Incremental XPath Evaluation

Wim Martens
Technical University of Dortmund

Joint work with:
Henrik Björklund
Wouter Gelade
Marcel Marquardt

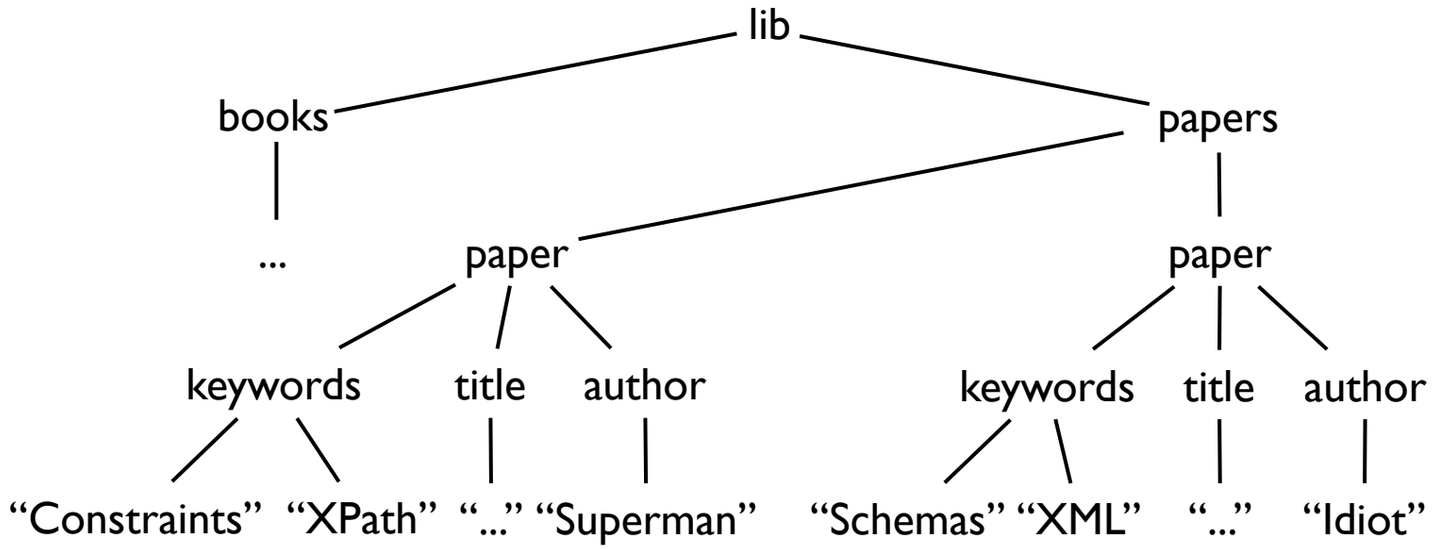
Outline

- Motivation
- Terminology
- Results
- Final Remarks

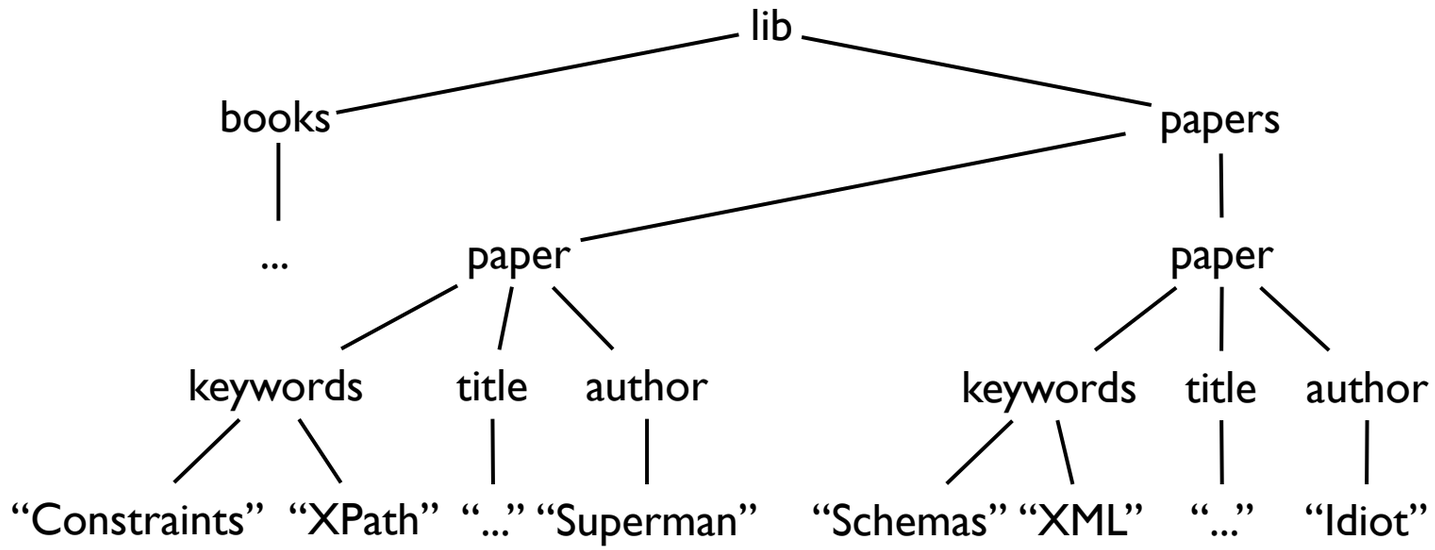
Outline

- Motivation
- Terminology
- Results
- Final Remarks

Motivation


lib
books
papers
...
paper
paper
keywords
title
author
keywords
title
author
“Constraints”
“XPath”
“...”
“Superman”
“Schemas”
“XML”
“...”
“Idiot”

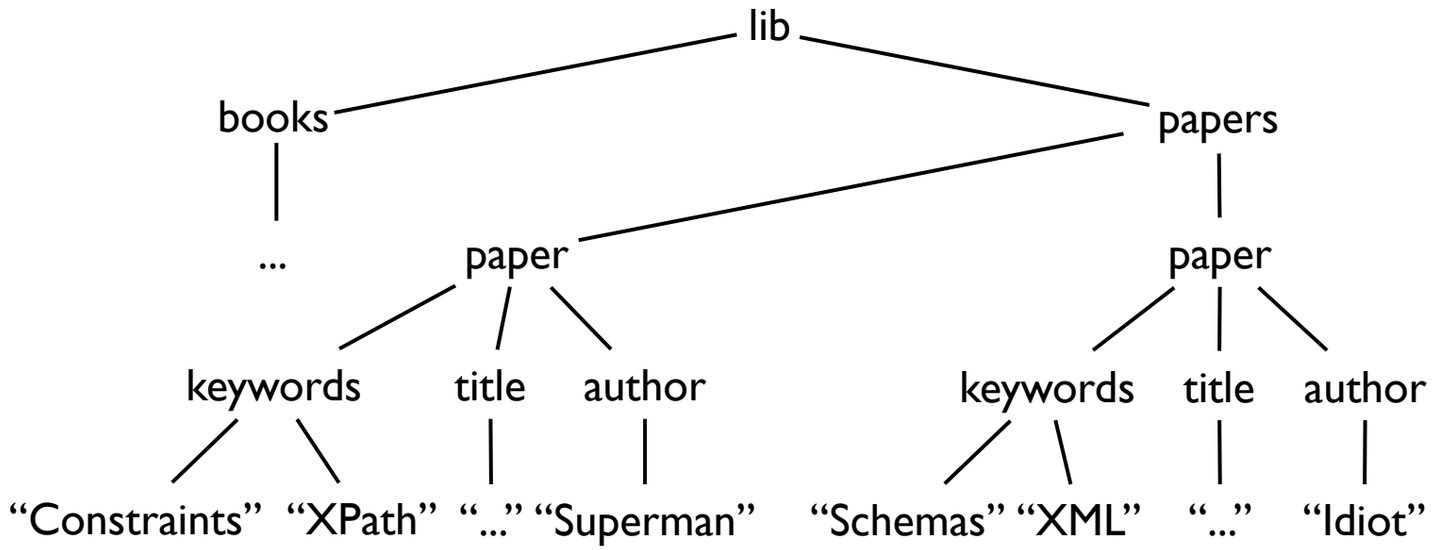
Motivation



I’m interested in:

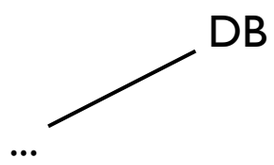
papers about XML or XPath which are not written by Idiot

