XPath Whole Query Optimization

Sebastian Maneth[1,2] Kim Nguyễn[3,formerly 2]

VLDB 2010, Singapore
Sep. 14-17 2010

1 University of New South Wales, Sydney, Australia
2 National ICT Australia, Sydney, Australia
3 LRI, University Paris-Sud 11, Orsay, France

Fast query evaluation over indexed XML documents

Fast query evaluation over indexed XML documents

Tree automata

Fast query evaluation over indexed XML documents

- Staircase join (used in MonetDB)
- Twig joins
- ...

Tree automata

Fast query evaluation over indexed XML documents

- Staircase join (used in MonetDB)
- Twig joins
- ...

Tree automata

- Streaming XML queries
- XML typechecking
- XML pattern-matching
- Theoretical work (expressivity or complexity results)

"Core XPath can be evaluated in time $O(|D| \cdot |Q|)$"

Gottlob, Koch, Pichler, *Efficient algorithms for processing XPath queries*, VLDB'02

"Core XPath can be evaluated in time $O(|D| \cdot |Q|)$"

Gottlob, Koch, Pichler, *Efficient algorithms for processing XPath queries*, VLDB'02

If we can translate an XPath query into a tree automaton, the algorithm for evaluating tree automata is $o(|D| \cdot |Q|)$

"Core XPath can be evaluated in time $O(|D| \cdot |Q|)$"

Gottlob, Koch, Pichler, *Efficient algorithms for processing XPath queries*, VLDB'02

If we can translate an XPath query into a tree automaton, the algorithm for evaluating tree automata is $o(|D| \cdot |Q|)$

$|D|$: do we need to visit the whole document ?

"Core XPath can be evaluated in time $O(|D| \cdot |Q|)$"

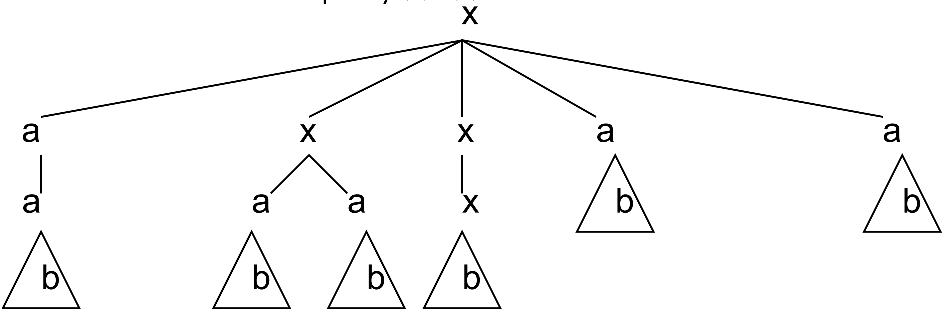
Gottlob, Koch, Pichler, *Efficient algorithms for processing XPath queries*, VLDB'02

If we can translate an XPath query into a tree automaton, the algorithm for evaluating tree automata is $o(|D| \cdot |Q|)$

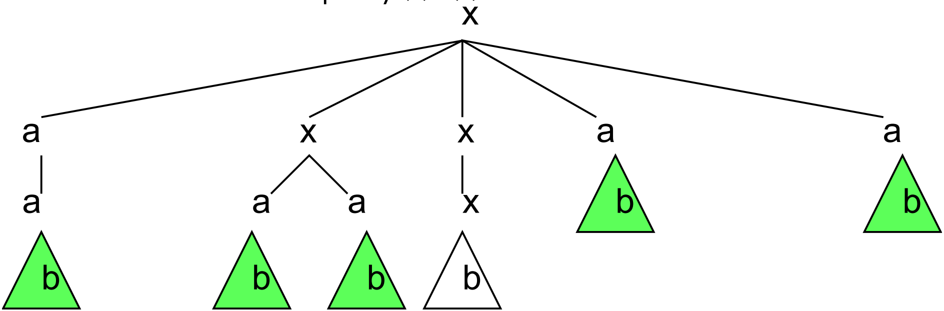
$|D|$: do we need to visit the whole document ?

$|Q|$: for each visited node, how much work is needed ?

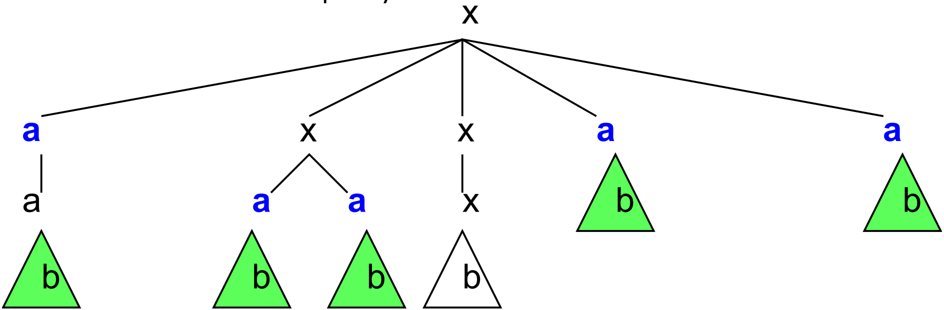
Consider the XPath query `//a//b` and the document :



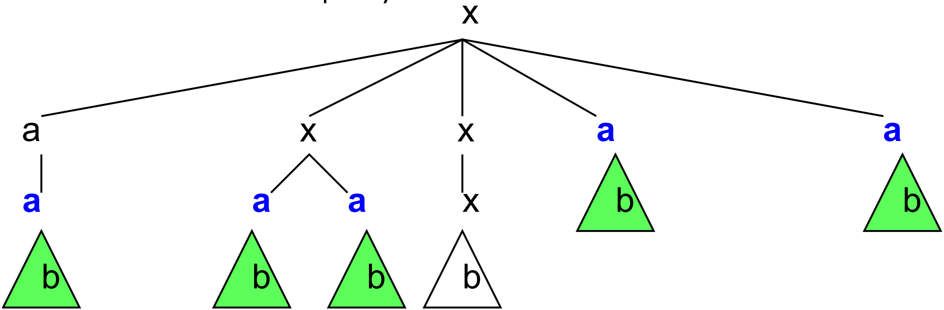
Consider the XPath query `//a//b` and the document :



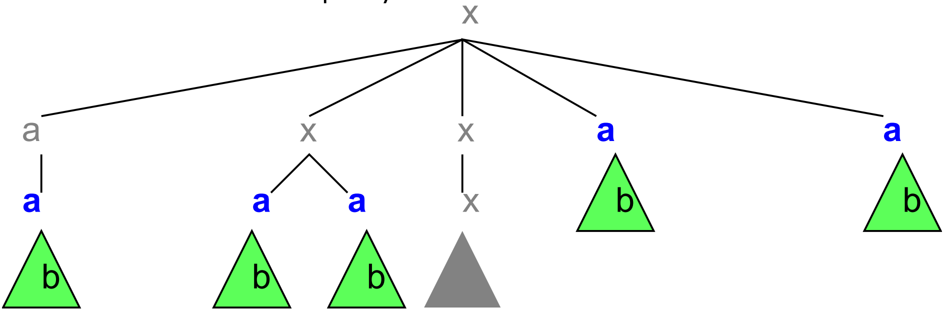
Consider the XPath query `//a//b` and the document :



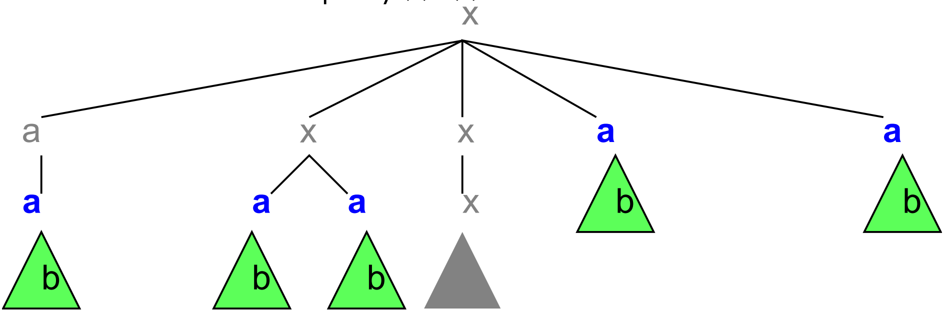
Consider the XPath query `//a//b` and the document :



Consider the XPath query `//a//b` and the document :

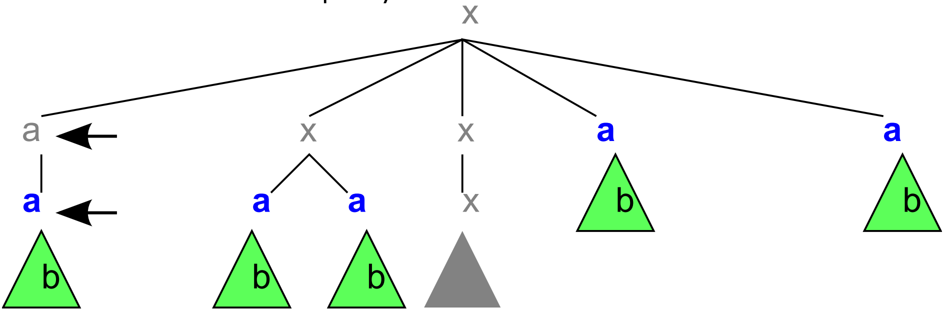


Consider the XPath query `//a//b` and the document :



- Does the engine need to know about **these nodes** ?

Consider the XPath query `//a//b` and the document :



- Does the engine need to know about **these nodes** ?
- How does it choose between to equally good nodes ?

