Querying Regular Sets of XML Documents

(with an application to Consistent Query Answering)

Slawek Staworko[1,2]
(joint work with E. Fillot[1] and J. Chomicki[2])

[1]INRIA Nord Europe, Lille

[2]University at Buffalo

Logic in Databases 2008

Motivation

In many novel XML database applications...

Instead of querying a given document t we try to reason about a collection of documents that are obtained from t by

- Some nondeterministic process/transformation (data cleaning, conflict resolution, consistent query answers)
- Some mapping that leaves many aspects of the target document unspecified (data exchange)

The collection of the documents is often regular

- Nondeterministic decisions are typically independent
- Output documents follow a DTD

Overview

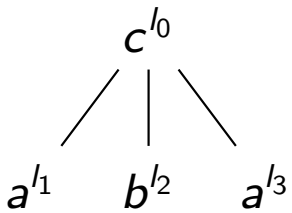
Trees and streams (with attributes)

Trees

- unranked, ordered, labeled with tags from a finite Σ
- node additionally labeled with a attribute from an infinite Λ (node identifier, text values, etc.)

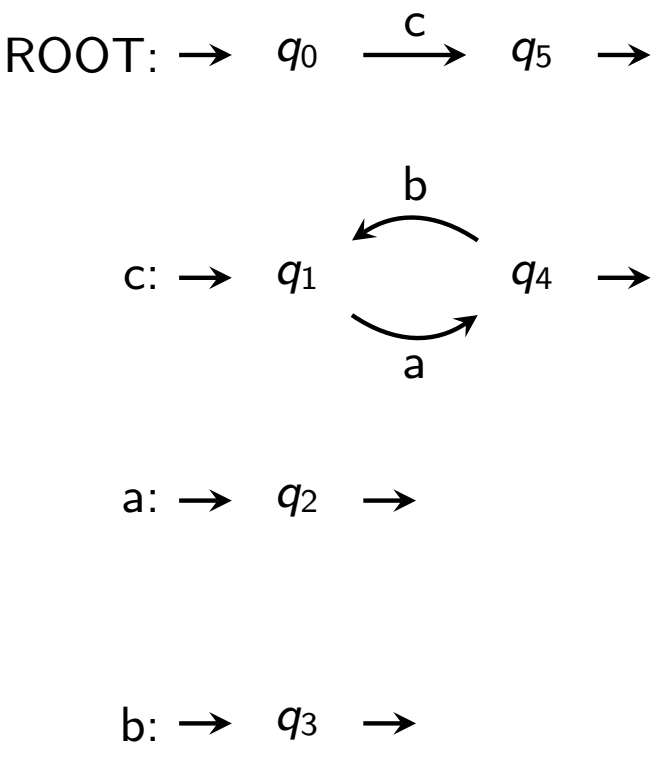
Streams

- well-formed sequence of opening and closing tags
- opening tags have a text attribute