Motivation

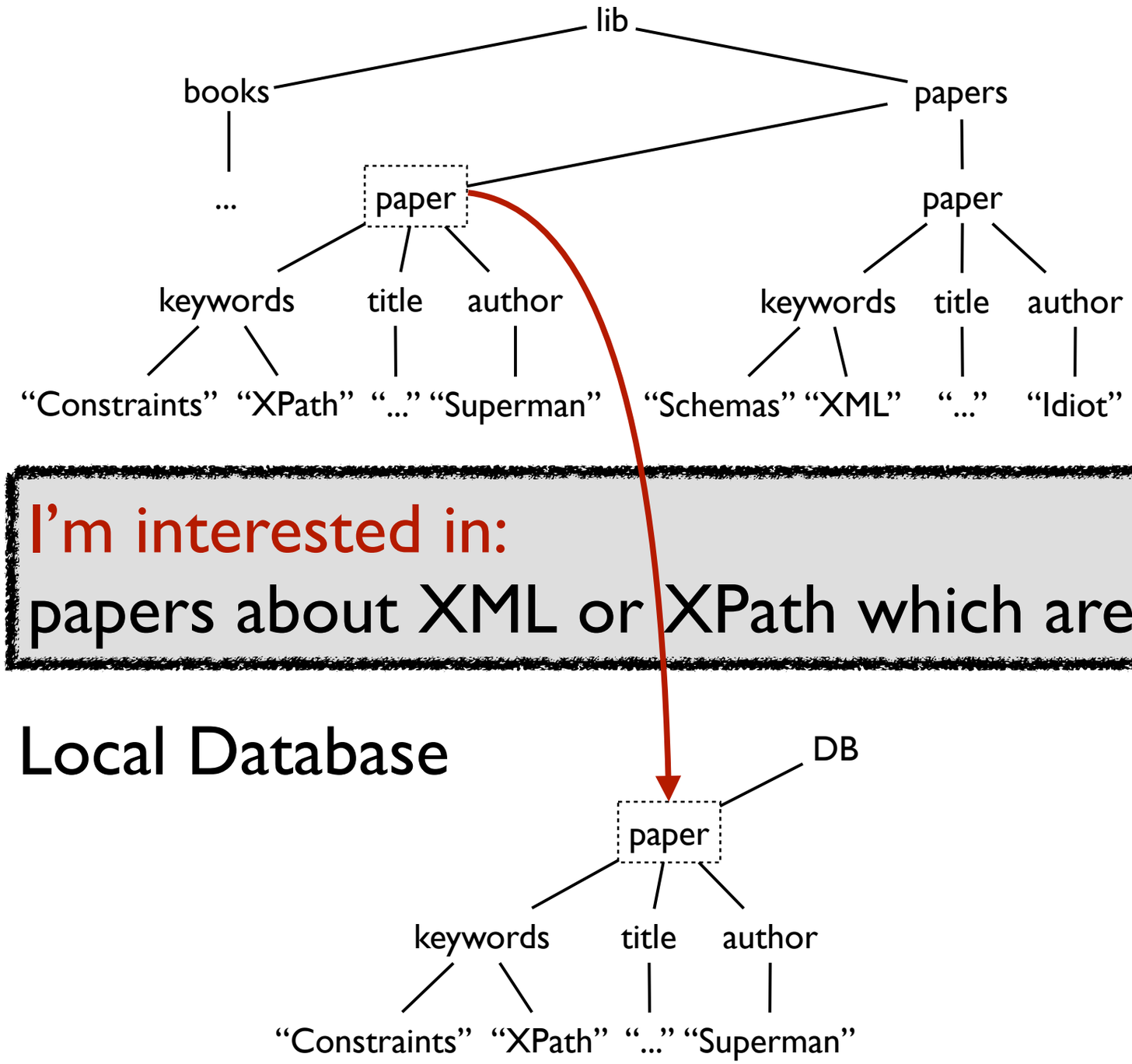


I’m interested in:
papers about XML or XPath which are not written by Idiot

Local Database



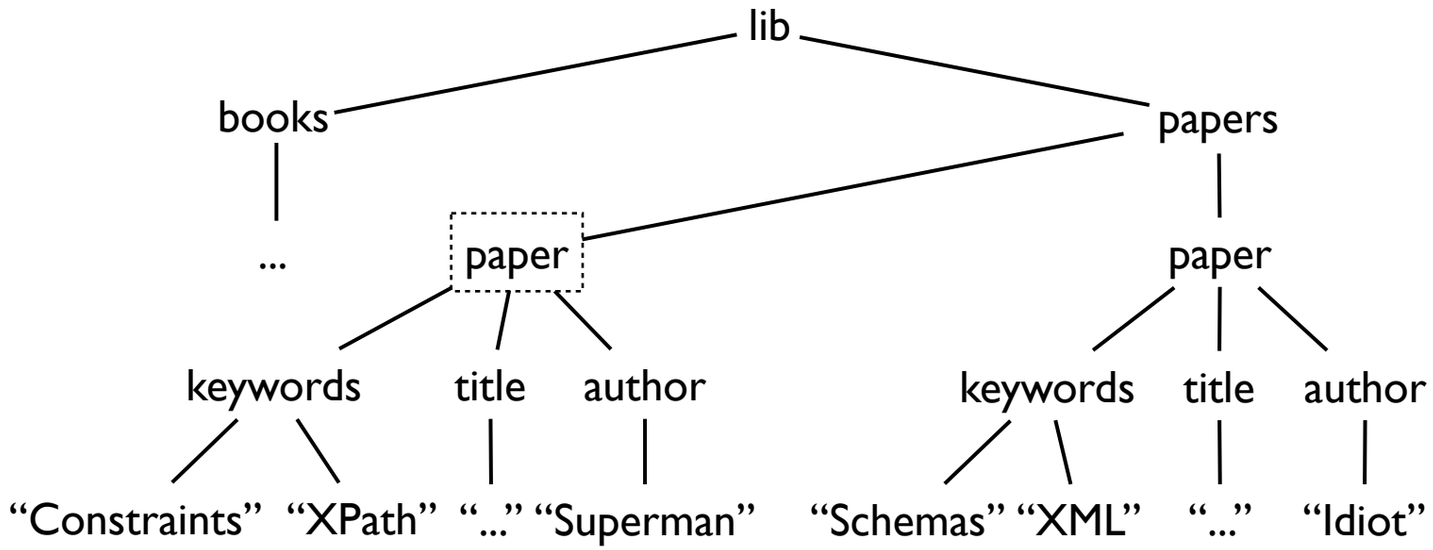
Motivation



I’m interested in:
papers about XML or XPath which are not written by Idiot

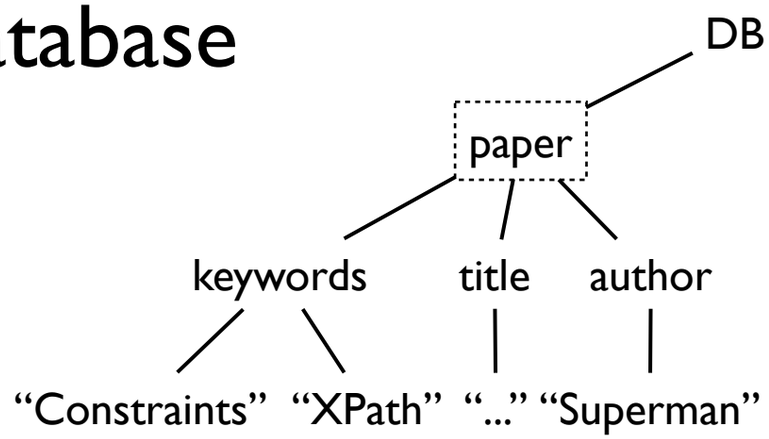
Local Database

Motivation

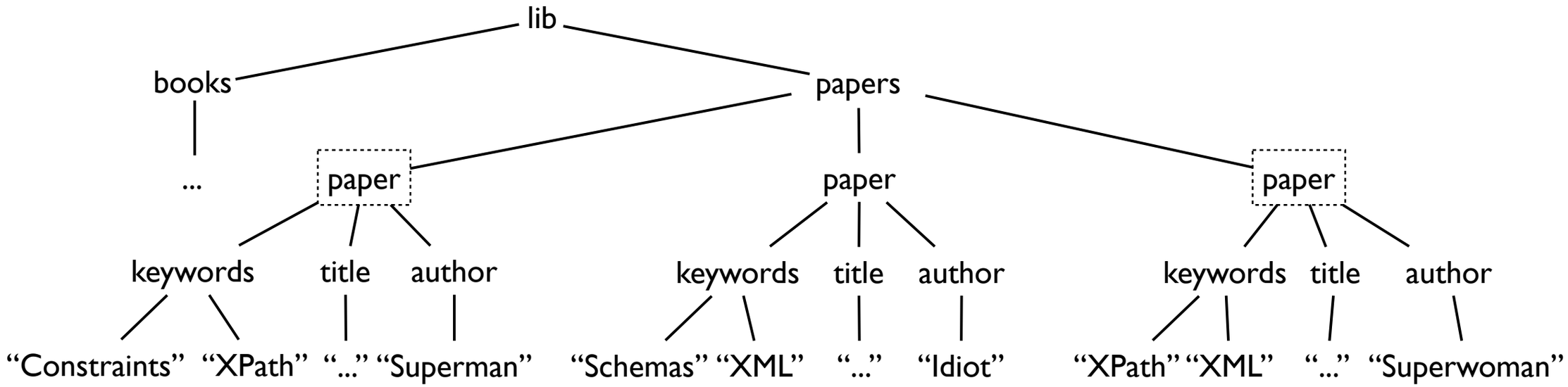


I’m interested in:
papers about XML or XPath which are not written by Idiot

Local Database

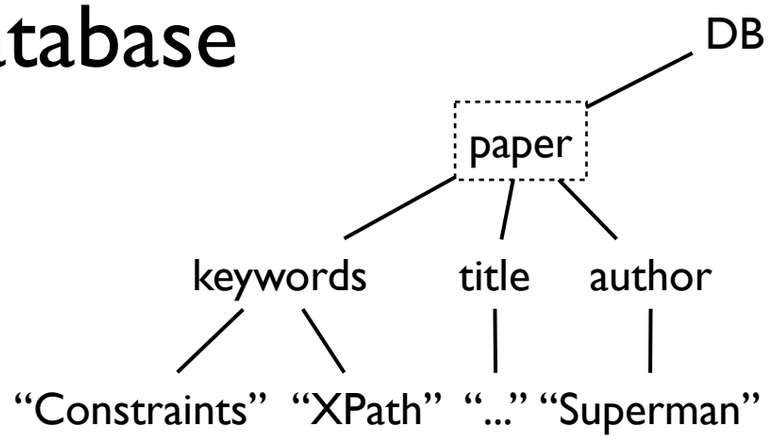


Motivation

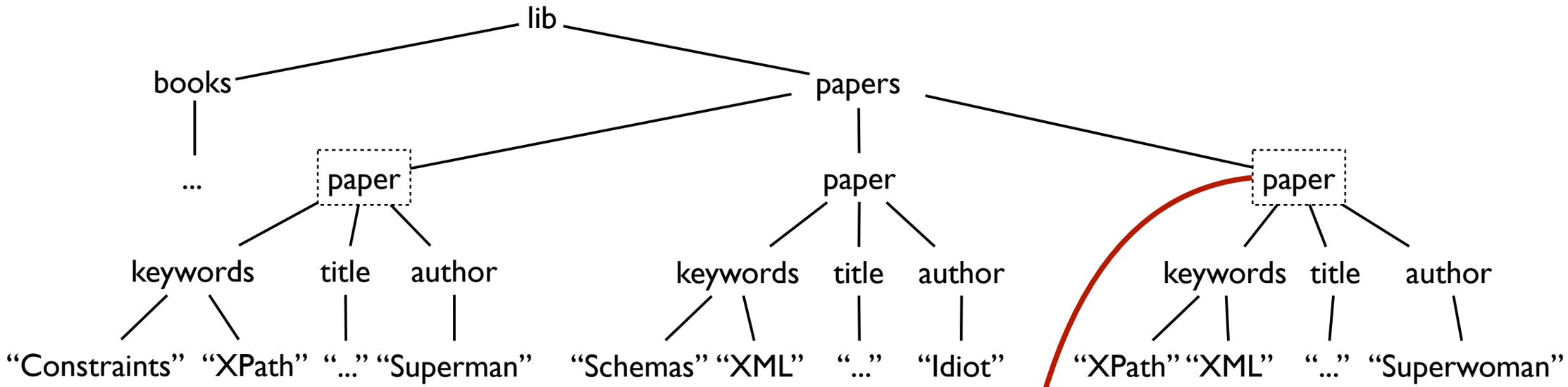


I’m interested in:
papers about XML or XPath which are not written by Idiot

Local Database

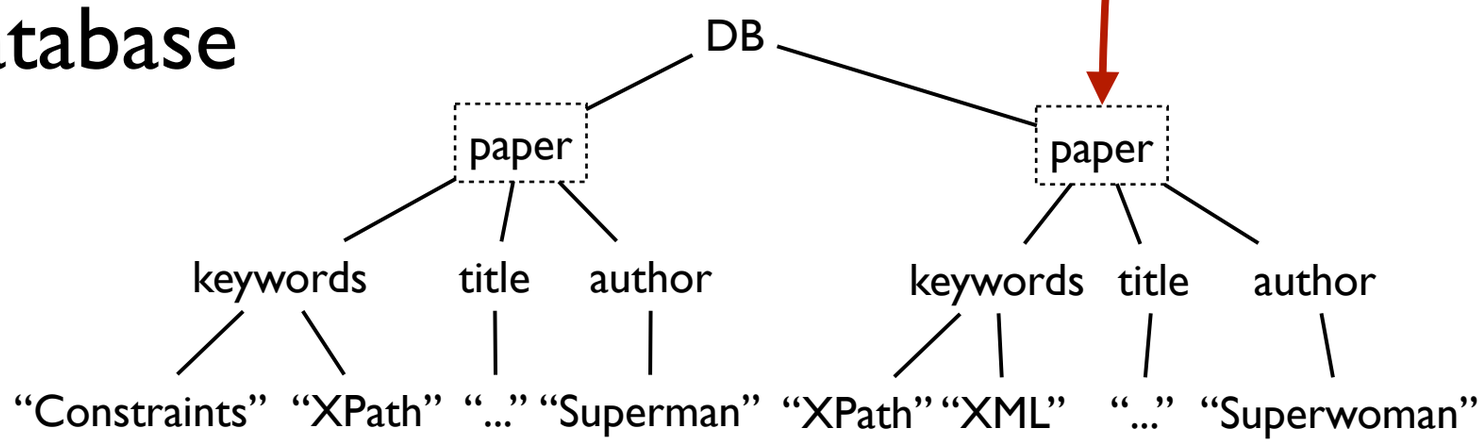


Motivation

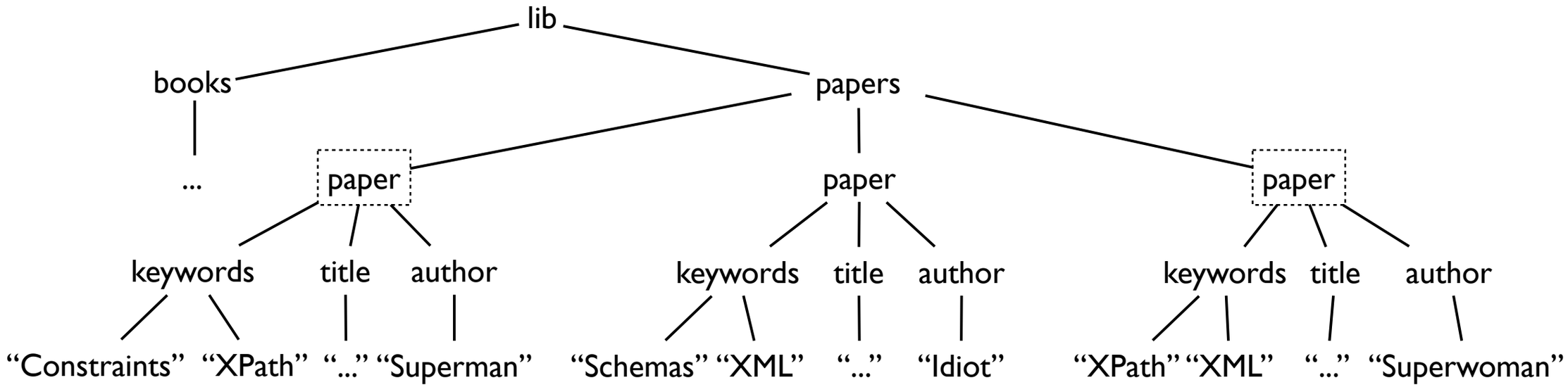


I’m interested in:
papers about XML or XPath which are not written by Idiot

Local Database

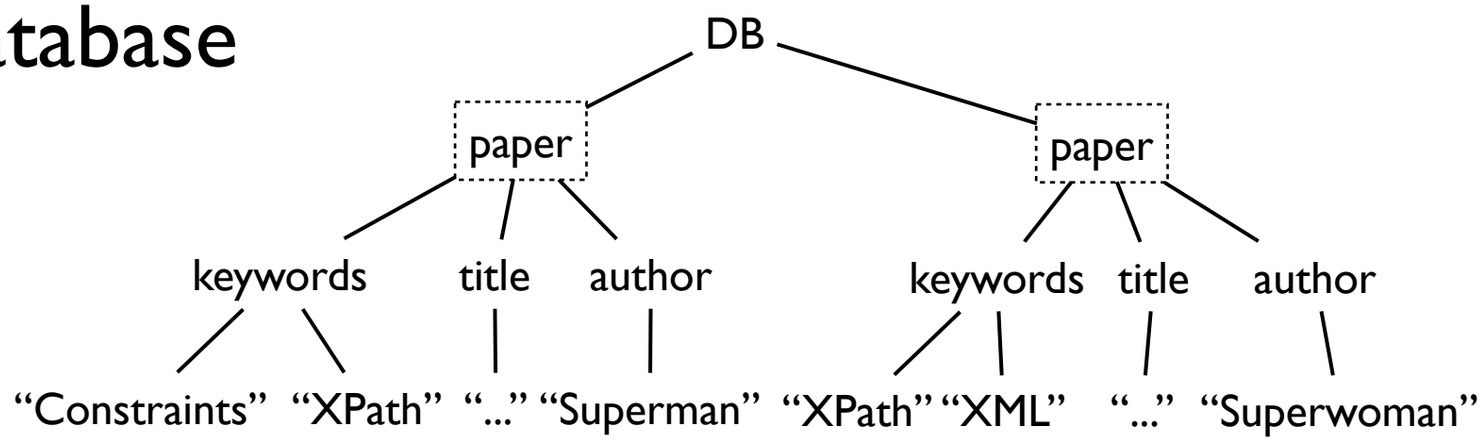


Motivation

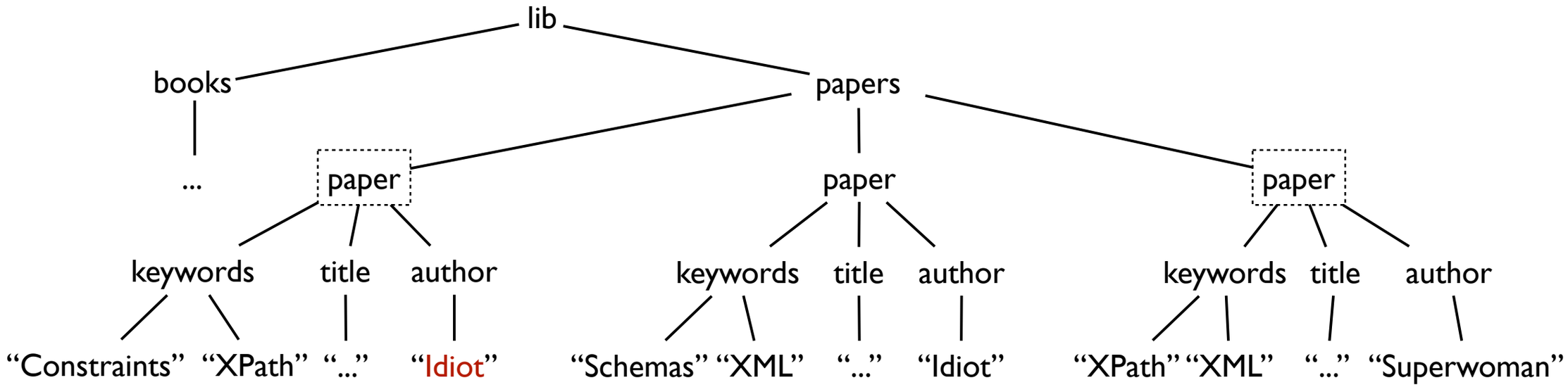


I’m interested in:
papers about XML or XPath which are not written by Idiot

Local Database

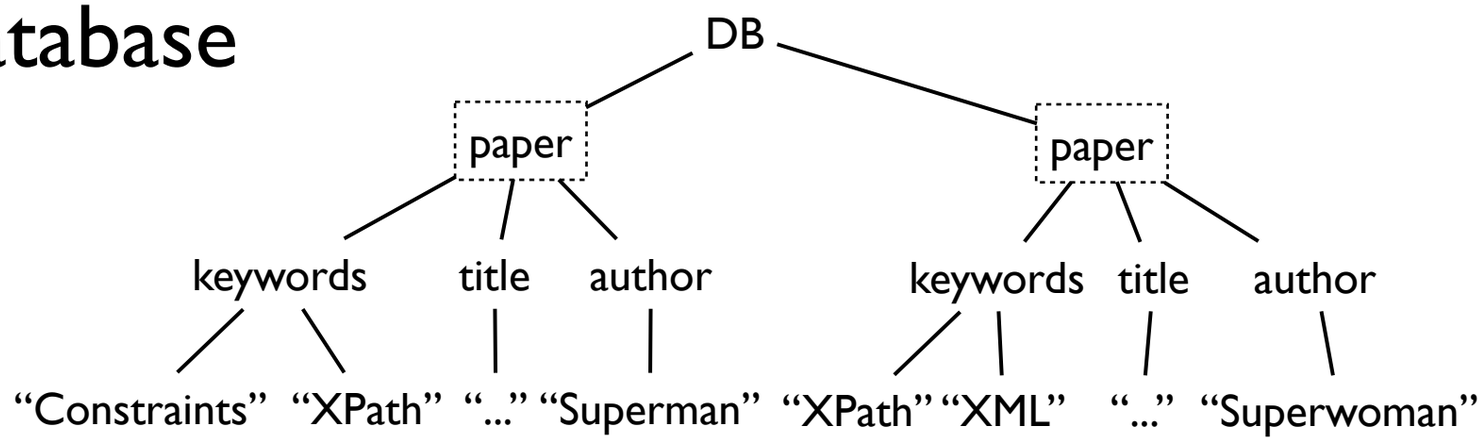


Motivation

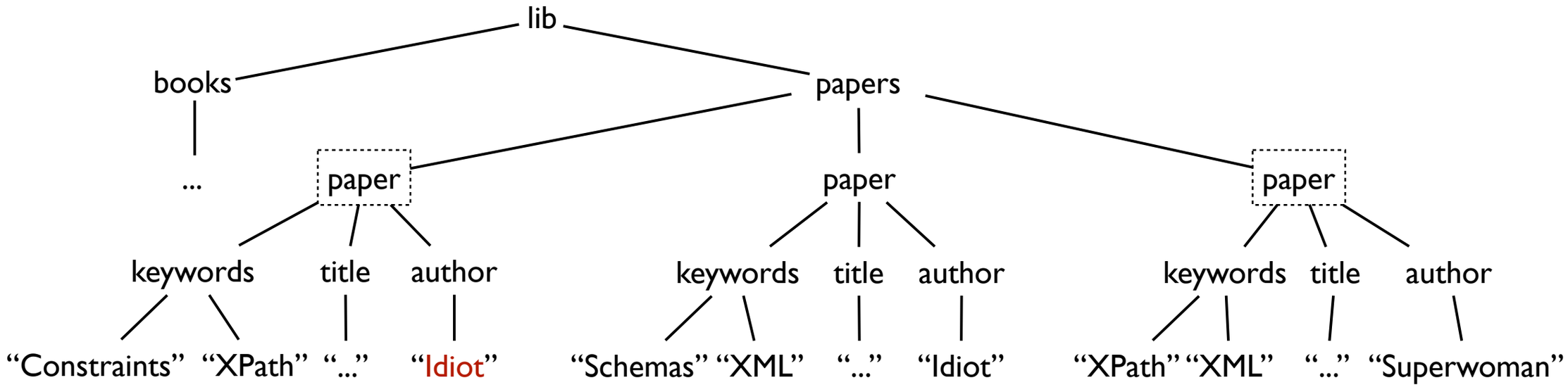


I’m interested in:
papers about XML or XPath which are not written by Idiot

Local Database

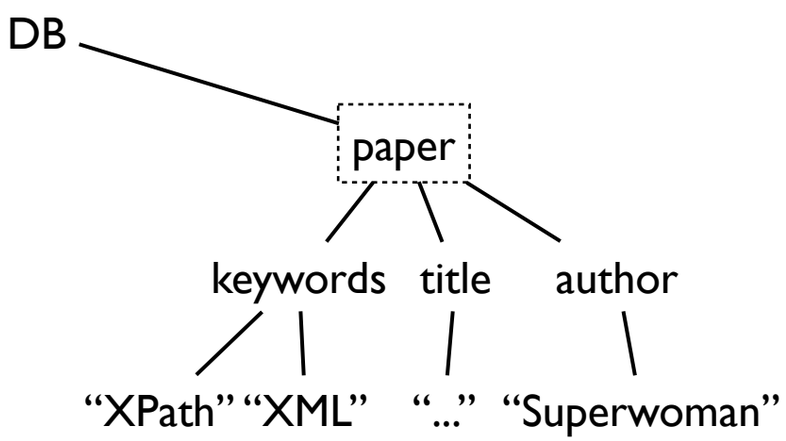


Motivation



I’m interested in:
papers about XML or XPath which are not written by Idiot

Local Database



Motivation

So this is:

Efficient XPath View Maintenance for XML Data

Outline

- Motivation
- Terminology
- Results
- Final Remarks

Terminology

Two Versions of the problem:

(1) We want to maintain a view:

Incremental View Maintenance

(2) We want to maintain (non)-satisfaction of a trigger:

Incremental Boolean Maintenance

Terminology

Two Versions of the problem:

(1) We want to maintain a view:

Incremental View Maintenance

(2) We want to maintain (non)-satisfaction of a trigger:

Incremental Boolean Maintenance

Incremental XPath Evaluation

Problem Definition

Incremental Boolean Maintenance

Given:
- XPath query Q
- XML document D
- Update u, that updates D to D’

Question:
Does D’$\models$Q?

(Does Q return a non-empty answer on D’?)

Problem Definition

Incremental Boolean Maintenance

Question: Does $D' \models Q$?