Definition (Binary tree)

Binary trees over Σ : $T(\Sigma)$ smallest set s.t. :

- $\# \in \Sigma$ (leaf symbol)
- $t_1, t_2 \in T(\Sigma) \Rightarrow \forall l \in \Sigma, l(t_1, t_2) \in T(\Sigma)$

Definition (Binary tree)

Binary trees over Σ : $T(\Sigma)$ smallest set s.t. :

- $\# \in \Sigma$ (leaf symbol)
- $t_1, t_2 \in T(\Sigma) \Rightarrow \forall l \in \Sigma, l(t_1, t_2) \in T(\Sigma)$

Definition (Set of nodes of a tree t)

Smallest set $\mathrm{Dom}(t) \subseteq \{1, 2\}^*$ such that :

- $\epsilon \in \mathrm{Dom}(t)$
- $t(\epsilon) = l$ if $t \equiv l(t_1, t_2)$
- $t(\epsilon) = \#$ if $t \equiv \#$
- $t(i \cdot \pi) = t_i(\pi)$, if $t \equiv l(t_1, t_2)$

Definition (Selecting Tree Automaton (STA))

$\mathcal{A} = (\Sigma, Q, T, B, S, \delta)$

- Σ input symbols, Q set of states
- T set of top states, B set of bottom-states
- $S \subseteq Q \times \Sigma$ set of selecting configurations
- δ set of transitions : $q, L \rightarrow q_1, q_2$ (where $L \subseteq \Sigma$)

Definition (Run R of an STA $\mathcal{A}$ over t)

A run is a mapping from $\text{Dom}(t)$ to Q generated by δ

Definition (Selected nodes)

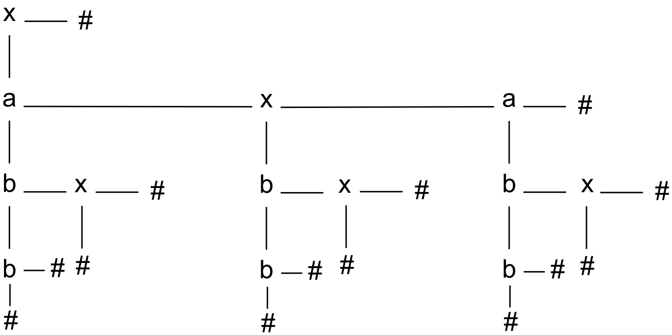
For a given run R, the set of selected nodes is

$$\mathcal{A}^R(t) = \{\pi \in \text{Dom}(t) \mid (R(\pi), t(\pi)) \in S\}$$

Example (//a//b)

$$\mathcal{A}_{//\mathtt{a}//\mathtt{b}} = (\underbrace{\{\mathtt{a},\mathtt{b},\mathtt{x}\}}_{\Sigma}, \underbrace{\{q_0,q_1\}}_{Q}, \underbrace{\{q_0\}}_{T}, \underbrace{\{q_0,q_1\}}_{B}, \underbrace{\{(q_1,\mathtt{b})\}}_{S}, \delta)$$

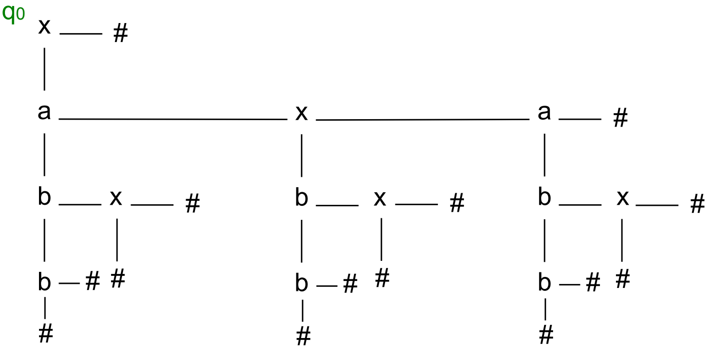
$$\delta = \begin{array}{llll} q_0,\{\mathtt{a}\} & \rightarrow (q_1,q_0) & q_1,\{\mathtt{b}\} & \Rightarrow (q_1,q_1) \\ q_0,\Sigma \setminus \{\mathtt{a}\} & \rightarrow (q_0,q_0) & q_1,\Sigma \setminus \{\mathtt{b}\} & \rightarrow (q_1,q_1) \end{array}$$



Example (//a//b)

$$\mathcal{A}_{//a//b} = (\underbrace{\{a, b, x\}}_{\Sigma}, \underbrace{\{q_0, q_1\}}_{Q}, \underbrace{\{q_0\}}_{T}, \underbrace{\{q_0, q_1\}}_{B}, \underbrace{\{(q_1, b)\}}_{S}, \delta)$$

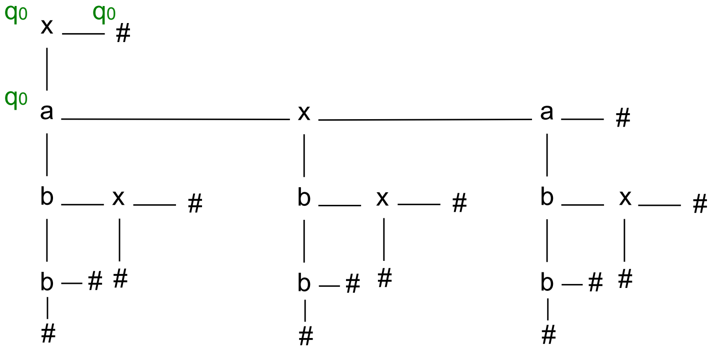
$$\delta = \begin{array}{llll} q_0, \{a\} & \rightarrow (q_1, q_0) & q_1, \{b\} & \Rightarrow (q_1, q_1) \\ q_0, \Sigma \setminus \{a\} & \rightarrow (q_0, q_0) & q_1, \Sigma \setminus \{b\} & \rightarrow (q_1, q_1) \end{array}$$



Example (//a//b)

$$\mathcal{A}_{//\mathtt{a}//\mathtt{b}} = (\underset{\Sigma}{\{\mathtt{a},\mathtt{b},\mathtt{x}\}}, \underset{Q}{\{q_0,q_1\}}, \underset{T}{\{q_0\}}, \underset{B}{\{q_0,q_1\}}, \underset{S}{\{(q_1,\mathtt{b})\}}, \delta)$$

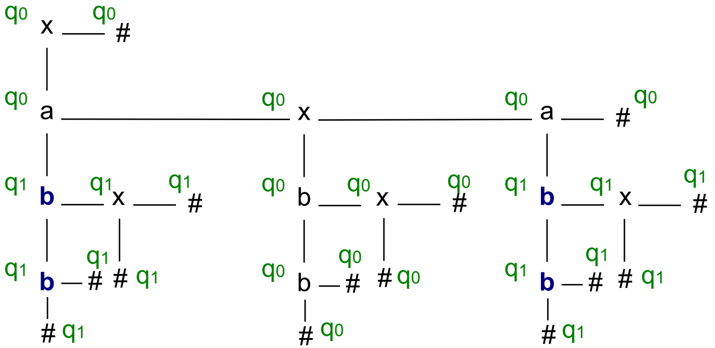
$$\delta = \begin{array}{llll} q_0,\{\mathtt{a}\} & \rightarrow (q_1,q_0) & q_1,\{\mathtt{b}\} & \Rightarrow (q_1,q_1) \\ q_0,\Sigma\setminus\{\mathtt{a}\} & \rightarrow (q_0,q_0) & q_1,\Sigma\setminus\{\mathtt{b}\} & \rightarrow (q_1,q_1) \end{array}$$



Example (//a//b)

$$\mathcal{A}_{//a//b} = (\underbrace{\{a, b, x\}}_{\Sigma}, \underbrace{\{q_0, q_1\}}_{Q}, \underbrace{\{q_0\}}_{T}, \underbrace{\{q_0, q_1\}}_{B}, \underbrace{\{(q_1, b)\}}_{S}, \delta)$$

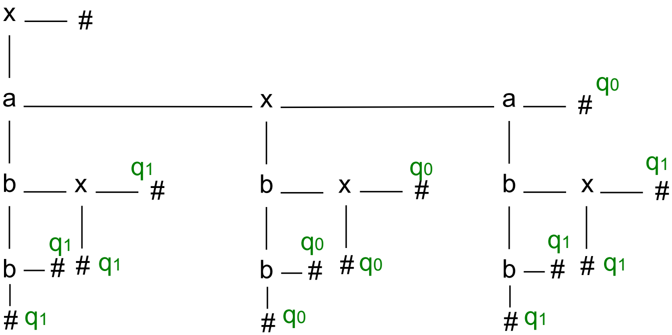
$$\delta = \begin{array}{llll} q_0, \{a\} & \to (q_1, q_0) & q_1, \{b\} & \Rightarrow (q_1, q_1) \\ q_0, \Sigma \setminus \{a\} & \to (q_0, q_0) & q_1, \Sigma \setminus \{b\} & \to (q_1, q_1) \end{array}$$



Example (//a//b)

$$\mathcal{A}_{//\mathtt{a}//\mathtt{b}} = (\underbrace{\{\mathtt{a},\mathtt{b},\mathtt{x}\}}_{\Sigma}, \underbrace{\{q_0,q_1\}}_{Q}, \underbrace{\{q_0\}}_{T}, \underbrace{\{q_0,q_1\}}_{B}, \underbrace{\{(q_1,\mathtt{b})\}}_{S}, \delta)$$

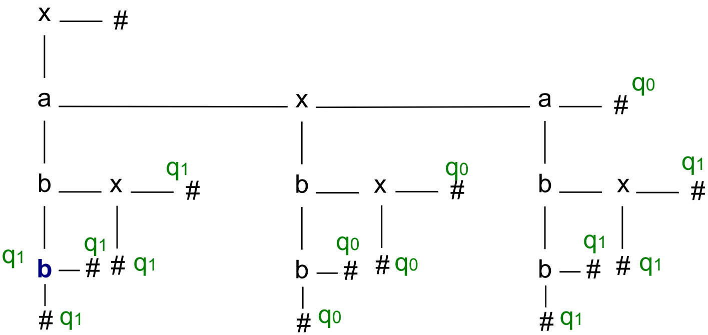
$$\delta = \begin{array}{llll} q_0, \{\mathtt{a}\} & \rightarrow (q_1, q_0) & q_1, \{\mathtt{b}\} & \Rightarrow (q_1, q_1) \\ q_0, \Sigma \setminus \{\mathtt{a}\} & \rightarrow (q_0, q_0) & q_1, \Sigma \setminus \{\mathtt{b}\} & \rightarrow (q_1, q_1) \end{array}$$