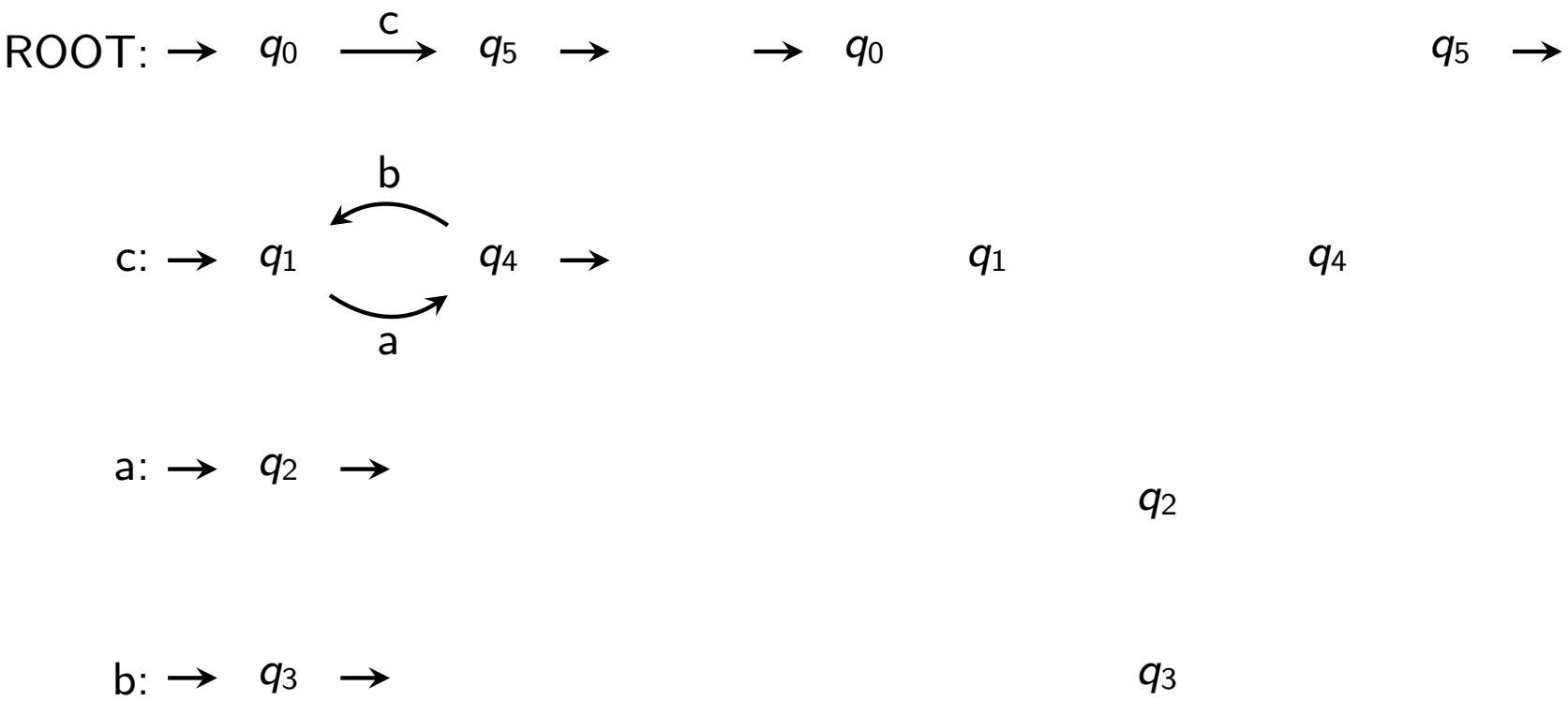


	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8
tag	$\langle c \rangle$	$\langle a \rangle$	$\langle /a \rangle$	$\langle b \rangle$	$\langle /b \rangle$	$\langle a \rangle$	$\langle /a \rangle$	$\langle /c \rangle$
att	l_0	l_1	–	l_2	–	l_3	–	–

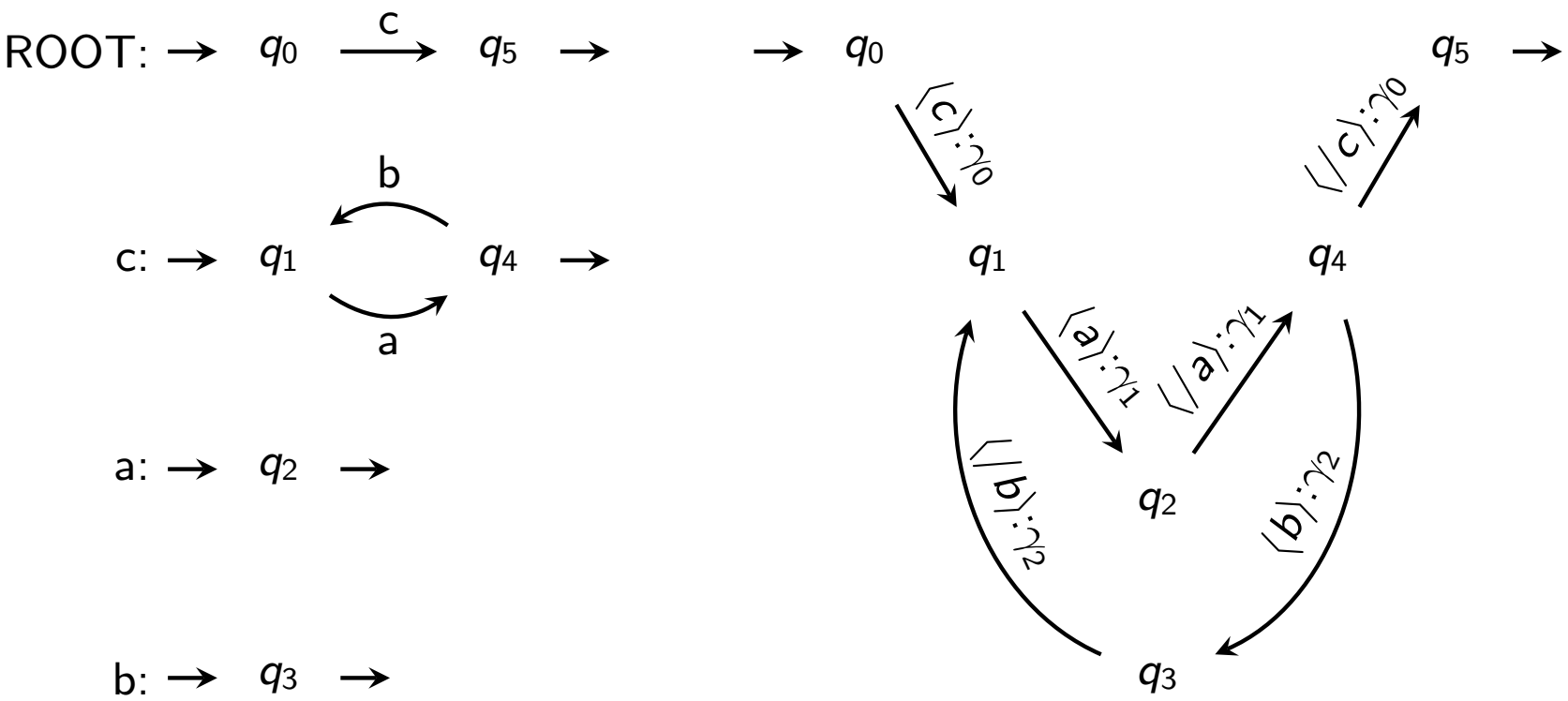
Streaming Tree Automata (derived from VPA)