We can maintain an auxiliary data structure Aux(D)

Algorithms are evaluated w.r.t.:
- Size of Aux(D)
- Time needed to
  - compute whether $D' \models Q$
  - update Aux(D) to Aux(D')

Problem Definition

Incremental Boolean Maintenance

Updates:

- Relabel(u,a): overwrite label of u with a
- InsertNS(u,a): insert leaf labeled a as next sibling of u
- InsertFC(u,a): insert leaf labeled a as first child of u
- Delete(u): delete subtree rooted at u

Problem Definition

Incremental View Maintenance

Similar to Boolean maintenance, but:

We want to maintain the set Q(D) of output nodes

Given update u that updates D to D',
compute update v that updates Q(D) to Q(D')

Wishful Thinking

Prove that XPath Maintenance is possible in
- time polylog(D) . poly(Q)
- auxspace poly(D) . poly(Q)

Outline

- Motivation
- Terminology
- Results
- Final Remarks

Result Overview

Boolean Maintenance

View Maintenance

Core XPath	Time: polylog(D) . $2^{O(Q)}$ AuxSize: D . $2^{O(Q)}$
Core XPath	Time: depth(D) . log(width(D)) . $2^{O(Q)}$ AuxSize: D . $2^{O(Q)}$
/, //, [ ] and, or, not	Time: depth(D) . Q AuxSize: D . Q
nextsib, follow-sib [ ], and	Time: log(D) . poly(Q) AuxSize: D . Q^3
/, //, ns, fs [ ], and	Time: depth(D) . log(width(D)) . poly(Q) AuxSize: D . Q^3

Result Overview

Boolean Maintenance

View Maintenance

Core XPath	Time: polylog(D) . $2^{O(Q)}$ AuxSize: D . $2^{O(Q)}$
Core XPath	Time: depth(D) . log(width(D)) . $2^{O(Q)}$ AuxSize: D . $2^{O(Q)}$
/, //, [ ] and, or, not	Time: depth(D) . Q AuxSize: D . Q
nextsib, follow-sib [ ], and	Time: log(D) . poly(Q) AuxSize: D . Q^3
/, //, ns, fs [ ], and	Time: depth(D) . log(width(D)) . poly(Q) AuxSize: D . Q^3

Full Core XPath

Theorem (Balmin, Papakonstantinou, Vianu TODS 2005)

Incremental Maintenance for unranked tree automaton A on document D is in

- time $(\log(D))^2 . \text{poly}(A)$
- auxspace $D . \text{poly}(A)$

Theorem

A Core XPath query Q can be compiled into an unranked tree automaton of size $2^{O(Q)}$ in time $2^{O(Q)}$

(Standard techniques only seem to give $2^{O(Q^2)}$)

Full Core XPath

Corollary

Incremental Boolean Maintenance for Core XPath is possible in

- time $(\log(D))^2 \cdot 2^{O(Q)}$
- auxspace $D \cdot 2^{O(Q)}$

Similarly, with a different Balmin et al. [TODS 05] result:

Corollary

Incremental Boolean Maintenance for Core XPath is possible in

- time $\text{depth}(D) \cdot \log(\text{width}(D)) \cdot 2^{O(Q)}$
- auxspace $D \cdot 2^{O(Q)}$

Result Overview

Boolean Maintenance

View Maintenance

Core XPath	Time: polylog(D) . $2^{O(Q)}$ AuxSize: D . $2^{O(Q)}$
Core XPath	Time: depth(D) . log(width(D)) . $2^{O(Q)}$ AuxSize: D . $2^{O(Q)}$
/, //, [ ] and, or, not	Time: depth(D) . Q AuxSize: D . Q
nextsib, follow-sib [ ], and	Time: log(D) . poly(Q) AuxSize: D . Q^3
/, //, ns, fs [ ], and	Time: depth(D) . log(width(D)) . poly(Q) AuxSize: D . Q^3

Downward XPath

child (/), descendant (//), predicate [], and, or, not

Downward XPath

child (/), descendant (//), predicate [], and, or, not

I'm interested in:
papers about XML or XPath which are not written by Idiot

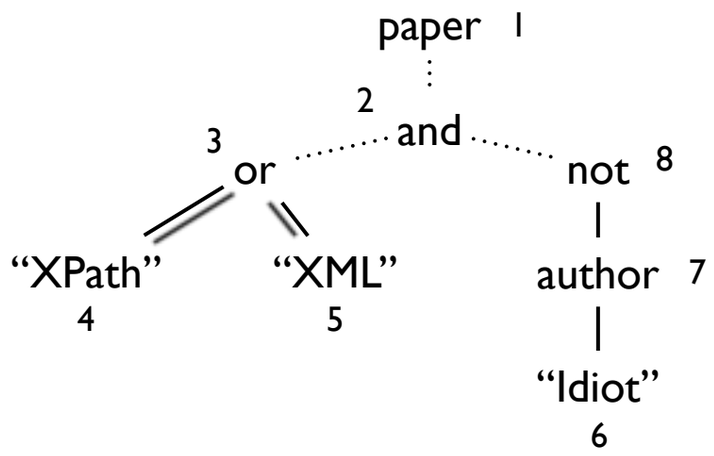
paper[(.//XPath or .//XML) and not ./author/Idiot]

Downward XPath

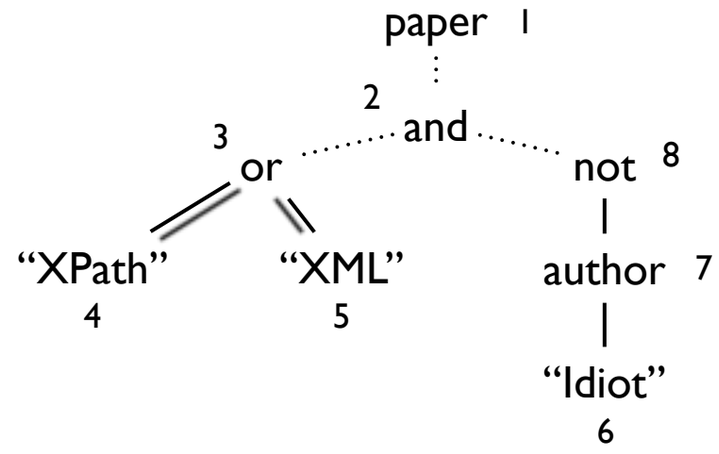
child (/), descendant (//), predicate [], and, or, not

I'm interested in:
papers about XML or XPath which are not written by Idiot

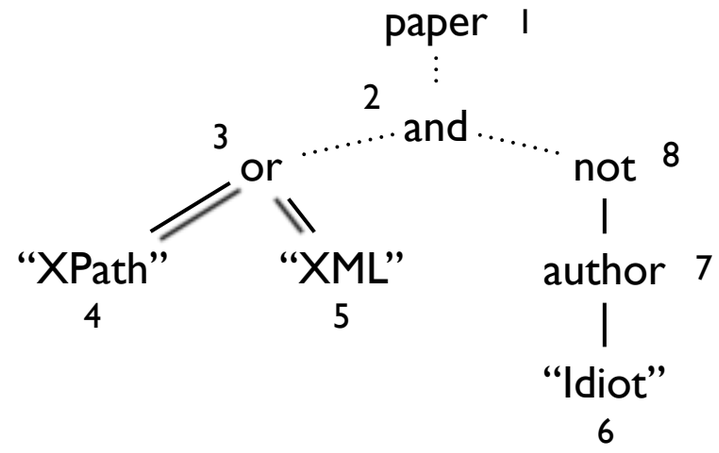
paper[(.//XPath or .//XML) and not ./author/Idiot]