Example (//a//b)

$$\mathcal{A}_{//a//b} = (\underbrace{\{a,b,x\}}_{\Sigma}, \underbrace{\{q_0,q_1\}}_{Q}, \underbrace{\{q_0\}}_{T}, \underbrace{\{q_0,q_1\}}_{B}, \underbrace{\{(q_1,b)\}}_{S}, \delta)$$

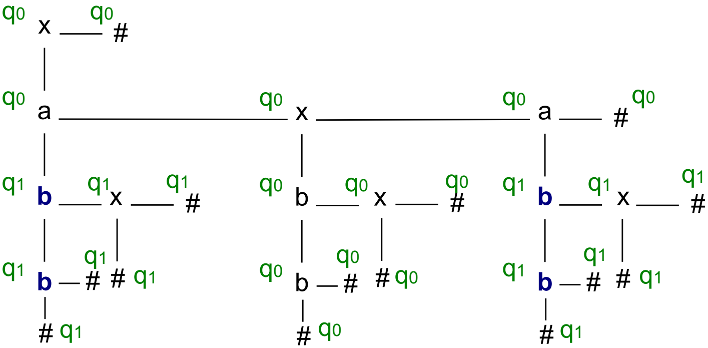
$$\delta = \begin{array}{llll} q_0,\{a\} & \rightarrow (q_1,q_0) & q_1,\{b\} & \Rightarrow (q_1,q_1) \\ q_0,\Sigma\setminus\{a\} & \rightarrow (q_0,q_0) & q_1,\Sigma\setminus\{b\} & \rightarrow (q_1,q_1) \end{array}$$



Example (//a//b)

$$\mathcal{A}_{//\mathtt{a}//\mathtt{b}} = (\underbrace{\{\mathtt{a},\mathtt{b},\mathtt{x}\}}_{\Sigma}, \underbrace{\{q_0,q_1\}}_{Q}, \underbrace{\{q_0\}}_{T}, \underbrace{\{q_0,q_1\}}_{B}, \underbrace{\{(q_1,\mathtt{b})\}}_{S}, \delta)$$

$$\delta = \begin{array}{llll} q_0,\{\mathtt{a}\} & \to (q_1,q_0) & q_1,\{\mathtt{b}\} & \Rightarrow (q_1,q_1) \\ q_0,\Sigma\setminus\{\mathtt{a}\} & \to (q_0,q_0) & q_1,\Sigma\setminus\{\mathtt{b}\} & \to (q_1,q_1) \end{array}$$



Definition

An STA is top-down (resp. bottom-up) deterministic if there is a unique top-down (resp. bottom-up) run

Definition

An STA is top-down (resp. bottom-up) deterministic if there is a unique top-down (resp. bottom-up) run

We write $\mathcal{A}_\top$ the universal recognizer (no selection)

Definition (Equivalence)

$\mathcal{A} \equiv \mathcal{A}'$ if and only if :

- $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$
- $\forall t \in T(\Sigma), \mathcal{A}(t) = \mathcal{A}'(t)$

Definition (Restriction to a state)

Let $\mathcal{A} = (\Sigma, Q, T, B, S, \delta)$. $\mathcal{A}[q] \equiv (\Sigma, Q, \{q\}, B, S, \delta)$.

Main idea :

- An STA $\mathcal{A}$, a tree t and a run R of $\mathcal{A}$ over t

Main idea :

- An STA $\mathcal{A}$, a tree t and a run R of $\mathcal{A}$ over t
- Detect the nodes where $\mathcal{A}$ "does something"

Main idea :

- An STA $\mathcal{A}$, a tree t and a run R of $\mathcal{A}$ over t
- Detect the nodes where $\mathcal{A}$ "does something"

Definition (Relevant node)

- let $\mathcal{A}$ be an STA, $t \in T(\Sigma)$ and R an accepting run
- let $\pi \in \mathrm{Dom}(t)$ s.t $\pi \cdot i \in \mathrm{Dom}(t), i \in \{1, 2\}$

π is relevant in R iff it is selected or none of the following hold :

- $\mathcal{A}[R(\pi)] \equiv \mathcal{A}[R(\pi \cdot 1)] \equiv \mathcal{A}[R(\pi \cdot 2)]$
- $\mathcal{A}[R(\pi)] \equiv \mathcal{A}[R(\pi \cdot 1)]$ and $\mathcal{A}[R(\pi \cdot 2)] \equiv \mathcal{A}_{\top}$
- $\mathcal{A}[R(\pi)] \equiv \mathcal{A}[R(\pi \cdot 2)]$ and $\mathcal{A}[R(\pi \cdot 1)] \equiv \mathcal{A}_{\top}$

So we can tell which nodes are relevant to a query…

So we can tell which nodes are relevant to a query. . .

:-(

Checking STA equivalence is hard (EXPTIME-complete)

Non-det. $\Rightarrow$ many runs $\Rightarrow$ different sets of relevant nodes

Consider a minimal top-down deterministic STA (TDSTA)

$$\mathcal{A}[q] \equiv \mathcal{A}_{\top} \quad \Leftrightarrow \quad q, \Sigma \to q, q$$

Consider a minimal top-down deterministic STA (TDSTA)

$$\mathcal{A}[q] \equiv \mathcal{A}_\top \quad \Leftrightarrow \quad q, \Sigma \rightarrow q, q$$

$$\mathcal{A}[q] \equiv \mathcal{A}[q'] \quad \Leftrightarrow \quad q \equiv q'$$

Consider a minimal top-down deterministic STA (TDSTA)

$$\mathcal{A}[q] \equiv \mathcal{A}_{\top} \quad \Leftrightarrow \quad q, \Sigma \rightarrow q, q$$

$$\mathcal{A}[q] \equiv \mathcal{A}[q'] \quad \Leftrightarrow \quad q \equiv q'$$

Toptdown-relevance

In a minimal TDSTA, a node is relevant iff it is either selected or if a state-change occurs for that node.

Consider a minimal top-down deterministic STA (TDSTA)

$$\mathcal{A}[q] \equiv \mathcal{A}_\top \quad \Leftrightarrow \quad q, \Sigma \rightarrow q, q$$

$$\mathcal{A}[q] \equiv \mathcal{A}[q'] \quad \Leftrightarrow \quad q \equiv q'$$

Toptdown-relevance

In a minimal TDSTA, a node is relevant iff it is either selected or if a state-change occurs for that node.

Example (//a//b)

$$\mathcal{A}_{//a//b} = (\underbrace{\{a, b, x\}}_{\Sigma}, \underbrace{\{q_0, q_1\}}_{Q}, \underbrace{\{q_0\}}_{T}, \underbrace{\{q_0, q_1\}}_{B}, \underbrace{\{(q_1, b)\}}_{S}, \delta)$$

$$\delta = \begin{array}{llll} q_0, \{a\} & \rightarrow (q_1, q_0) & q_1, \{b\} & \Rightarrow (q_1, q_1) \\ q_0, \Sigma \setminus \{a\} & \rightarrow (q_0, q_0) & q_1, \Sigma \setminus \{b\} & \rightarrow (q_1, q_1) \end{array}$$

Consider a minimal top-down deterministic STA (TDSTA)

$$\mathcal{A}[q] \equiv \mathcal{A}_\top \quad \Leftrightarrow \quad q, \Sigma \rightarrow q, q$$

$$\mathcal{A}[q] \equiv \mathcal{A}[q'] \quad \Leftrightarrow \quad q \equiv q'$$

Toptdown-relevance

In a minimal TDSTA, a node is relevant iff it is either selected or if a state-change occurs for that node.

Example (//a//b)

$$\mathcal{A}_{//\mathtt{a}//\mathtt{b}} = (\underbrace{\{\mathtt{a},\mathtt{b},\mathtt{x}\}}_{\Sigma}, \underbrace{\{q_0, q_1\}}_{Q}, \underbrace{\{q_0\}}_{T}, \underbrace{\{q_0, q_1\}}_{B}, \underbrace{\{(q_1, \mathtt{b})\}}_{S}, \delta)$$

$$\delta = \begin{array}{llll} q_0, \{\mathtt{a}\} & \rightarrow (q_1, q_0) & q_1, \{\mathtt{b}\} & \Rightarrow (q_1, q_1) \\ q_0, \Sigma \setminus \{\mathtt{a}\} & \rightarrow (q_0, q_0) & q_1, \Sigma \setminus \{\mathtt{b}\} & \rightarrow (q_1, q_1) \end{array}$$

- Left jump :

$$
\begin{aligned}
q, L_1 &\rightarrow (q_1^1, q_1^2) \\
&\vdots \\
q, L_n &\rightarrow (q_n^1, q_n^2) \\
q, \Sigma - \bigcup_{i \in 1..n} L_i &\rightarrow (q, q_\top)
\end{aligned}
$$

- Left jump :

$$
\begin{array}{ll}
q, L_1 & \rightarrow (q_1^1, q_1^2) \\
 & \vdots \\
q, L_n & \rightarrow (q_n^1, q_n^2) \\
q, \Sigma - \bigcup_{i \in 1..n} L_i & \rightarrow (q, q_\top)
\end{array}
$$

Note : in a TDSTA, L_is are pairwise disjoint.