Streaming Tree Automata (derived from VPA)



Streaming Tree Automata (derived from VPA)

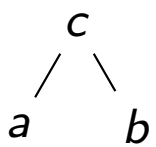


Attributed STAs

$$\longrightarrow q_0 \xrightarrow{\langle c \rangle : \gamma} q_1 \xrightarrow{\langle a \rangle : \gamma} q_2 \xrightarrow{\langle /a \rangle : \gamma} q_3 \xrightarrow{\langle b \rangle : \gamma} q_4 \xrightarrow{\langle /b \rangle : \gamma} q_5 \xrightarrow{\langle /c \rangle : \gamma} q_6 \longrightarrow$$

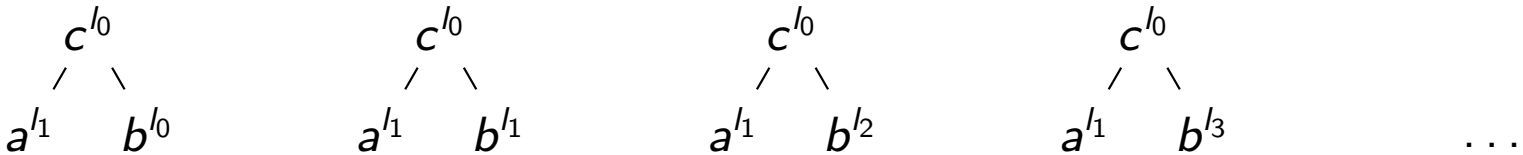
Attributed STAs

$$\longrightarrow q_0 \xrightarrow{\langle c \rangle : \gamma} q_1 \xrightarrow{\langle a \rangle : \gamma} q_2 \xrightarrow{\langle /a \rangle : \gamma} q_3 \xrightarrow{\langle b \rangle : \gamma} q_4 \xrightarrow{\langle /b \rangle : \gamma} q_5 \xrightarrow{\langle /c \rangle : \gamma} q_6 \longrightarrow$$

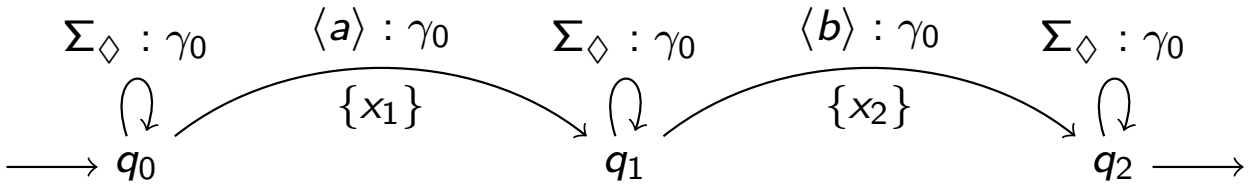


Attributed STAs

$$\longrightarrow q_0 \xrightarrow[l_0]{\langle c \rangle : \gamma} q_1 \xrightarrow[l_1]{\langle a \rangle : \gamma} q_2 \xrightarrow{\langle /a \rangle : \gamma} q_3 \xrightarrow[*]{\langle b \rangle : \gamma} q_4 \xrightarrow{\langle /b \rangle : \gamma} q_5 \xrightarrow{\langle /c \rangle : \gamma} q_6 \longrightarrow$$