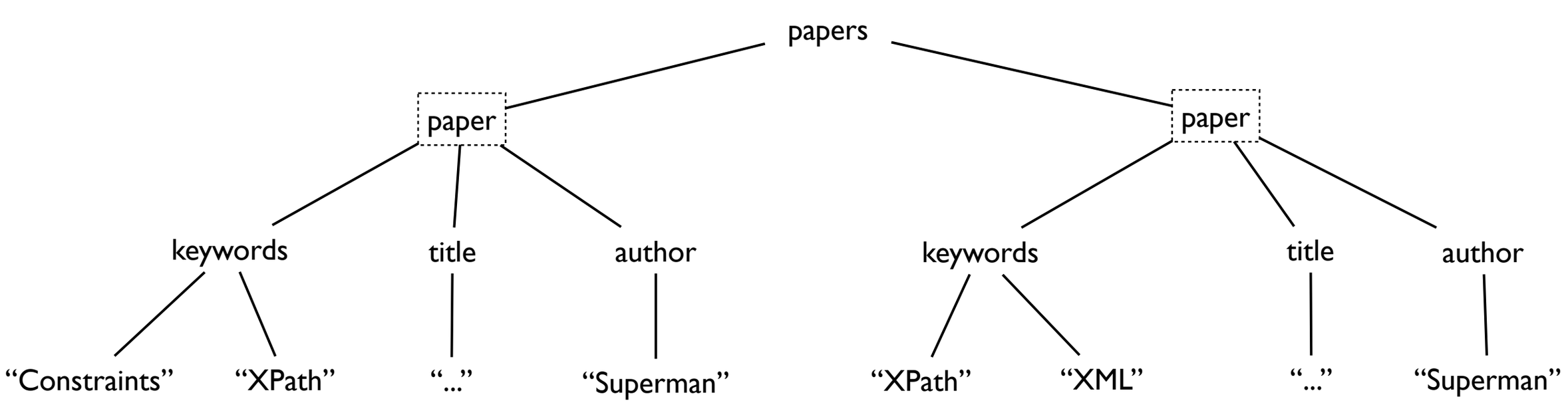


Downward XPath

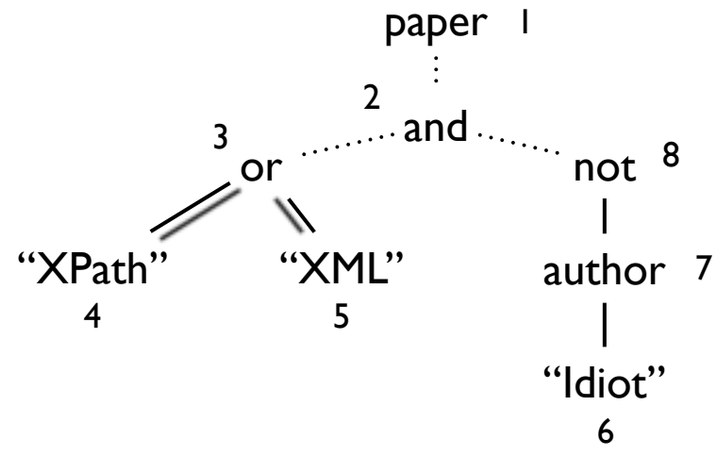


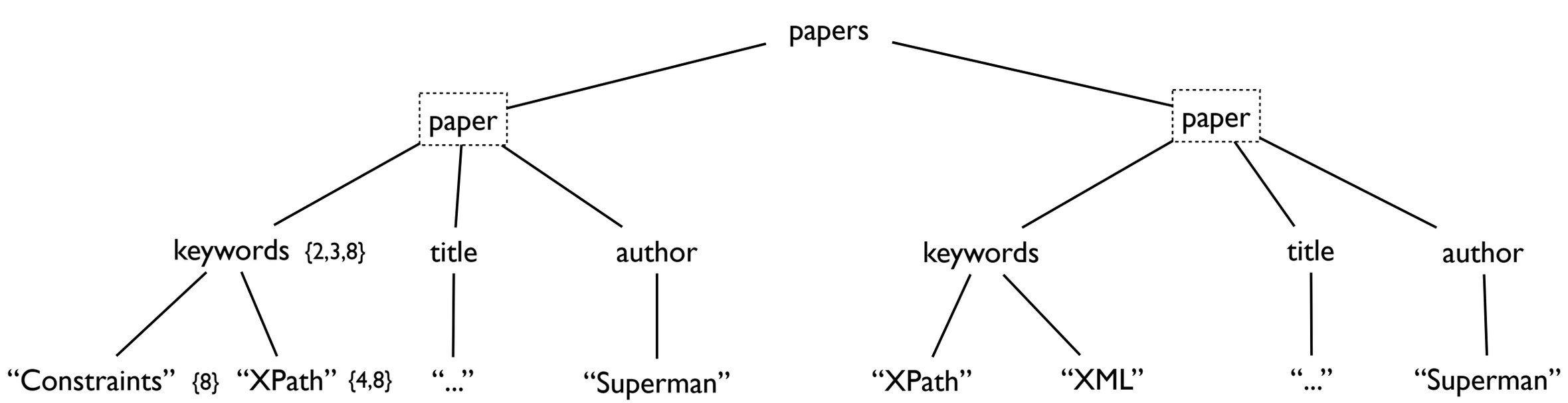
Downward XPath



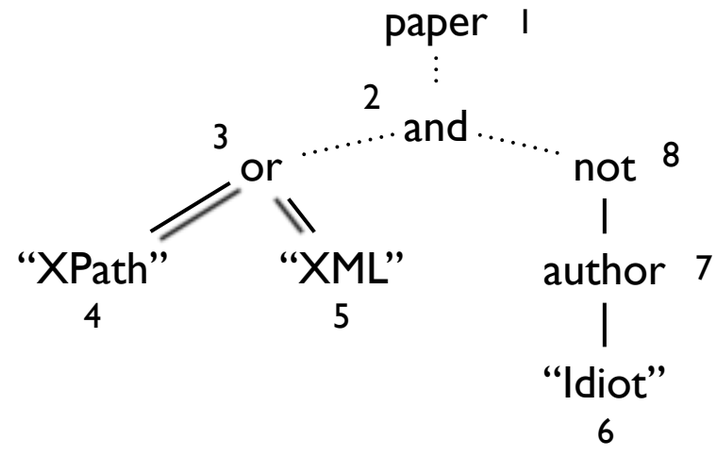


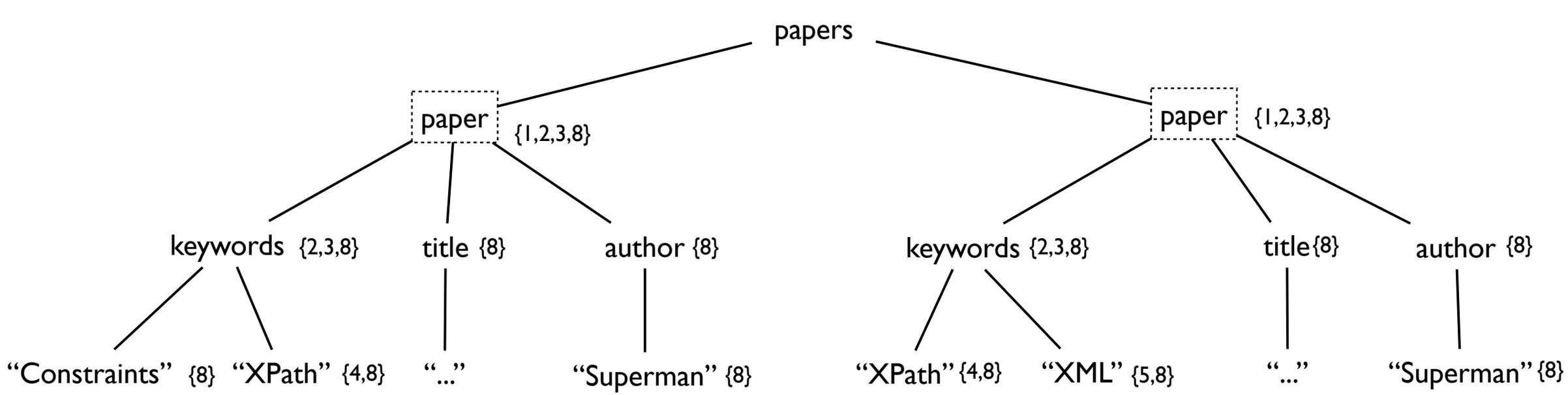
Downward XPath



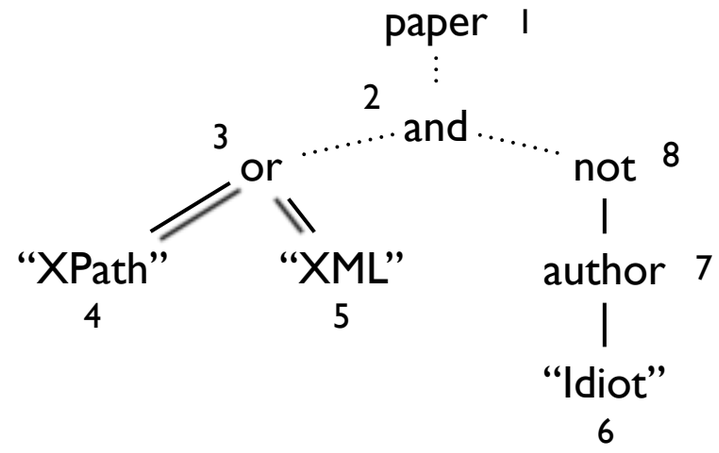


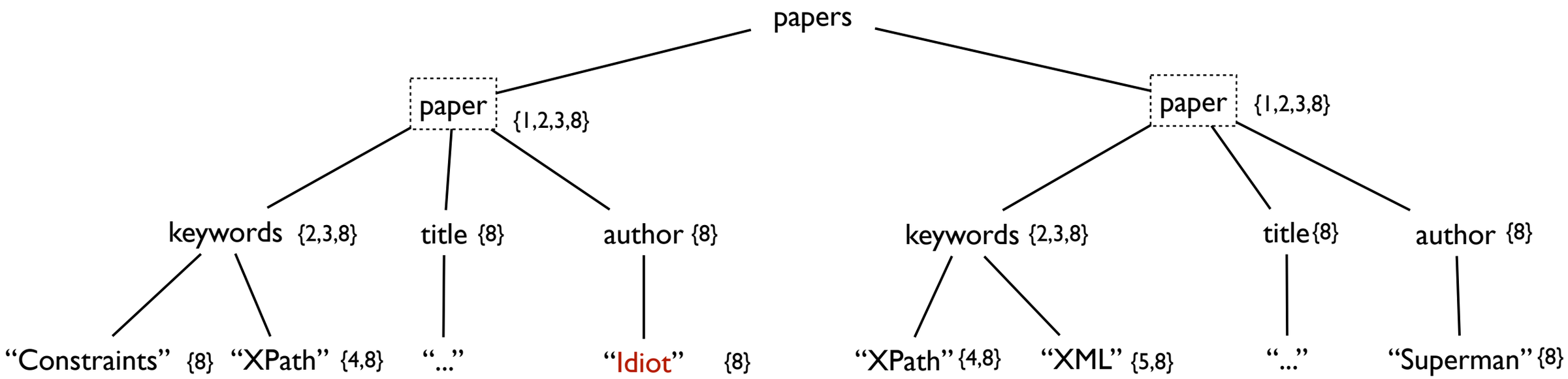
Downward XPath



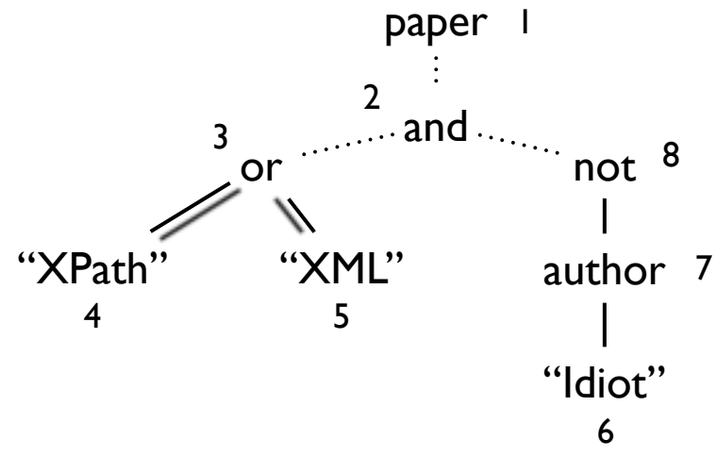


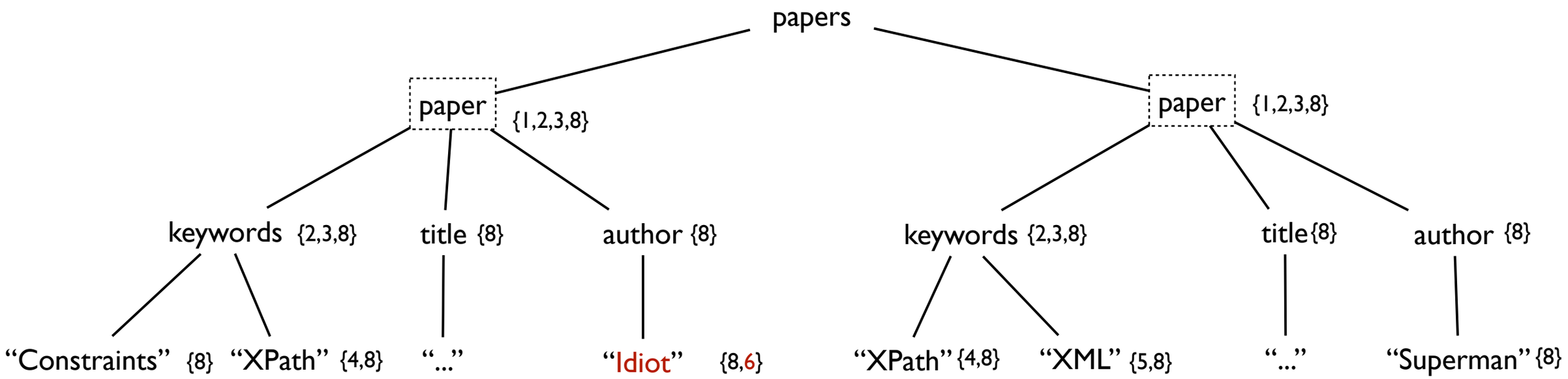
Downward XPath



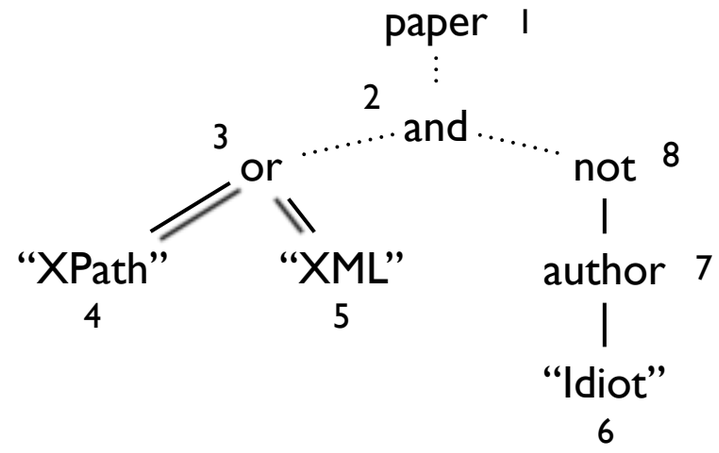


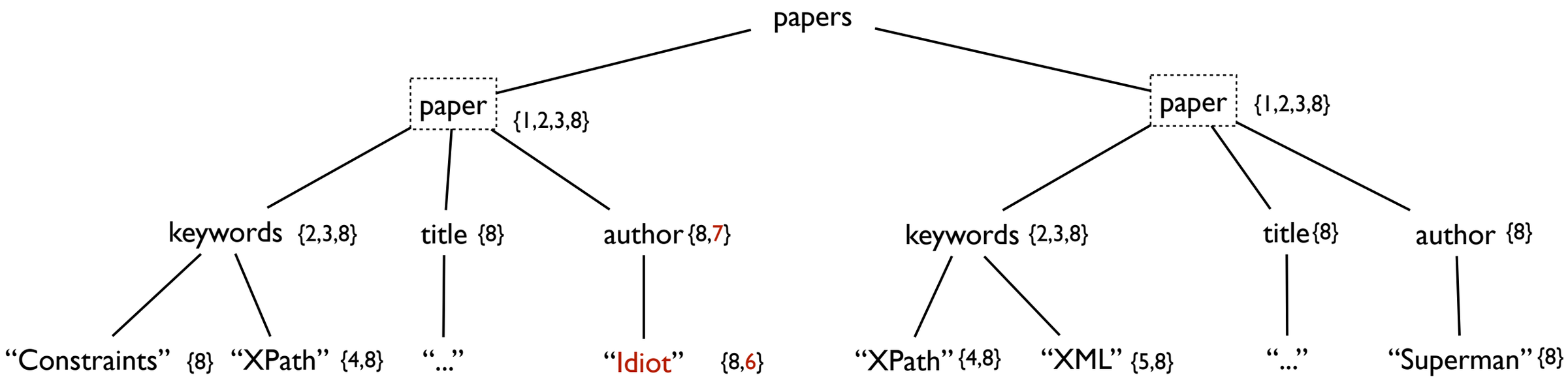
Downward XPath



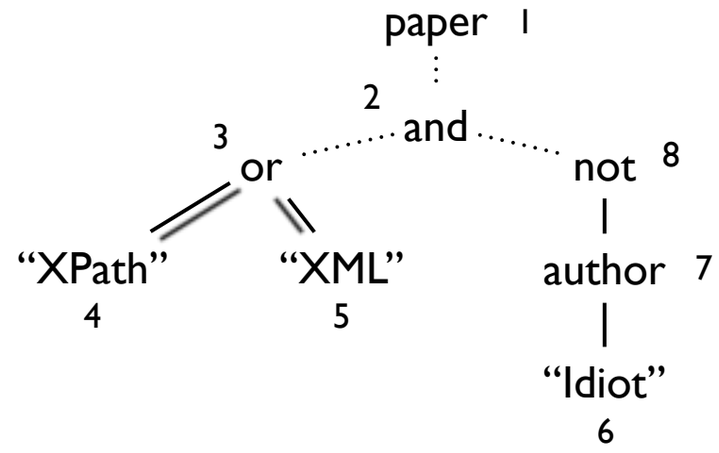


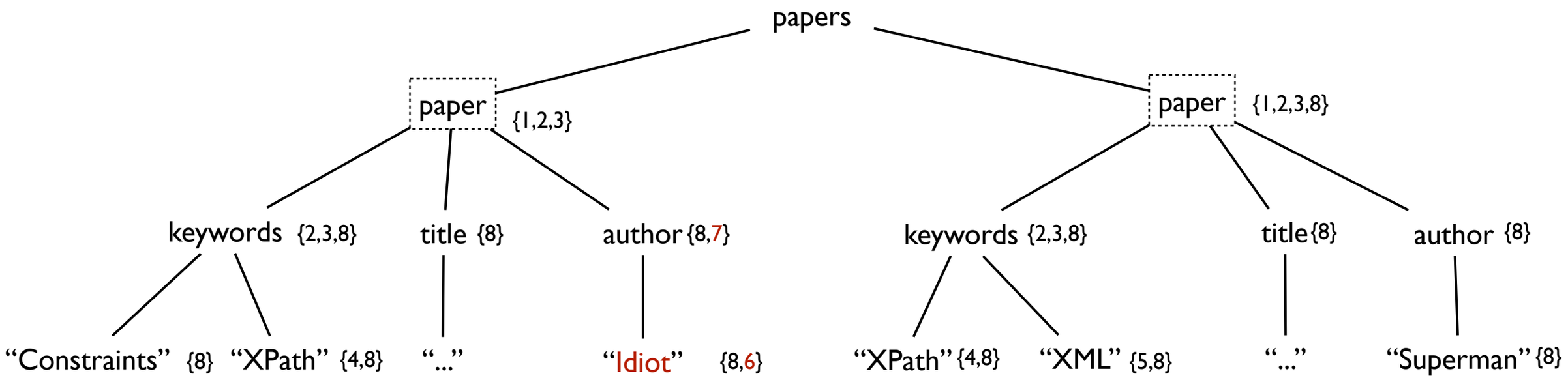
Downward XPath



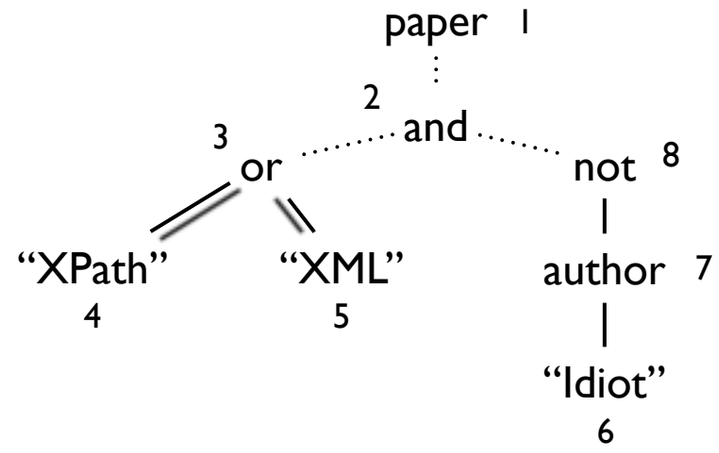


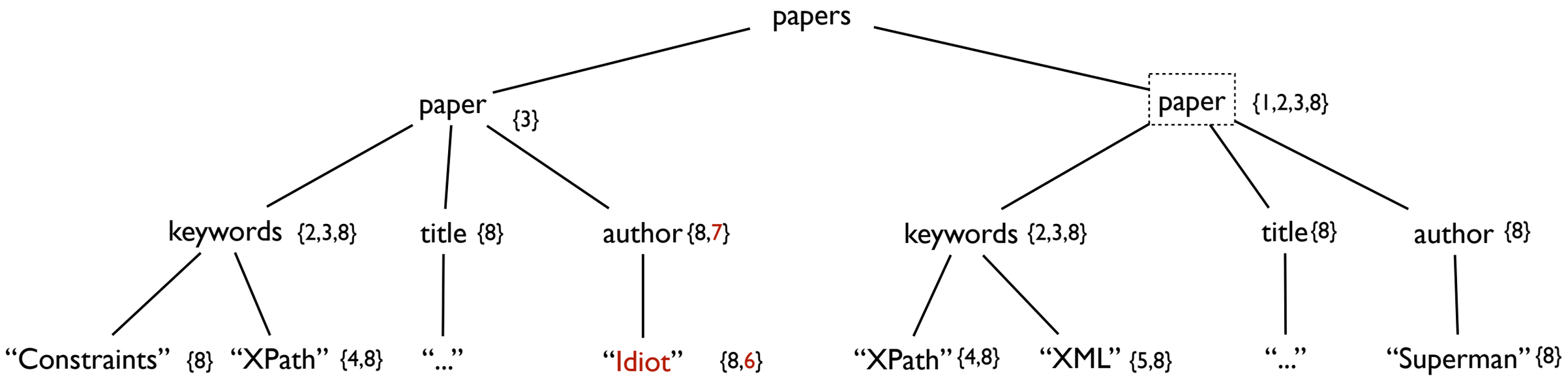
Downward XPath





Downward XPath





Downward XPath

Theorem

Incremental Boolean Maintenance for Downward XPath is possible in

- time depth(D) . Q
- auxspace D . Q

Downward XPath

Theorem

Incremental View Maintenance for Downward XPath is possible in

- time depth(D) . Q
- auxspace D . Q

(in restricted cases)

Downward XPath

Theorem