- Left jump :

$$
\begin{array}{ll}
q, L_1 & \rightarrow (q_1^1, q_1^2) \\
 & \vdots \\
q, L_n & \rightarrow (q_n^1, q_n^2) \\
q, \Sigma - \bigcup_{i \in 1..n} L_i & \rightarrow (q, q_\top)
\end{array}
$$

Note : in a TDSTA, L_is are pairwise disjoint.
A state change is decided by occurences of particular labels

- Left jump :

$$\begin{array}{ll} q, L_1 & \rightarrow (q_1^1, q_1^2) \\ & \vdots \\ q, L_n & \rightarrow (q_n^1, q_n^2) \\ q, \Sigma - \bigcup_{i \in 1..n} L_i & \rightarrow (q, q_\top) \end{array}$$

  Note : in a TDSTA, L_is are pairwise disjoint.
  A state change is decided by occurences of particular labels
- Right jump : symmetric (i.e. $q, \Sigma - \bigcup_{i \in 1..n} L_i \;\rightarrow\; (q_\top, q)$)

- Left jump :

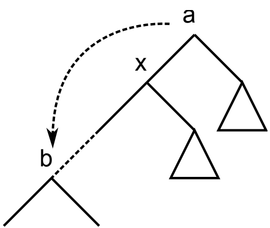
$$\begin{array}{ll} q, L_1 & \rightarrow (q_1^1, q_1^2) \\ & \vdots \\ q, L_n & \rightarrow (q_n^1, q_n^2) \\ q, \Sigma - \bigcup_{i \in 1..n} L_i & \rightarrow (q, q_\top) \end{array}$$

  Note : in a TDSTA, L_is are pairwise disjoint.
  A state change is decided by occurences of particular labels
- Right jump : symmetric (i.e. $q, \Sigma - \bigcup_{i \in 1..n} L_i \rightarrow (q_\top, q)$)
- Topmost nodes :

$$\begin{array}{ll} q, L_1 & \rightarrow (q_1^1, q_1^2) \\ & \vdots \\ q, L_n & \rightarrow (q_n^1, q_n^2) \\ q, \Sigma - \bigcup_{i \in 1..n} L_i & \rightarrow (q, q) \end{array}$$

- Left jump :

$$\begin{aligned} q, L_1 &\rightarrow (q_1^1, q_1^2) \\ &\vdots \\ q, L_n &\rightarrow (q_n^1, q_n^2) \\ q, \Sigma - \bigcup_{i \in 1..n} L_i &\rightarrow (q, q_\top) \end{aligned}$$

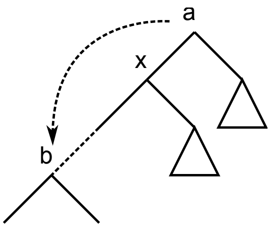


Note : in a TDSTA, L_is are pairwise disjoint.
A state change is decided by occurences of particular labels

- Right jump : symmetric (i.e. $q, \Sigma - \bigcup_{i \in 1..n} L_i \rightarrow (q_\top, q)$)
- Topmost nodes :

$$\begin{aligned} q, L_1 &\rightarrow (q_1^1, q_1^2) \\ &\vdots \\ q, L_n &\rightarrow (q_n^1, q_n^2) \\ q, \Sigma - \bigcup_{i \in 1..n} L_i &\rightarrow (q, q) \end{aligned}$$

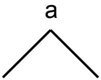
- Left jump :



$$q, L_1 \rightarrow (q_1^1, q_1^2)$$
$$\vdots$$
$$q, L_n \rightarrow (q_n^1, q_n^2)$$
$$q, \Sigma - \bigcup_{i \in 1..n} L_i \rightarrow (q, q_\top)$$

Note : in a TDSTA, L_is are pairwise disjoint.
A state change is decided by occurences of particular labels

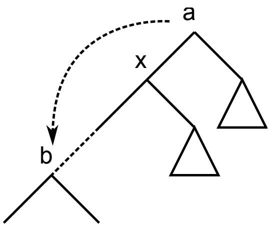
- Right jump : symmetric (i.e. $q, \Sigma - \bigcup_{i \in 1..n} L_i \rightarrow (q_\top, q)$)
- Topmost nodes :



$$q, L_1 \rightarrow (q_1^1, q_1^2)$$
$$\vdots$$
$$q, L_n \rightarrow (q_n^1, q_n^2)$$
$$q, \Sigma - \bigcup_{i \in 1..n} L_i \rightarrow (q, q)$$

- Left jump :

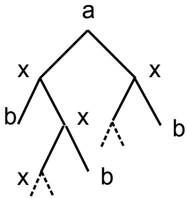
$$
\begin{array}{ll}
q, L_1 & \rightarrow (q_1^1, q_1^2) \\
 & \vdots \\
q, L_n & \rightarrow (q_n^1, q_n^2) \\
q, \Sigma - \bigcup_{i \in 1..n} L_i & \rightarrow (q, q_\top)
\end{array}
$$



Note : in a TDSTA, L_is are pairwise disjoint.
A state change is decided by occurences of particular labels

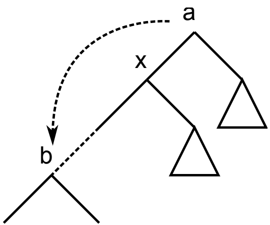
- Right jump : symmetric (i.e. $q, \Sigma - \bigcup_{i \in 1..n} L_i \rightarrow (q_\top, q)$)
- Topmost nodes :

$$
\begin{array}{ll}
q, L_1 & \rightarrow (q_1^1, q_1^2) \\
 & \vdots \\
q, L_n & \rightarrow (q_n^1, q_n^2) \\
q, \Sigma - \bigcup_{i \in 1..n} L_i & \rightarrow (q, q)
\end{array}
$$



- Left jump :

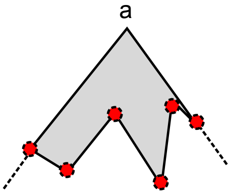
$$\begin{array}{ll} q, L_1 & \to (q_1^1, q_1^2) \\ & \vdots \\ q, L_n & \to (q_n^1, q_n^2) \\ q, \Sigma - \bigcup_{i\in 1..n} L_i & \to (q, q_\top) \end{array}$$



Note : in a TDSTA, L_is are pairwise disjoint.
A state change is decided by occurences of particular labels

- Right jump : symmetric (i.e. $q, \Sigma - \bigcup_{i\in 1..n} L_i \;\to\; (q_\top, q)$)
- Topmost nodes :

$$\begin{array}{ll} q, L_1 & \to (q_1^1, q_1^2) \\ & \vdots \\ q, L_n & \to (q_n^1, q_n^2) \\ q, \Sigma - \bigcup_{i\in 1..n} L_i & \to (q, q) \end{array}$$



- Defined symmetrically by a state-change in the bottom-up run for a minimal BDSTA
- Automata moves :
  - bottom-most nodes with some label
  - lowest common ancestor of two nodes with some label
  - lowest node in the $\uparrow_1 *$ or $\uparrow_2 *$ direction with some label

- Defined symmetrically by a state-change in the bottom-up run for a minimal BDSTA
- Automata moves :
  - bottom-most nodes with some label
  - lowest common ancestor of two nodes with some label
  - lowest node in the $\uparrow_1 *$ or $\uparrow_2 *$ direction with some label

Problem : Top-down det. and bottom-up det. are incomparable and weaker than non-det.

Example

- `//a//b` is TD but not BD

- Defined symmetrically by a state-change in the bottom-up run for a minimal BDSTA
- Automata moves :
  - bottom-most nodes with some label
  - lowest common ancestor of two nodes with some label
  - lowest node in the $\uparrow_1 *$ or $\uparrow_2 *$ direction with some label

Problem : Top-down det. and bottom-up det. are incomparable and weaker than non-det.

Example

- `//a//b` is TD but not BD
- `//a[.//b]` is BD but not TD

- Defined symmetrically by a state-change in the bottom-up run for a minimal BDSTA
- Automata moves :
    - bottom-most nodes with some label
    - lowest common ancestor of two nodes with some label
    - lowest node in the $\uparrow_1 *$ or $\uparrow_2 *$ direction with some label

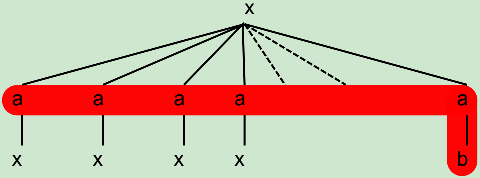
Problem : Top-down det. and bottom-up det. are incomparable and weaker than non-det.

Example

- `//a//b` is TD but not BD
- `//a[.//b]` is BD but not TD
- `//a//b[.//c]` is neither TD nor BD

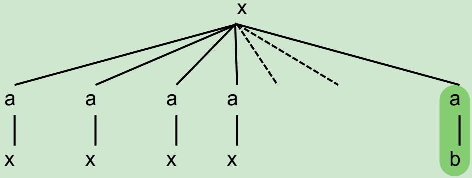
Example (`//a//b`)

`//a//b` is top-down deterministic.



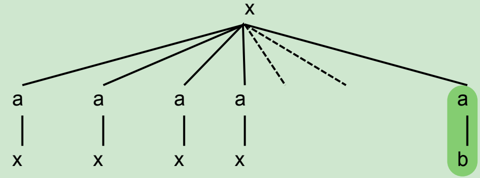
Example (`//a//b`)

`//a//b` is top-down deterministic.However, there is a bottom-up non-deterministic run with less relevant nodes



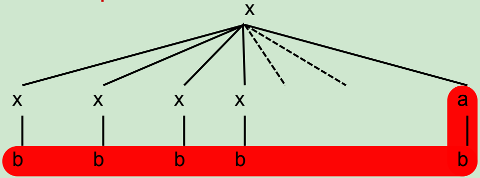
Example (//a//b)

//a//b is top-down deterministic.However, there is a bottom-up non-deterministic run with less relevant nodes



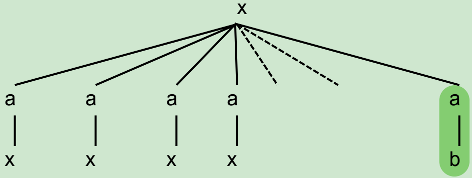
Example (//a[.//b])

//a[.//b] is bottom-up deterministic.



