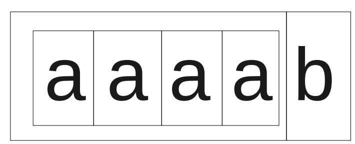
Simon's decomposition:

Every word u in the decomposition tree we split into

- 2 (arbitrary) parts $u = u_1 u_2$, or
- arbitrarly many parts $u = u_1 \ldots u_k$, where all $u_i \ldots u_j$ are equivalent.

Example

L=(a+b)*b



Simon's decomposition:

Every word u in the decomposition tree we split into

- 2 (arbitrary) parts $u = u_1 u_2$, or
- arbitrarly many parts $u = u_1 \ldots u_k$,
  where all $u_i \ldots u_j$ are equivalent.

Example

L=(a+b)*b

a a a a b

b b a a a b a b a a a

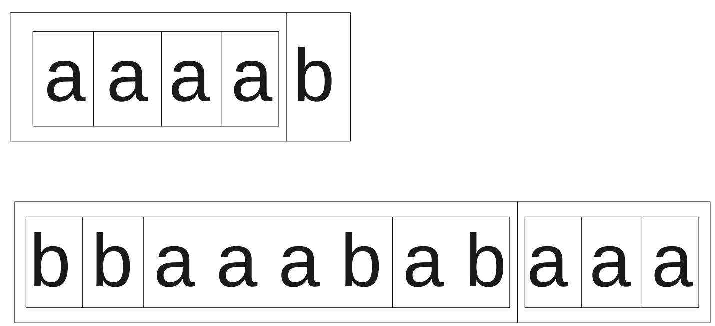
Simon's decomposition:

Every word u in the decomposition tree we split into

- 2 (arbitrary) parts $u = u_1 u_2$, or
- arbitrarly many parts $u = u_1 \ldots u_k$, where all $u_i \ldots u_j$ are equivalent.

Example

L=(a+b)*b



Subproblem:

Fix a regular language L. A word $u = a_1 \dots a_n$ is given.

Preprocessing:

- calculate the Simon's decomposition
- for every subword in the decomposition compute the transitions of the automaton

time: $O(n)$

Does $a_i \dots a_j \in L$?

- It is enough to compose remembered transitions

time: $O(1)$

Subproblem:

Fix a regular language L. A word $u = a_1 \ldots a_n$ is given.

Prepro

• calcu

• for ev

comp

time: (

Does

• It is

time:

Dependance on language L

Height of the decomposition tree is proportional to the number of abstraction classes, which is exponential in the automaton size.

However the tree has at most $2\mathrm{n}-1$ nodes.

<u>Our contribution:</u>
how to deal with this decomposition in time polynomial in the automaton size.

Thank you