Incremental View Maintenance for Downward XPath is possible in
- time depth(D) . Q
- auxspace D . Q

(in restricted cases)

Restricted cases: root element must be selected

Example

paper[(.//XPath or .//XML) and not ./author/Idiot]

Result Overview

Boolean Maintenance

View Maintenance

Core XPath	Time: polylog(D) . $2^{O(Q)}$ AuxSize: D . $2^{O(Q)}$
Core XPath	Time: depth(D) . log(width(D)) . $2^{O(Q)}$ AuxSize: D . $2^{O(Q)}$
/, //, [ ] and, or, not	Time: depth(D) . Q AuxSize: D . Q
nextsib, follow-sib [ ], and	Time: log(D) . poly(Q) AuxSize: D . Q^3
/, //, ns, fs [ ], and	Time: depth(D) . log(width(D)) . poly(Q) AuxSize: D . Q^3

Forward XPath

child (/), descendant (//), nextsib, following-sib, predicate [], and

Forward XPath

child (/), descendant (//), nextsib, following-sib, predicate [], and

First: extremely shallow trees

Forward XPath

child (/), descendant (//), nextsib, following-sib, predicate [], and

First: extremely shallow trees

(well, strings)

Forward XPath

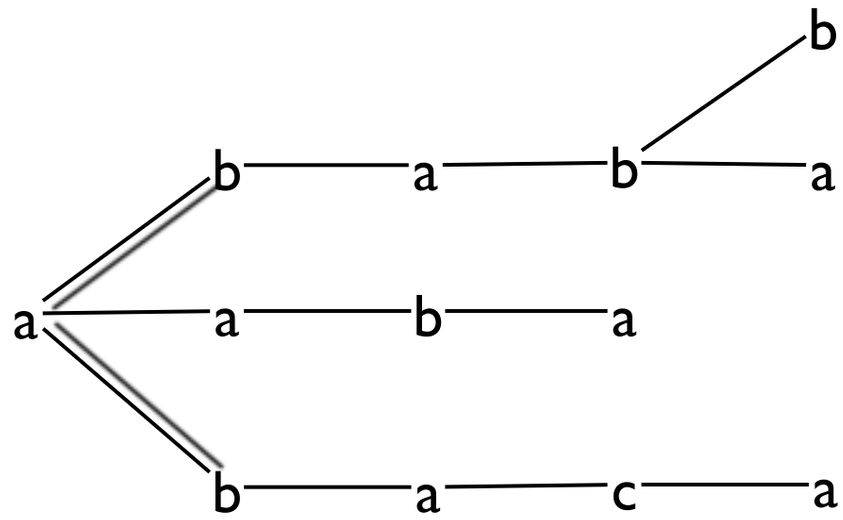
nextsib, following-sib, predicate [], and

First: extremely shallow trees

(well, strings)