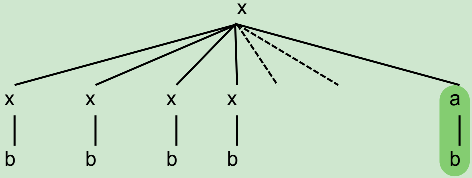
Example (//a//b)

//a//b is top-down deterministic.However, there is a bottom-up non-deterministic run with less relevant nodes



Example (//a[.//b])

//a[.//b] is bottom-up deterministic. However, there is a top-down non-deterministic run with less relevant nodes



We cannot use STA directly :

1. XPath query $\rightsquigarrow$ STA : exponential blow-up
2. we only know how to do things for minimal TD or BD STA

We cannot use STA directly :

1. XPath query $\rightsquigarrow$ STA : exponential blow-up
2. we only know how to do things for minimal TD or BD STA

Definition

Alternating Selecting Tree Automata (ASTA) :

- transitions of the form q, L, τ, ϕ
- τ is either $\rightarrow$ (normal) or $\Rightarrow$ (selecting)
- ϕ is a boolean formula generated by :

$$\phi \quad ::= \quad \top \mid \bot \mid \phi \vee \phi \mid \phi \wedge \phi \mid \neg\phi \mid \downarrow_1 q \mid \downarrow_2 q \quad (q \in Q)$$

We cannot use STA directly :

1. XPath query $\rightsquigarrow$ STA : exponential blow-up
2. we only know how to do things for minimal TD or BD STA

Definition

Alternating Selecting Tree Automata (ASTA) :

- transitions of the form q, L, τ, ϕ
- τ is either $\rightarrow$ (normal) or $\Rightarrow$ (selecting)
- ϕ is a boolean formula generated by :

$$\phi \quad ::= \quad \top \mid \bot \mid \phi \vee \phi \mid \phi \wedge \phi \mid \neg\phi \mid \downarrow_1 q \mid \downarrow_2 q \quad (q \in Q)$$

Example (//a//b[c])

$$\mathcal{A}_{//\texttt{a}//\texttt{b(c)}} = (\Sigma, \{q_0, q_1, q_2\}, \{q_0\}, \delta)$$

where δ is :

$$\begin{array}{l|l|l} q_0, \{a\} \rightarrow \downarrow_1 q_1 & q_1, \{b\} \Rightarrow \downarrow_1 q_2 & q_2, \{c\} \rightarrow \top \\ q_0, \Sigma \rightarrow \downarrow_1 q_0 \vee \downarrow_2 q_0 & q_1, \Sigma \rightarrow \downarrow_1 q_1 \vee \downarrow_2 q_1 & q_2, \Sigma \rightarrow \downarrow_2 q_2 \end{array}$$

Allow us to write XPath queries with :

- `self`, `child`, `descendant`, `following-sibling`
- tag test or $*$
- arbitrary filters built on paths, `or`, `and`, `not`

Allow us to write XPath queries with :

- `self`, `child`, `descendant`, `following-sibling`
- tag test or $*$
- arbitrary filters built on paths, `or`, `and`, `not`

The translation is linear in the size of the query

Allow us to write XPath queries with :

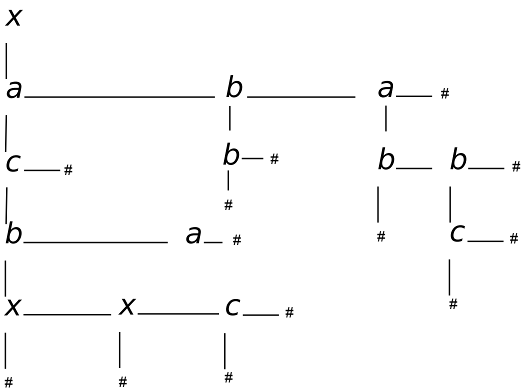
- `self`, `child`, `descendant`, `following-sibling`
- tag test or $*$
- arbitrary filters built on paths, `or`, `and`, `not`

The translation is linear in the size of the query

We can use a textbook evaluation algorithm : $O(|D| \cdot |\mathcal{A}|)$

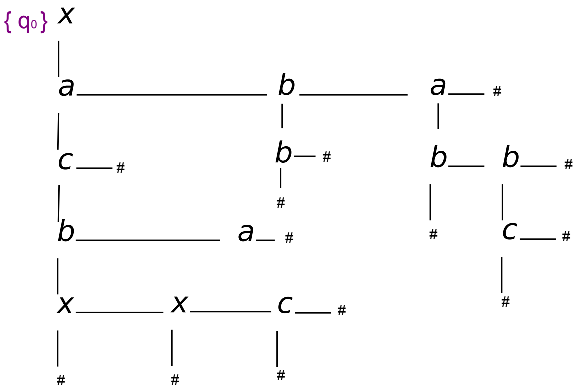
Example (//a//b[c])

$q_0, \{a\} \rightarrow \downarrow_1 q_1$	$q_1, \{b\} \Rightarrow \downarrow_1 q_2$	$q_2, \{c\} \rightarrow \top$
$q_0, \Sigma \rightarrow \downarrow_1 q_0 \vee \downarrow_2 q_0$	$q_1, \Sigma \rightarrow \downarrow_1 q_1 \vee \downarrow_2 q_1$	$q_2, \Sigma \rightarrow \downarrow_2 q_2$



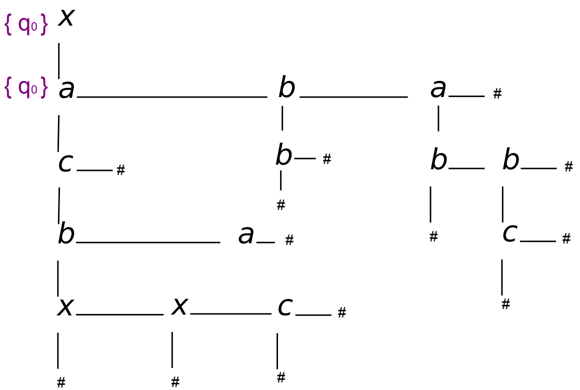
Example (`//a//b[c]`)

$q_0, \{a\} \rightarrow \downarrow_1 q_1$	$q_1, \{b\} \Rightarrow \downarrow_1 q_2$	$q_2, \{c\} \rightarrow \top$
$q_0, \Sigma \rightarrow \downarrow_1 q_0 \vee \downarrow_2 q_0$	$q_1, \Sigma \rightarrow \downarrow_1 q_1 \vee \downarrow_2 q_1$	$q_2, \Sigma \rightarrow \downarrow_2 q_2$



Example (`//a//b[c]`)

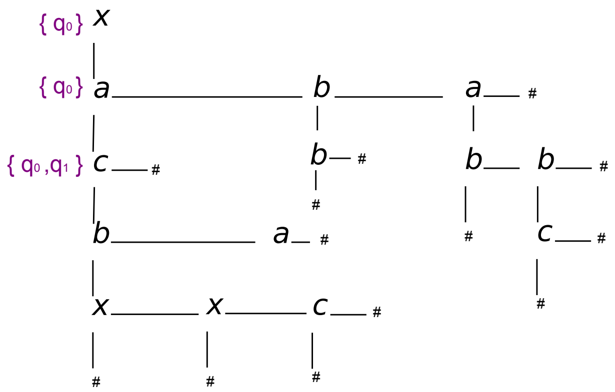
$q_0, \{a\} \rightarrow \downarrow_1 q_1$	$q_1, \{b\} \Rightarrow \downarrow_1 q_2$	$q_2, \{c\} \rightarrow \top$
$q_0, \Sigma \rightarrow \downarrow_1 q_0 \vee \downarrow_2 q_0$	$q_1, \Sigma \rightarrow \downarrow_1 q_1 \vee \downarrow_2 q_1$	$q_2, \Sigma \rightarrow \downarrow_2 q_2$



Example (`//a//b[c]`)

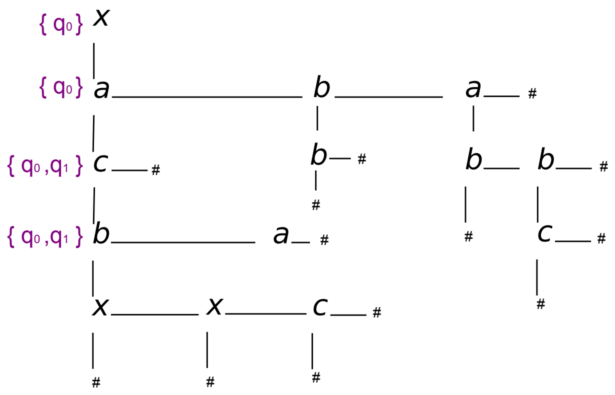
$q_0, \{a\} \rightarrow \downarrow_1 q_1$ | $q_1, \{b\} \Rightarrow \downarrow_1 q_2$ | $q_2, \{c\} \rightarrow \top$

$q_0, \Sigma \rightarrow \downarrow_1 q_0 \vee \downarrow_2 q_0$ | $q_1, \Sigma \rightarrow \downarrow_1 q_1 \vee \downarrow_2 q_1$ | $q_2, \Sigma \rightarrow \downarrow_2 q_2$



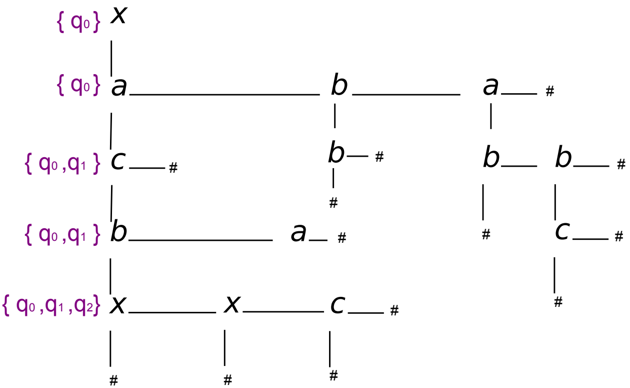
Example (//a//b[c])