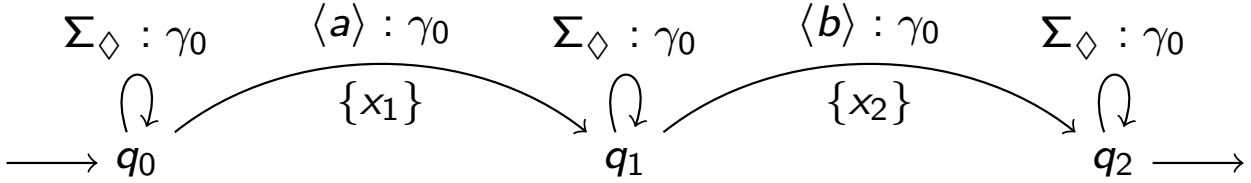


Querying with STA



$\langle c \rangle$	$\langle a \rangle$	$\langle /a \rangle$	$\langle b \rangle$	$\langle /b \rangle$	$\langle a \rangle$	$\langle /a \rangle$	$\langle /c \rangle$
l_0	l_1	–	l_2	–	l_3	–	–

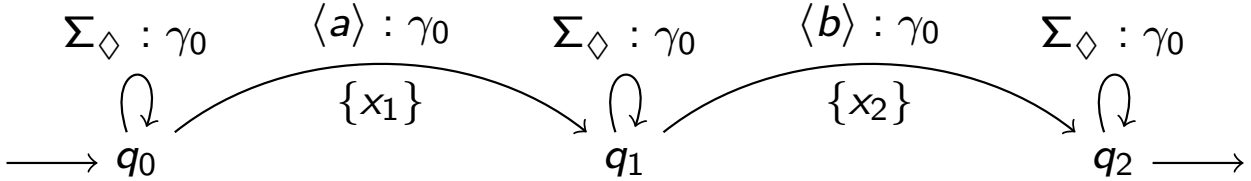
Querying with STA



$\langle c \rangle$	$\langle a \rangle$	$\langle /a \rangle$	$\langle b \rangle$	$\langle /b \rangle$	$\langle a \rangle$	$\langle /a \rangle$	$\langle /c \rangle$
l_0	l_1	–	l_2	–	l_3	–	–

q_0
$(\bot, \bot)$

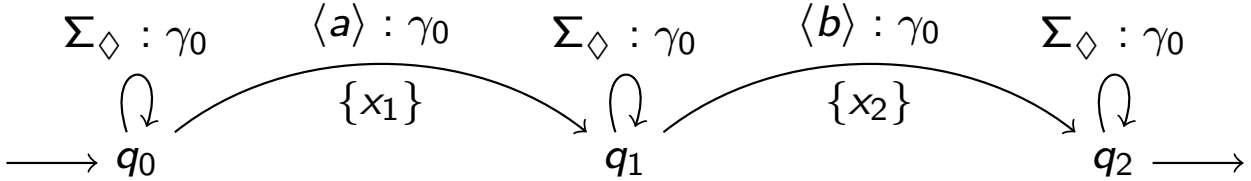
Querying with STA



$\langle c \rangle$	$\langle a \rangle$	$\langle /a \rangle$	$\langle b \rangle$	$\langle /b \rangle$	$\langle a \rangle$	$\langle /a \rangle$	$\langle /c \rangle$
l_0	l_1	–	l_2	–	l_3	–	–