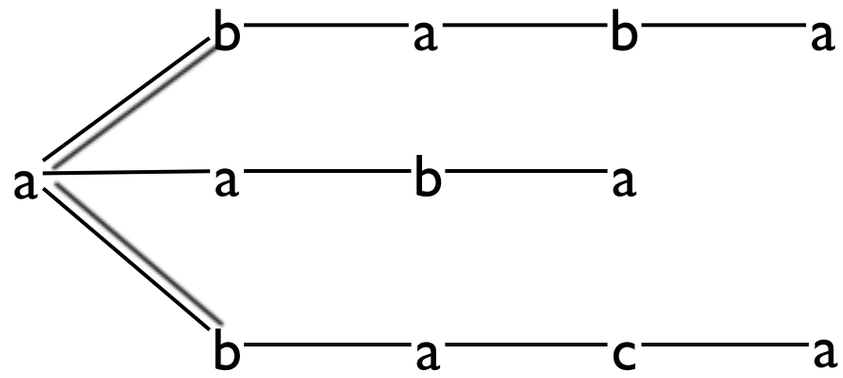
Forward XPath

nextsib, following-sib, predicate [], and



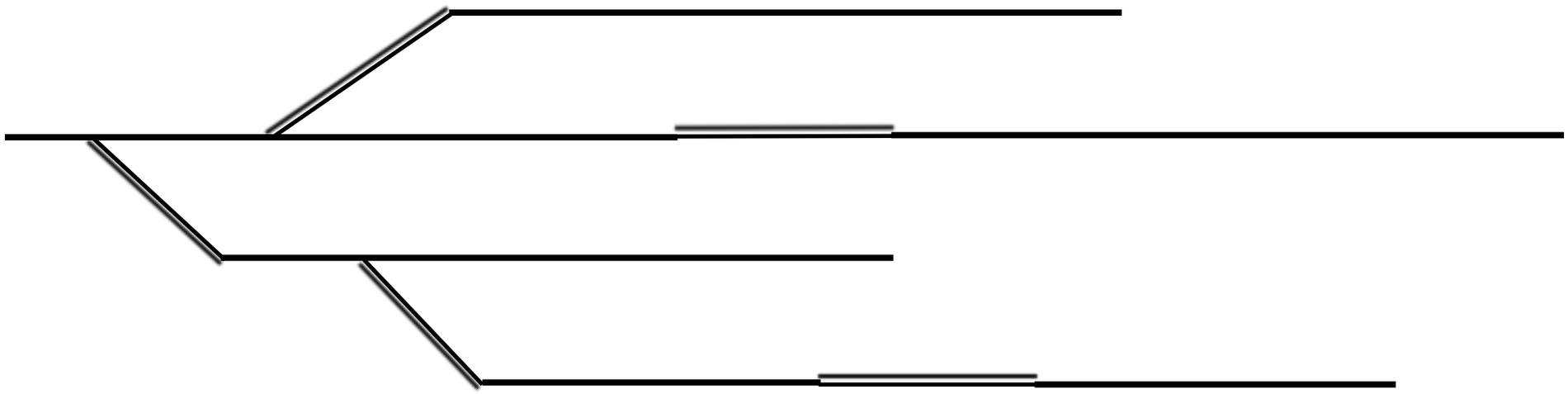
Forward XPath

nextsib, following-sib, predicate [], and



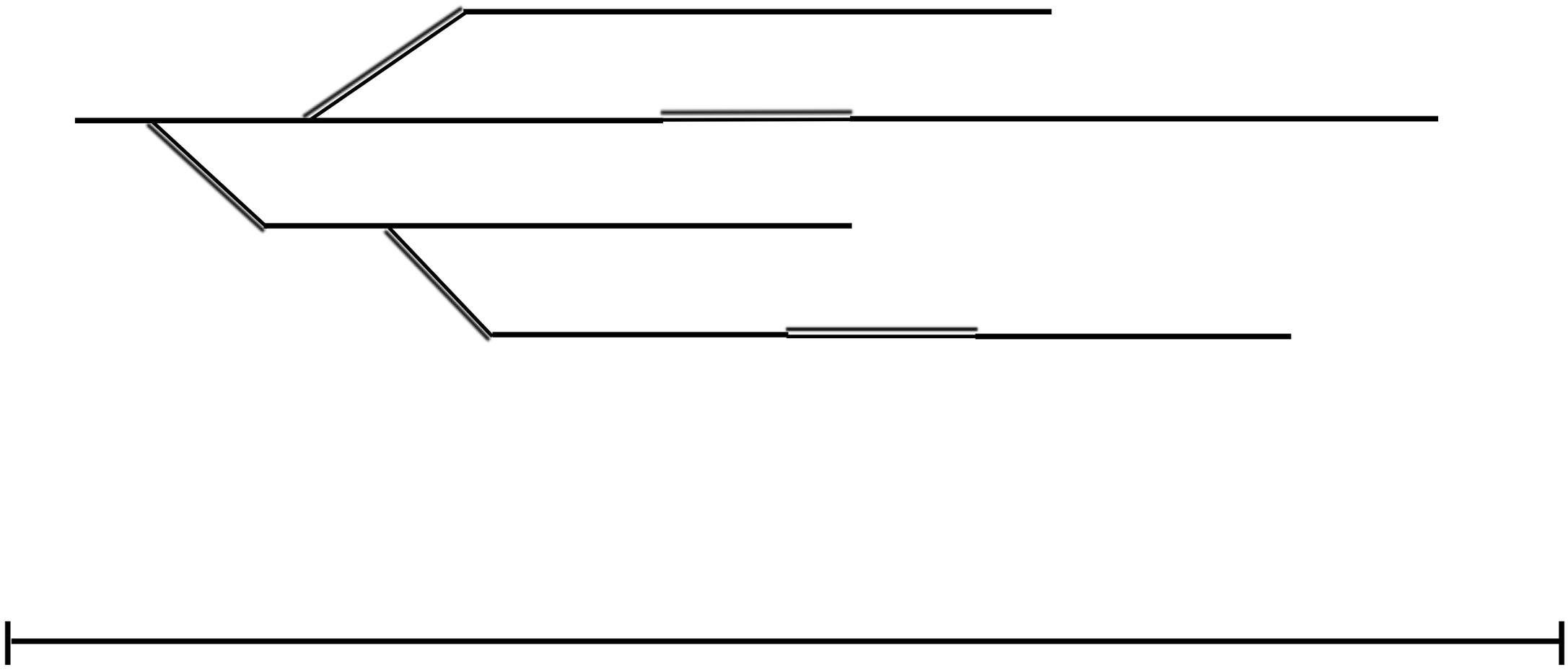
Forward XPath

nextsib, following-sib, predicate [], and



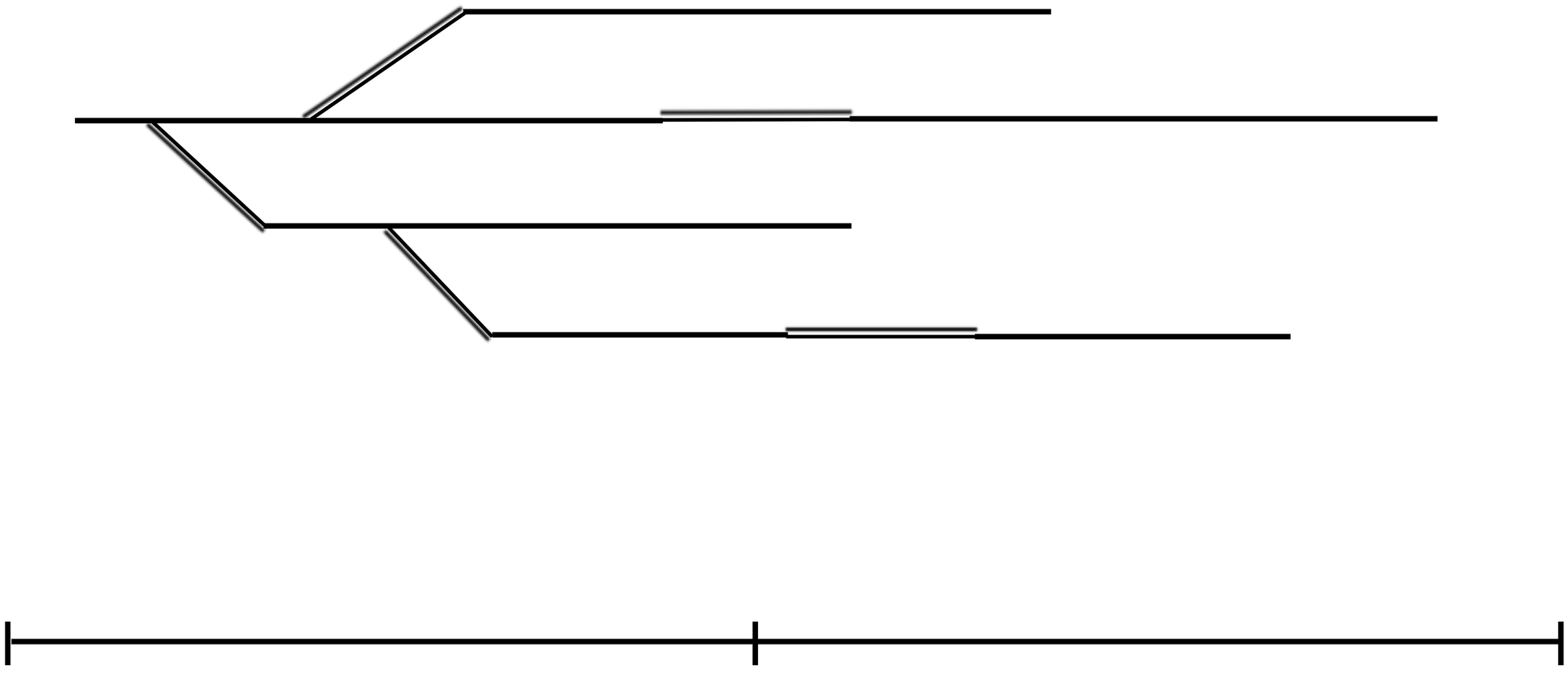
Forward XPath

nextsib, following-sib, predicate [], and



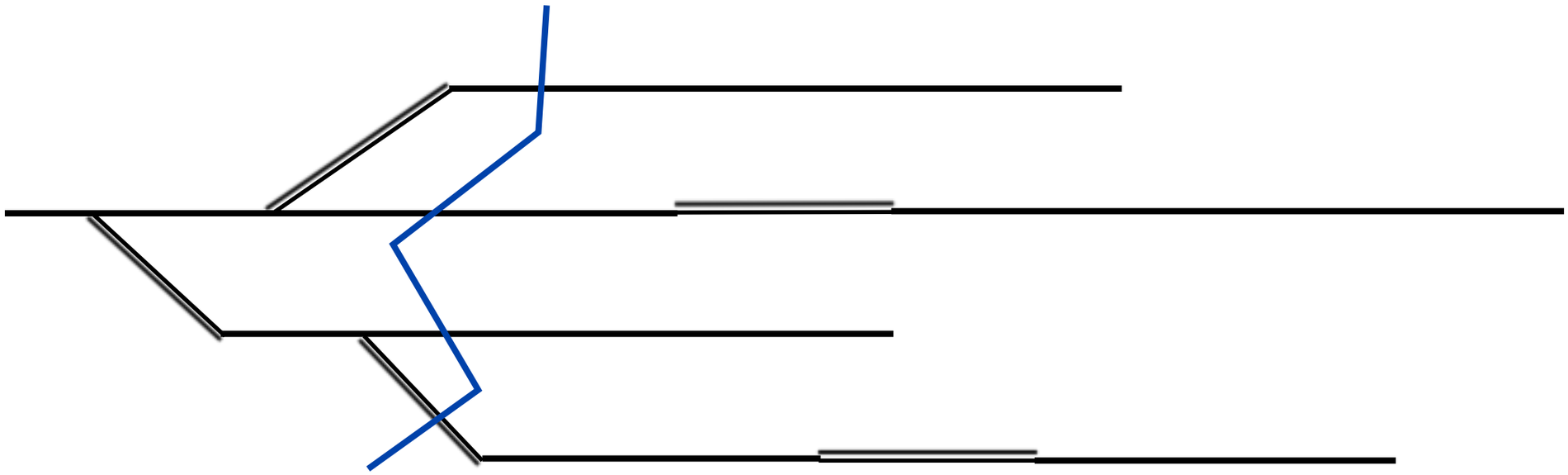
Forward XPath

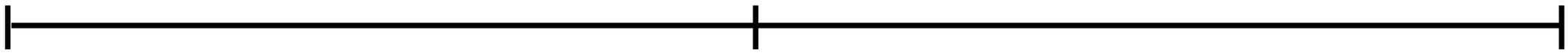
nextsib, following-sib, predicate [], and



Forward XPath

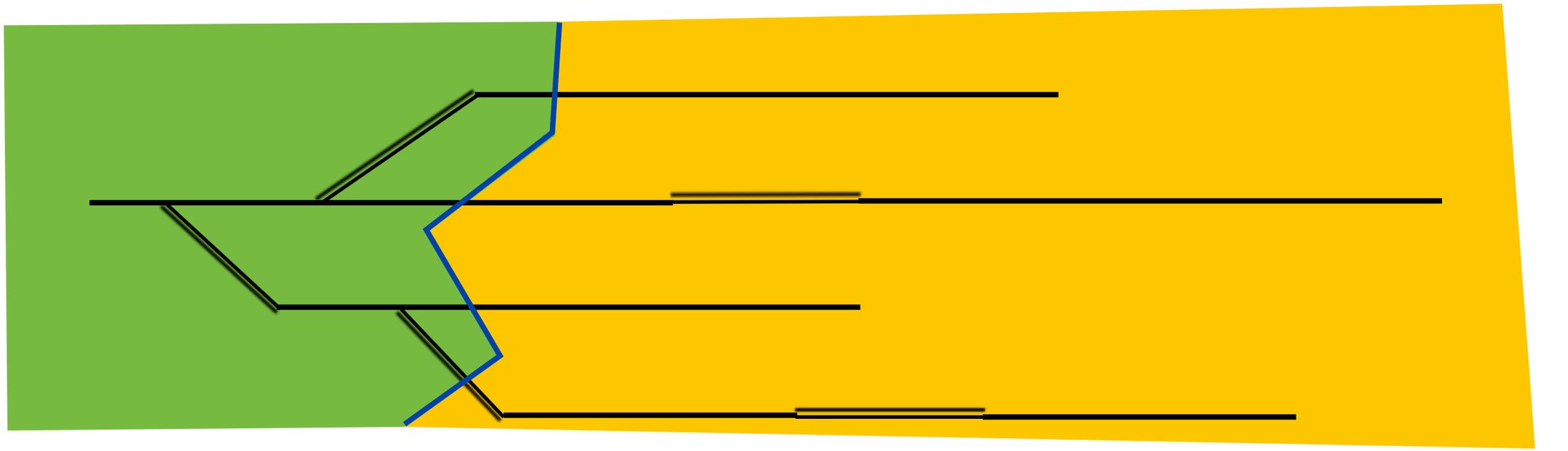
nextsib, following-sib, predicate [], and