$q_0, \{a\} \rightarrow \downarrow_1 q_1$	$q_1, \{b\} \Rightarrow \downarrow_1 q_2$	$q_2, \{c\} \rightarrow \top$
$q_0, \Sigma \rightarrow \downarrow_1 q_0 \vee \downarrow_2 q_0$	$q_1, \Sigma \rightarrow \downarrow_1 q_1 \vee \downarrow_2 q_1$	$q_2, \Sigma \rightarrow \downarrow_2 q_2$



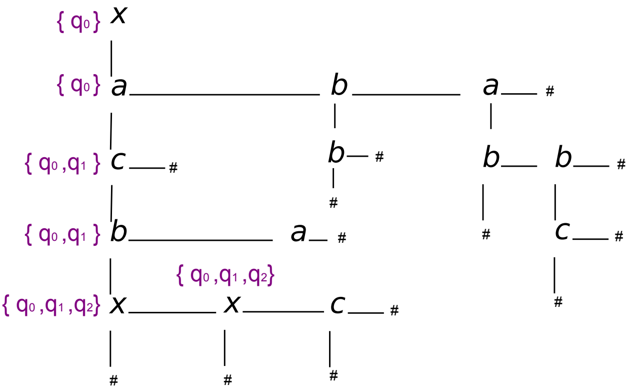
Example (`//a//b[c]`)

$q_0, \{a\} \rightarrow \downarrow_1 q_1$	$q_1, \{b\} \Rightarrow \downarrow_1 q_2$	$q_2, \{c\} \rightarrow \top$
$q_0, \Sigma \rightarrow \downarrow_1 q_0 \vee \downarrow_2 q_0$	$q_1, \Sigma \rightarrow \downarrow_1 q_1 \vee \downarrow_2 q_1$	$q_2, \Sigma \rightarrow \downarrow_2 q_2$



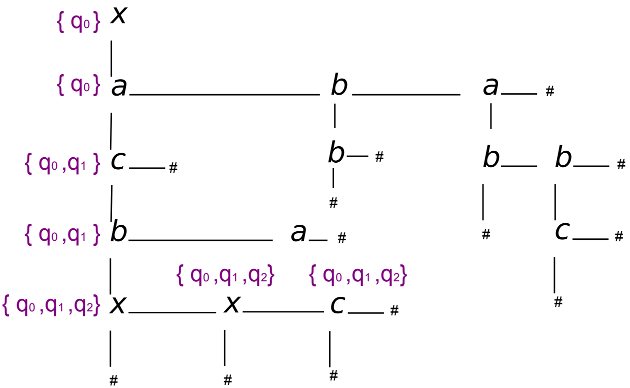
Example (//a//b[c])

$q_0, \{a\} \rightarrow \downarrow_1 q_1$	$q_1, \{b\} \Rightarrow \downarrow_1 q_2$	$q_2, \{c\} \rightarrow \top$
$q_0, \Sigma \rightarrow \downarrow_1 q_0 \vee \downarrow_2 q_0$	$q_1, \Sigma \rightarrow \downarrow_1 q_1 \vee \downarrow_2 q_1$	$q_2, \Sigma \rightarrow \downarrow_2 q_2$



Example (`//a//b[c]`)

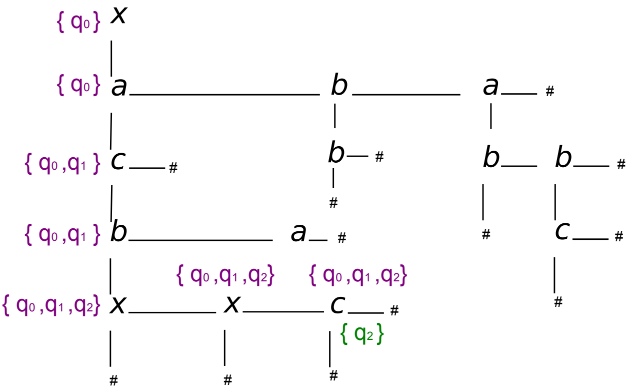
$q_0, \{a\} \rightarrow \downarrow_1 q_1$	$q_1, \{b\} \Rightarrow \downarrow_1 q_2$	$q_2, \{c\} \rightarrow \top$
$q_0, \Sigma \rightarrow \downarrow_1 q_0 \vee \downarrow_2 q_0$	$q_1, \Sigma \rightarrow \downarrow_1 q_1 \vee \downarrow_2 q_1$	$q_2, \Sigma \rightarrow \downarrow_2 q_2$



Example (`//a//b[c]`)

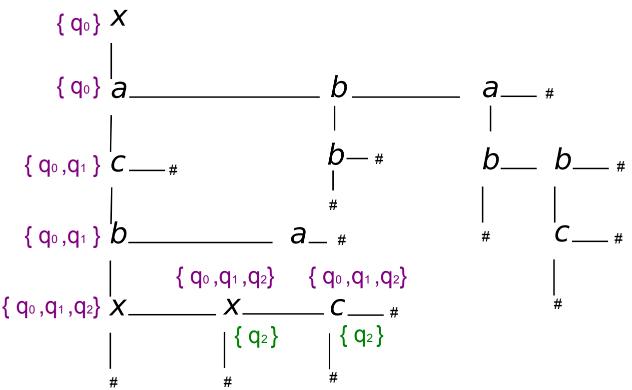
$q_0, \{a\} \rightarrow \downarrow_1 q_1$ | $q_1, \{b\} \Rightarrow \downarrow_1 q_2$ | $q_2, \{c\} \rightarrow \top$

$q_0, \Sigma \rightarrow \downarrow_1 q_0 \vee \downarrow_2 q_0$ | $q_1, \Sigma \rightarrow \downarrow_1 q_1 \vee \downarrow_2 q_1$ | $q_2, \Sigma \rightarrow \downarrow_2 q_2$



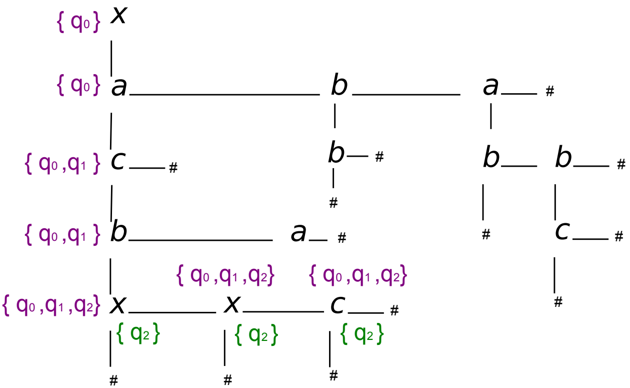
Example (`//a//b[c]`)

$q_0, \{a\} \rightarrow \downarrow_1 q_1$	$q_1, \{b\} \Rightarrow \downarrow_1 q_2$	$q_2, \{c\} \rightarrow \top$
$q_0, \Sigma \rightarrow \downarrow_1 q_0 \vee \downarrow_2 q_0$	$q_1, \Sigma \rightarrow \downarrow_1 q_1 \vee \downarrow_2 q_1$	$q_2, \Sigma \rightarrow \downarrow_2 q_2$



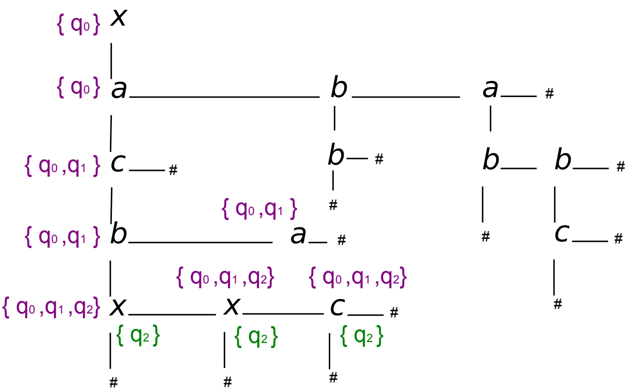
Example (//a//b[c])

$q_0, \{a\} \rightarrow \downarrow_1 q_1$	$q_1, \{b\} \Rightarrow \downarrow_1 q_2$	$q_2, \{c\} \rightarrow \top$
$q_0, \Sigma \rightarrow \downarrow_1 q_0 \vee \downarrow_2 q_0$	$q_1, \Sigma \rightarrow \downarrow_1 q_1 \vee \downarrow_2 q_1$	$q_2, \Sigma \rightarrow \downarrow_2 q_2$



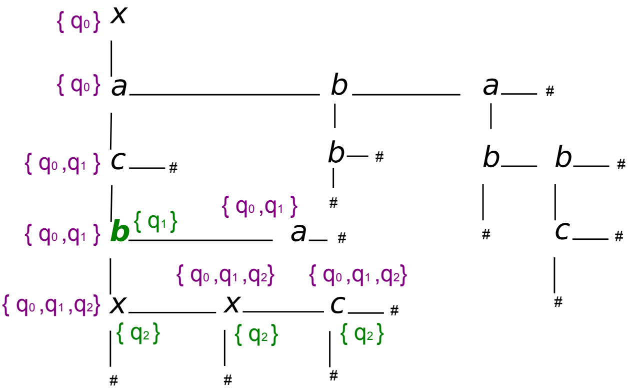
Example (//a//b[c])

$q_0, \{a\} \rightarrow \downarrow_1 q_1$	$q_1, \{b\} \Rightarrow \downarrow_1 q_2$	$q_2, \{c\} \rightarrow \top$
$q_0, \Sigma \rightarrow \downarrow_1 q_0 \vee \downarrow_2 q_0$	$q_1, \Sigma \rightarrow \downarrow_1 q_1 \vee \downarrow_2 q_1$	$q_2, \Sigma \rightarrow \downarrow_2 q_2$



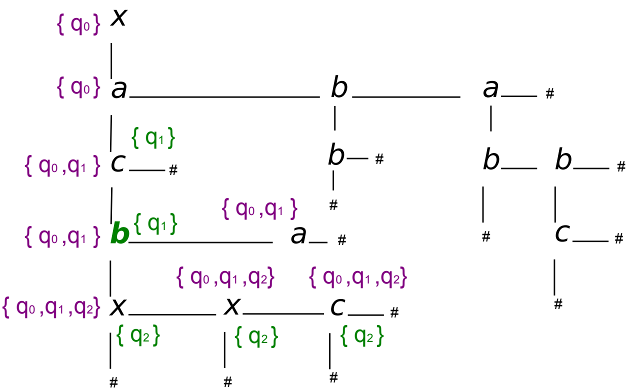
Example (`//a//b[c]`)

$q_0, \{a\} \rightarrow \downarrow_1 q_1$	$q_1, \{b\} \Rightarrow \downarrow_1 q_2$	$q_2, \{c\} \rightarrow \top$
$q_0, \Sigma \rightarrow \downarrow_1 q_0 \vee \downarrow_2 q_0$	$q_1, \Sigma \rightarrow \downarrow_1 q_1 \vee \downarrow_2 q_1$	$q_2, \Sigma \rightarrow \downarrow_2 q_2$



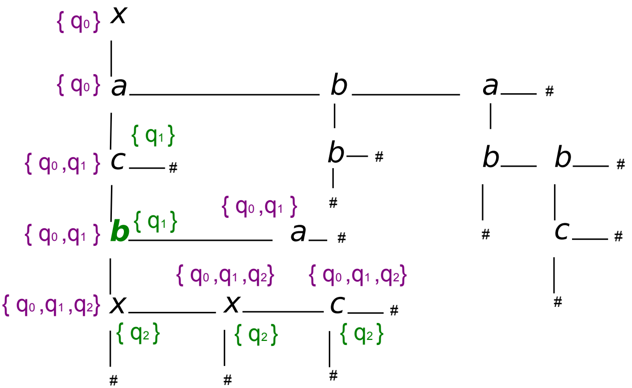
Example (`//a//b[c]`)

$q_0, \{a\} \rightarrow \downarrow_1 q_1$	$q_1, \{b\} \Rightarrow \downarrow_1 q_2$	$q_2, \{c\} \rightarrow \top$
$q_0, \Sigma \rightarrow \downarrow_1 q_0 \vee \downarrow_2 q_0$	$q_1, \Sigma \rightarrow \downarrow_1 q_1 \vee \downarrow_2 q_1$	$q_2, \Sigma \rightarrow \downarrow_2 q_2$



Example (//a//b[c])

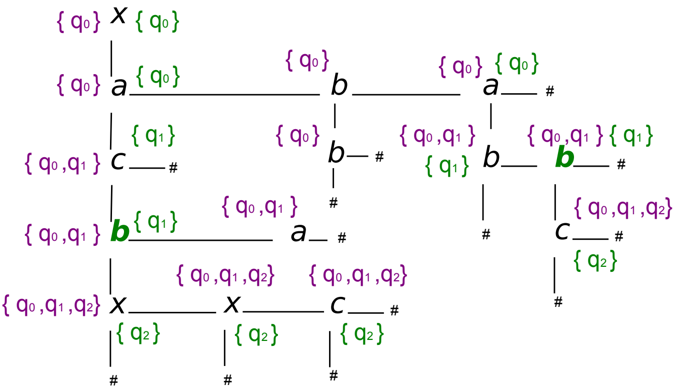
$$
\begin{array}{ll|ll|ll}
q_0, \{a\} & \rightarrow \downarrow_1 q_1 & q_1, \{b\} & \Rightarrow \downarrow_1 q_2 & q_2, \{c\} & \rightarrow \top \\
q_0, \Sigma & \rightarrow \downarrow_1 q_0 \vee \downarrow_2 q_0 & q_1, \Sigma & \rightarrow \downarrow_1 q_1 \vee \downarrow_2 q_1 & q_2, \Sigma & \rightarrow \downarrow_2 q_2
\end{array}
$$



Example (//a//b[c])

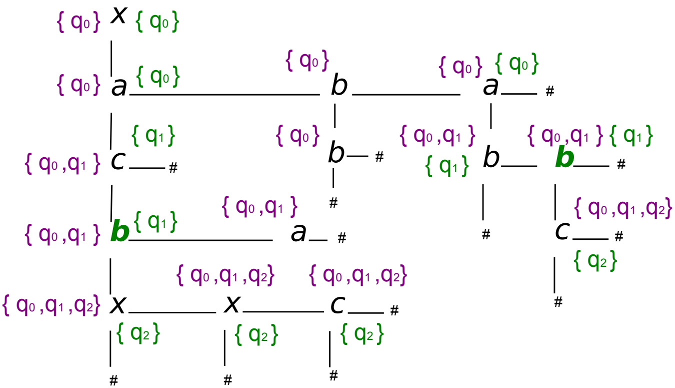
$q_0, \{a\} \rightarrow \downarrow_1 q_1$ | $q_1, \{b\} \Rightarrow \downarrow_1 q_2$ | $q_2, \{c\} \rightarrow \top$

$q_0, \Sigma \rightarrow \downarrow_1 q_0 \vee \downarrow_2 q_0$ | $q_1, \Sigma \rightarrow \downarrow_1 q_1 \vee \downarrow_2 q_1$ | $q_2, \Sigma \rightarrow \downarrow_2 q_2$



Example (`//a//b[c]`)

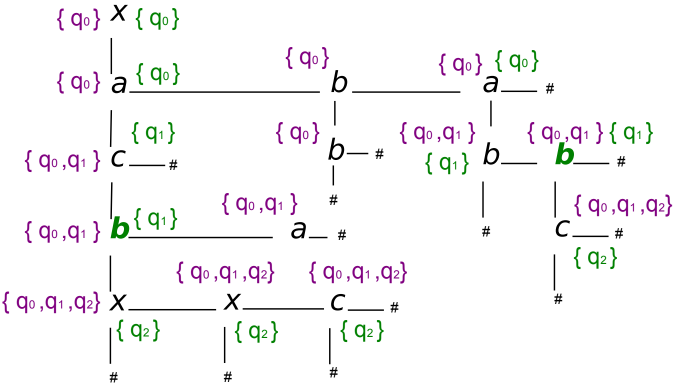
$$
\begin{array}{ll|ll|ll}
q_0,\{a\} & \rightarrow \downarrow_1 q_1 & q_1,\{b\} & \Rightarrow \downarrow_1 q_2 & q_2,\{c\} & \rightarrow \top \\
q_0,\Sigma & \rightarrow \downarrow_1 q_0 \vee \downarrow_2 q_0 & q_1,\Sigma & \rightarrow \downarrow_1 q_1 \vee \downarrow_2 q_1 & q_2,\Sigma & \rightarrow \downarrow_2 q_2
\end{array}
$$



$$
\begin{array}{lll}
Q_0 & = & \{q_0\} \\
Q_1 & = & \{q_0, q_1\} \\
Q_2 & = & \{q_0, q_1, q_2\}
\end{array}
$$

Example (`//a//b[c]`)

$q_0, \{a\} \rightarrow \downarrow_1 q_1$	$q_1, \{b\} \Rightarrow \downarrow_1 q_2$	$q_2, \{c\} \rightarrow \top$
$q_0, \Sigma \rightarrow \downarrow_1 q_0 \vee \downarrow_2 q_0$	$q_1, \Sigma \rightarrow \downarrow_1 q_1 \vee \downarrow_2 q_1$	$q_2, \Sigma \rightarrow \downarrow_2 q_2$



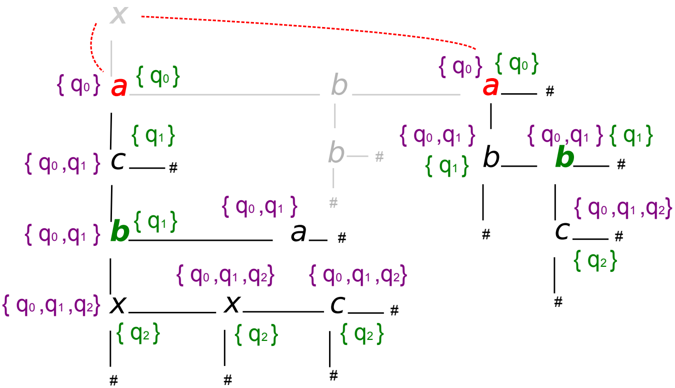
$$Q_0 = \{q_0\}$$
$$Q_1 = \{q_0, q_1\}$$
$$Q_2 = \{q_0, q_1, q_2\}$$

$\mathcal{A}_{\text{approx}}$

$Q_0, \{a\}$	$\rightarrow Q_1, Q_0$
$Q_0, \Sigma - \{a\}$	$\rightarrow Q_0, Q_0$
$Q_1, \{b\}$	$\rightarrow Q_2, Q_0$
$Q_1, \Sigma - \{b\}$	$\rightarrow Q_1, Q_1$
$Q_2, \{c\}$	$\rightarrow Q_1, Q_1$
$Q_2, \{b\}$	$\rightarrow Q_2, Q_2$
$Q_2, \Sigma - \{b, c\}$	$\rightarrow Q_1, Q_2$

Example (`//a//b[c]`)

$$
\begin{array}{lll|lll|lll}
q_0, \{a\} & \rightarrow & \downarrow_1 q_1 & q_1, \{b\} & \Rightarrow & \downarrow_1 q_2 & q_2, \{c\} & \rightarrow & \top \\
q_0, \Sigma & \rightarrow & \downarrow_1 q_0 \vee \downarrow_2 q_0 & q_1, \Sigma & \rightarrow & \downarrow_1 q_1 \vee \downarrow_2 q_1 & q_2, \Sigma & \rightarrow & \downarrow_2 q_2
\end{array}
$$