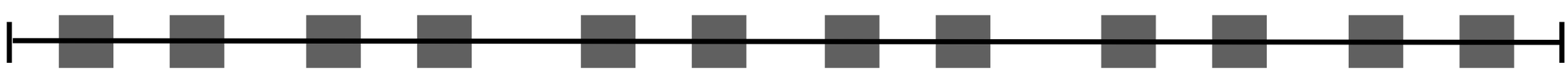
Forward XPath

nextsib, following-sib, predicate [], and



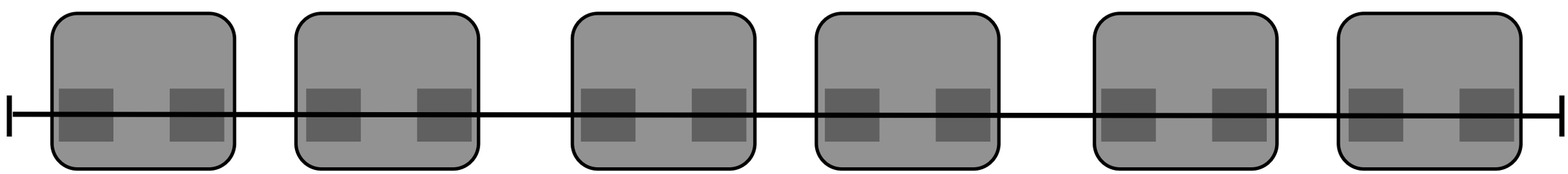
Forward XPath

nextsib, following-sib, predicate [], and



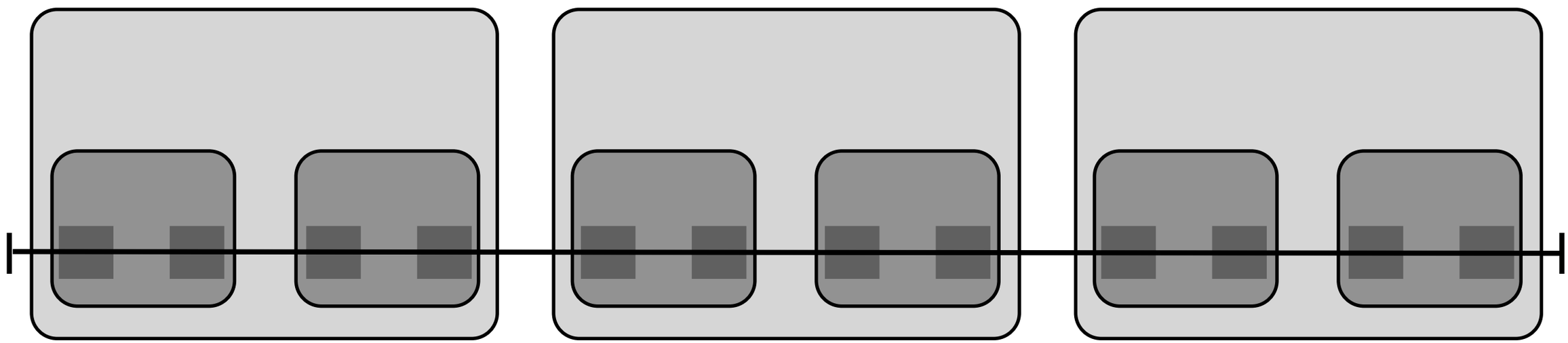
Forward XPath

nextsib, following-sib, predicate [], and



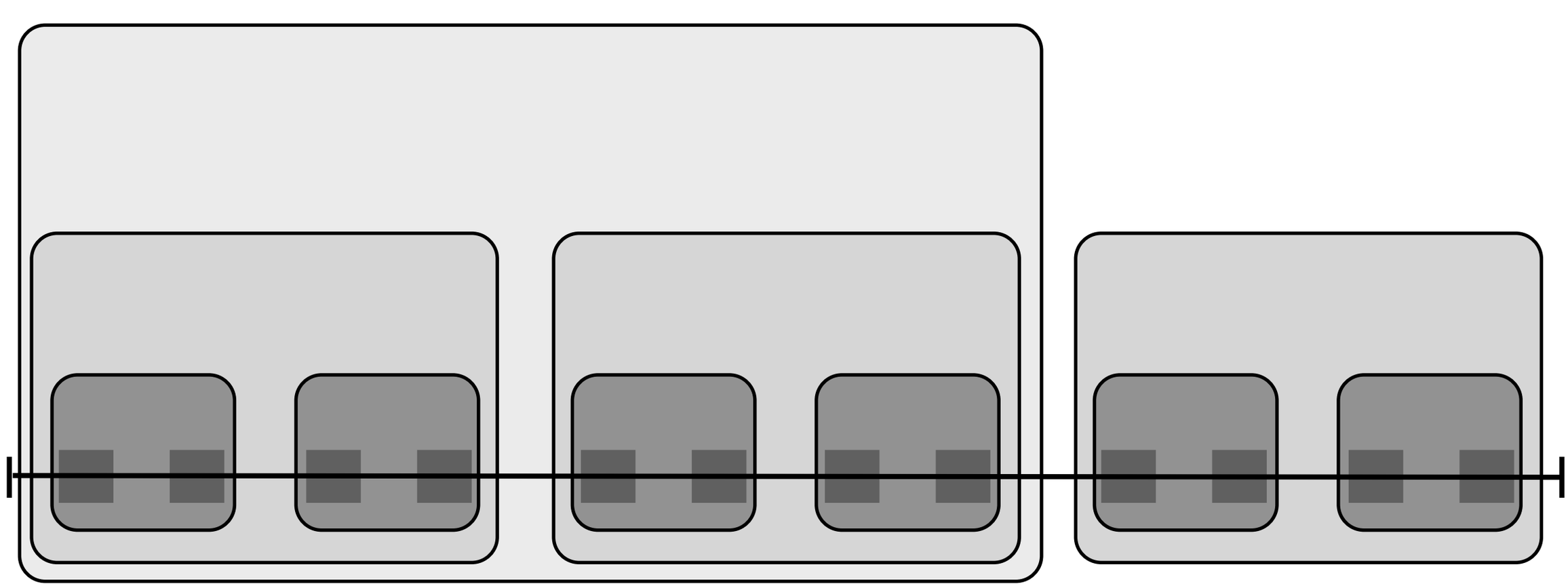
Forward XPath

nextsib, following-sib, predicate [], and



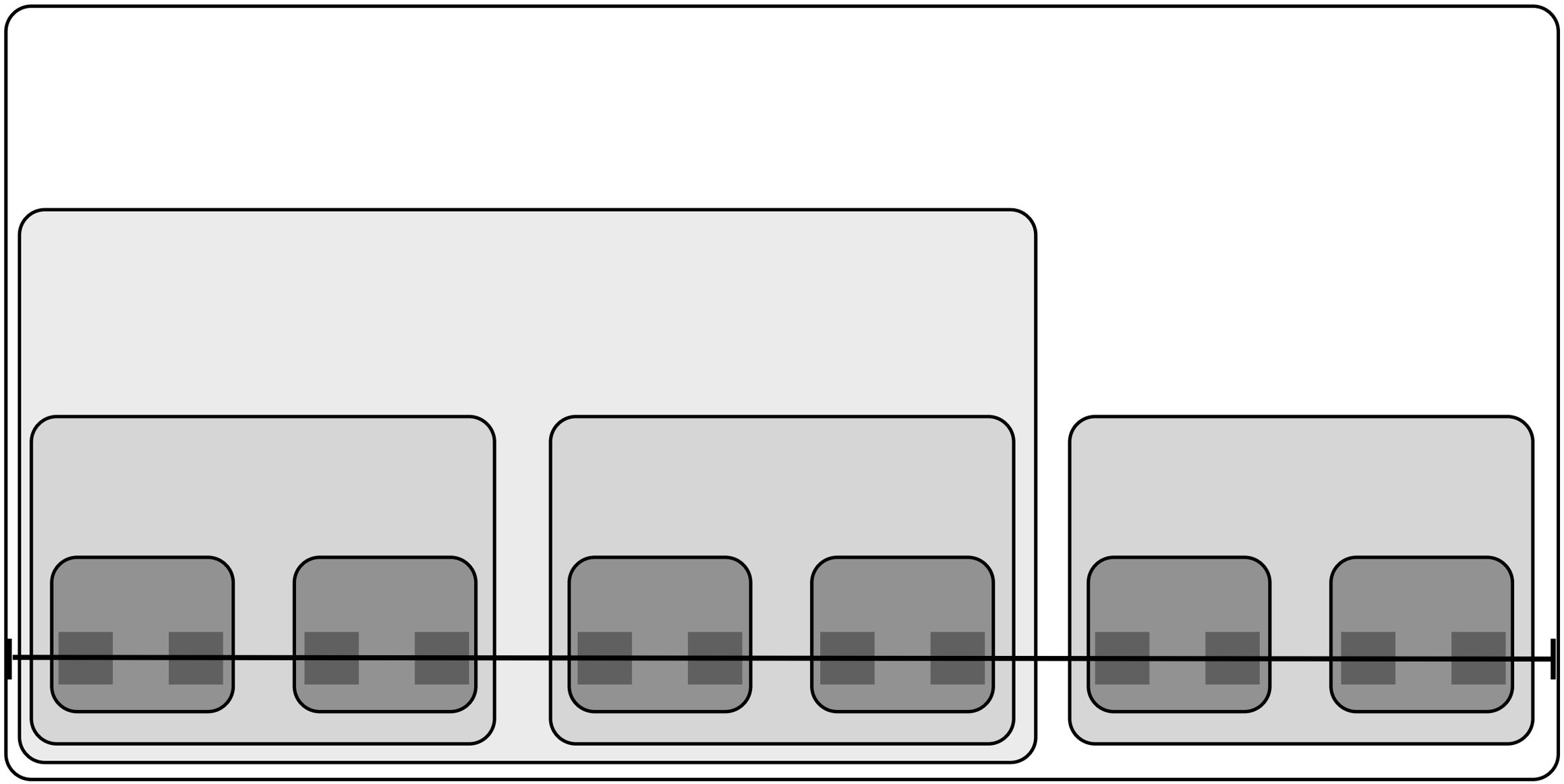
Forward XPath

nextsib, following-sib, predicate [], and



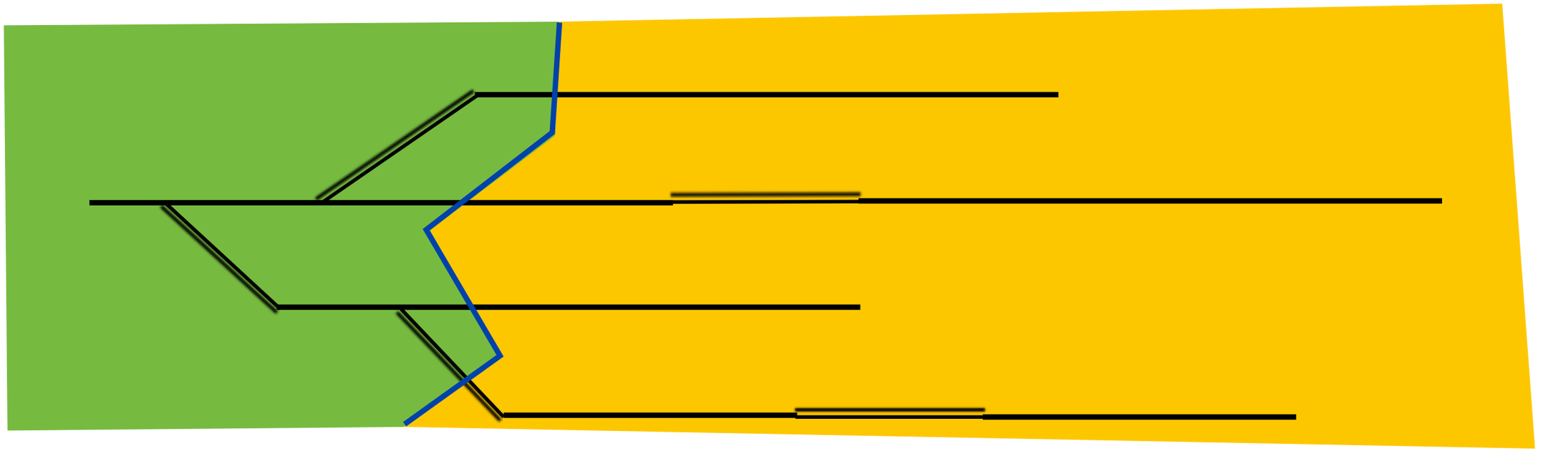
Forward XPath

nextsib, following-sib, predicate [], and



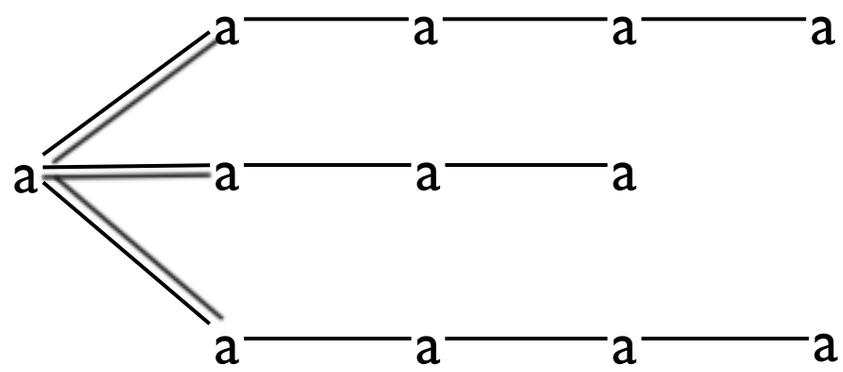
Forward XPath

nextsib, following-sib, predicate [], and



Forward XPath

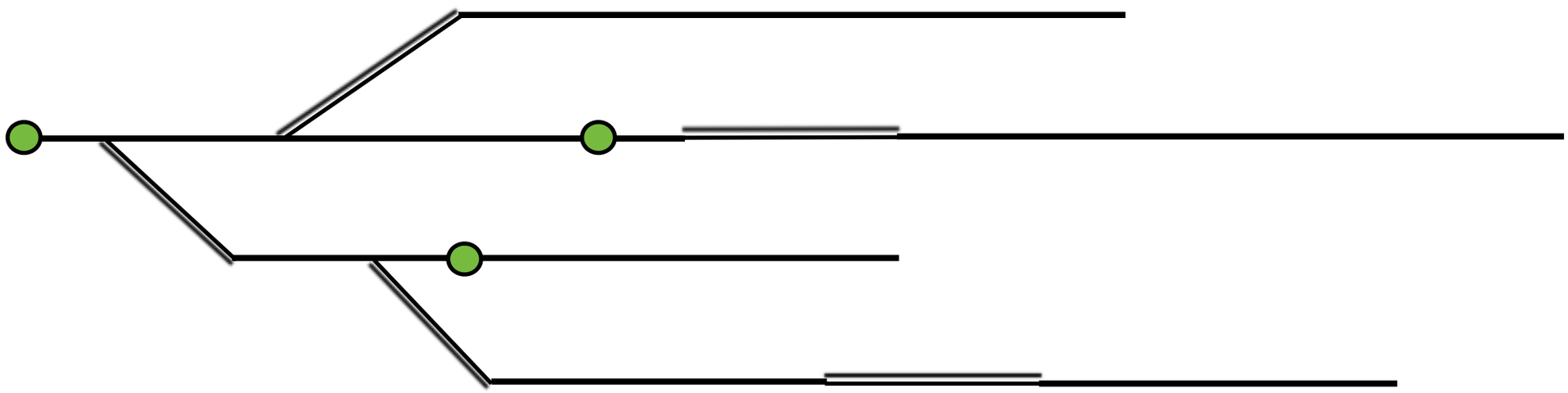
nextsib, following-sib, predicate [], and



a a a a a a a a | a a a a a a a a

Forward XPath

nextsib, following-sib, predicate [], and



Forward XPath

nextsib, following-sib, predicate [], and

Theorem

Incremental Boolean Maintenance for Forward XPath is possible in

- time $\log(D) \cdot \text{poly}(Q)$
- auxspace $D \cdot Q^3$

on strings

Forward XPath

child (/), descendant (//), nextsib, following-sib, predicate [], and

Combining this idea with the depth(D) algorithm:

Theorem

Incremental Boolean Maintenance for Forward XPath is possible in

- time depth(D) log(width(D)) . poly(Q)
- auxspace D . Q^3

on trees

Outline

- Motivation
- Terminology
- Results
- Final Remarks

Final Remarks

- Incremental XPath Evaluation is interesting
- Boolean version is already non-trivial

Final Remarks

- We like depth(D) maintenance for downward XPath
- Our Algorithm for Forward XPath is quite involved...
- But without NextSibling, it's much simpler

Outlook

The big questions:

For which XPath fragments is Boolean Maintenance possible in

- time polylog(D) . poly(Q)
- auxspace poly(D) . poly(Q)

Outlook

The big questions:

For which XPath fragments is View Maintenance possible in

- time polylog(D) . poly(Q)
- auxspace poly(D) . poly(Q)

Result Overview

Boolean Maintenance

View Maintenance

Core XPath	Time: polylog(D) . $2^{O(Q)}$ AuxSize: D . $2^{O(Q)}$
Core XPath	Time: depth(D) . log(width(D)) . $2^{O(Q)}$ AuxSize: D . $2^{O(Q)}$
/, //, [ ] and, or, not	Time: depth(D) . Q AuxSize: D . Q
nextsib, follow-sib [ ], and	Time: log(D) . poly(Q) AuxSize: D . Q^3
/, //, ns, fs [ ], and	Time: depth(D) . log(width(D)) . poly(Q) AuxSize: D . Q^3