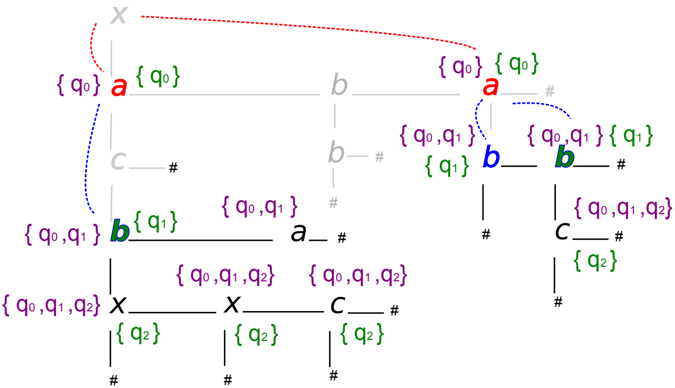
$$
\begin{array}{lcl}
Q_0 & = & \{q_0\} \\
Q_1 & = & \{q_0, q_1\} \\
Q_2 & = & \{q_0, q_1, q_2\}
\end{array}
$$

$\mathcal{A}_{\text{approx}}$

$$
\begin{array}{ll}
Q_0, \{a\} & \rightarrow Q_1, Q_0 \\
Q_0, \Sigma - \{a\} & \rightarrow Q_0, Q_0 \\
Q_1, \{b\} & \rightarrow Q_2, Q_0 \\
Q_1, \Sigma - \{b\} & \rightarrow Q_1, Q_1 \\
Q_2, \{c\} & \rightarrow Q_1, Q_1 \\
Q_2, \{b\} & \rightarrow Q_2, Q_2 \\
Q_2, \Sigma - \{b, c\} & \rightarrow Q_1, Q_2
\end{array}
$$

Example (`//a//b[c]`)

$q_0, \{a\} \rightarrow \downarrow_1 q_1$	$q_1, \{b\} \Rightarrow \downarrow_1 q_2$	$q_2, \{c\} \rightarrow \top$
$q_0, \Sigma \rightarrow \downarrow_1 q_0 \vee \downarrow_2 q_0$	$q_1, \Sigma \rightarrow \downarrow_1 q_1 \vee \downarrow_2 q_1$	$q_2, \Sigma \rightarrow \downarrow_2 q_2$



$$Q_0 = \{q_0\}$$
$$Q_1 = \{q_0, q_1\}$$
$$Q_2 = \{q_0, q_1, q_2\}$$

$\mathcal{A}_{\text{approx}}$

$Q_0, \{a\} \rightarrow Q_1, Q_0$

$Q_0, \Sigma - \{a\} \rightarrow Q_0, Q_0$

$Q_1, \{b\} \rightarrow Q_2, Q_0$

$Q_1, \Sigma - \{b\} \rightarrow Q_1, Q_1$

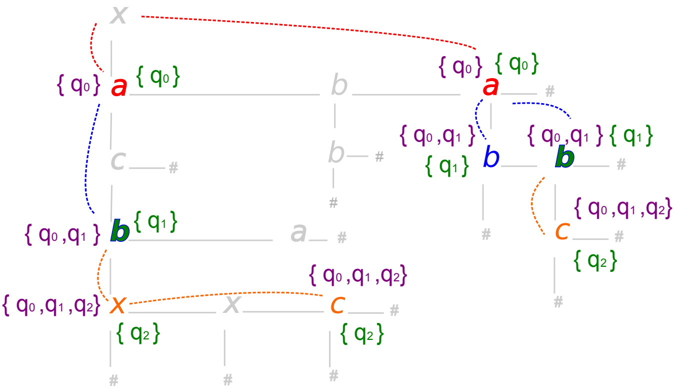
$Q_2, \{c\} \rightarrow Q_1, Q_1$

$Q_2, \{b\} \rightarrow Q_2, Q_2$

$Q_2, \Sigma - \{b, c\} \rightarrow Q_1, Q_2$

Example (`//a//b[c]`)

$$
\begin{array}{ll|ll|ll}
q_0,\{a\} \rightarrow \downarrow_1 q_1 & & q_1,\{b\} \Rightarrow \downarrow_1 q_2 & & q_2,\{c\} \rightarrow \top & \\
q_0,\Sigma \rightarrow \downarrow_1 q_0 \vee \downarrow_2 q_0 & & q_1,\Sigma \rightarrow \downarrow_1 q_1 \vee \downarrow_2 q_1 & & q_2,\Sigma \rightarrow \downarrow_2 q_2 &
\end{array}
$$



$$
\begin{array}{lcl}
Q_0 & = & \{q_0\} \\
Q_1 & = & \{q_0, q_1\} \\
Q_2 & = & \{q_0, q_1, q_2\}
\end{array}
$$

$\mathcal{A}_{\text{approx}}$

$$
\begin{array}{ll}
Q_0, \{a\} & \rightarrow Q_1, Q_0 \\
Q_0, \Sigma - \{a\} & \rightarrow Q_0, Q_0 \\
Q_1, \{b\} & \rightarrow Q_2, Q_0 \\
Q_1, \Sigma - \{b\} & \rightarrow Q_1, Q_1 \\
Q_2, \{c\} & \rightarrow Q_1, Q_1 \\
Q_2, \{b\} & \rightarrow Q_2, Q_2 \\
Q_2, \Sigma - \{b, c\} & \rightarrow Q_1, Q_2
\end{array}
$$

We use the SXSI index

"Fast In-Memory XPath Search using Compressed Indexes"
ICDE 2010

Gives us low-level primitives which can simulate jumps ($|D|$)

We use the SXSI index

"Fast In-Memory XPath Search using Compressed Indexes"
ICDE 2010

Gives us low-level primitives which can simulate jumps ($|D|$)

We perform just-in-time compilation of the ASTA ($|Q|$)

We use the SXSI index

"Fast In-Memory XPath Search using Compressed Indexes"
ICDE 2010

Gives us low-level primitives which can simulate jumps ($|D|$)

We perform just-in-time compilation of the ASTA ($|Q|$)

Early simplifications of formulas ($|Q|$ and $|D|$)

We use the SXSI index

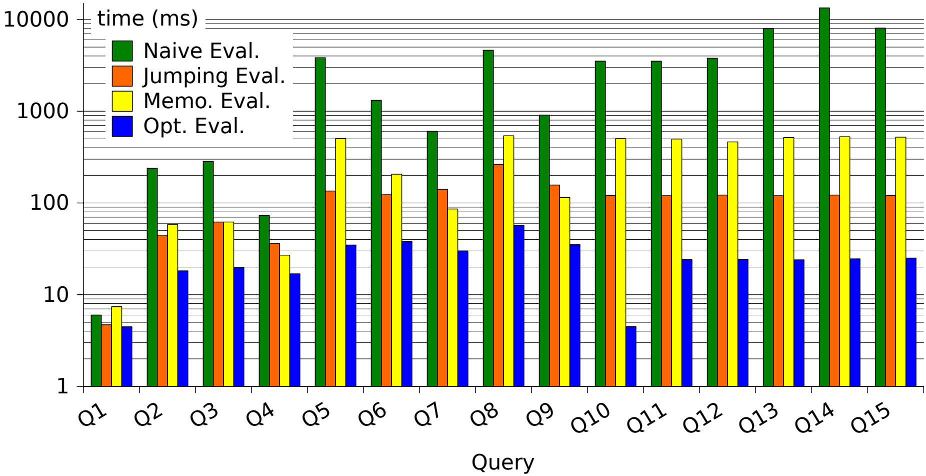
"Fast In-Memory XPath Search using Compressed Indexes"
ICDE 2010

Gives us low-level primitives which can simulate jumps ($|D|$)

We perform just-in-time compilation of the ASTA ($|Q|$)

Early simplifications of formulas ($|Q|$ and $|D|$)

Result-sets that allow $O(1)$ duplicate-free sorted insert ($|Q|$)



Q1-Q10 taken from XPathMark

Q5-Q15 contain //

Factored computations :

Q10 : `/site[ .//keyword ]` touches two nodes, return 1 node

Q11 : `/site//keyword` touches 73071 nodes, returns 73070

Q12 : `/site[.//keyword]//keyword` touches 73071 nodes

Q15 : `/site[ .//*//* ]//keyword` touches 73074 nodes

Factored computations :

Q10 : `/site[ .//keyword ]` touches two nodes, return 1 node

Q11 : `/site//keyword` touches 73071 nodes, returns 73070

Q12 : `/site[.//keyword]//keyword` touches 73071 nodes

Q15 : `/site[ .//*//* ]//keyword` touches 73074 nodes

Reliance to syntactic changes

`//a[ .//b or .//c ]//d` and `//a[ .//c or .//b ]//d` executed the same way (same nodes are touched)

- Tree automata are a nice tool to reason about queries

- Tree automata are a nice tool to reason about queries
- They give rise to the notion of relevant nodes

- Tree automata are a nice tool to reason about queries
- They give rise to the notion of relevant nodes
- They are also an effective compilation target

- Tree automata are a nice tool to reason about queries
- They give rise to the notion of relevant nodes
- They are also an effective compilation target
- The first time indexing and tree automata are combined

- We have partial handling of data-values (see ICDE 2010)

- We have partial handling of data-values (see ICDE 2010)
- Extend to joins on data-values

Future work

- We have partial handling of data-values (see ICDE 2010)
- Extend to joins on data-values
- Add the remaining XPath axes (possible in theory but we lack an efficient implementation)

- We have partial handling of data-values (see ICDE 2010)
- Extend to joins on data-values
- Add the remaining XPath axes (possible in theory but we lack an efficient implementation)
- Use static information (schemas, data-guides) to decide whether to run top-down or bottom-up

- We have partial handling of data-values (see ICDE 2010)
- Extend to joins on data-values
- Add the remaining XPath axes (possible in theory but we lack an efficient implementation)
- Use static information (schemas, data-guides) to decide whether to run top-down or bottom-up
- Can we characterize the average complexity ?