q_0	q_0
$(\bot, \bot)$	$(\bot, \bot)$

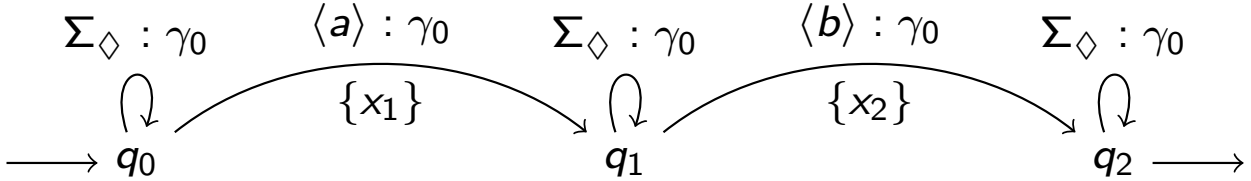
Querying with STA



$\langle c \rangle$	$\langle a \rangle$	$\langle /a \rangle$	$\langle b \rangle$	$\langle /b \rangle$	$\langle a \rangle$	$\langle /a \rangle$	$\langle /c \rangle$
l_0	l_1	–	l_2	–	l_3	–	–

q_0	q_0	q_1
$(\bot, \bot)$	$(\bot, \bot)$	$(l_1, \bot)$

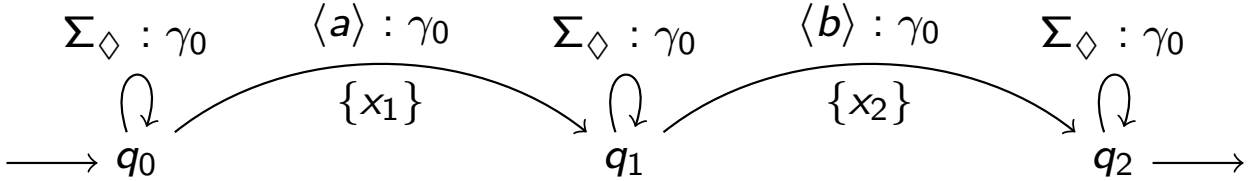
Querying with STA



$\langle c \rangle$	$\langle a \rangle$	$\langle /a \rangle$	$\langle b \rangle$	$\langle /b \rangle$	$\langle a \rangle$	$\langle /a \rangle$	$\langle /c \rangle$
l_0	l_1	–	l_2	–	l_3	–	–

q_0	q_0	q_1	q_1
$(\bot, \bot)$	$(\bot, \bot)$	$(l_1, \bot)$	$(l_1, \bot)$

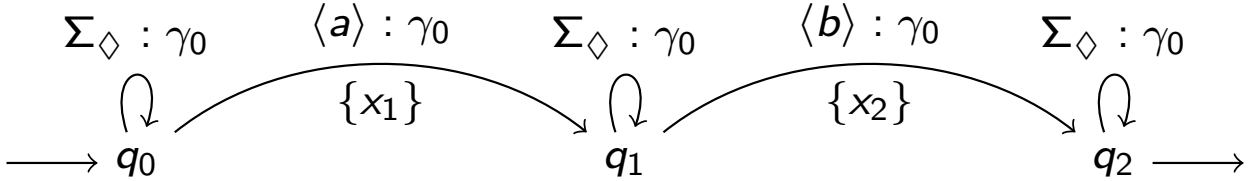
Querying with STA



$\langle c \rangle$	$\langle a \rangle$	$\langle /a \rangle$	$\langle b \rangle$	$\langle /b \rangle$	$\langle a \rangle$	$\langle /a \rangle$	$\langle /c \rangle$
l_0	l_1	–	l_2	–	l_3	–	–

q_0	q_0	q_1	q_1	q_2
$(\bot, \bot)$	$(\bot, \bot)$	$(l_1, \bot)$	$(l_1, \bot)$	(l_1, l_2)

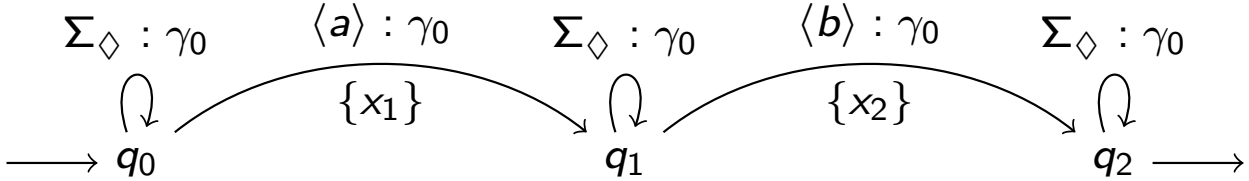
Querying with STA



$\langle c \rangle$	$\langle a \rangle$	$\langle /a \rangle$	$\langle b \rangle$	$\langle /b \rangle$	$\langle a \rangle$	$\langle /a \rangle$	$\langle /c \rangle$
l_0	l_1	–	l_2	–	l_3	–	–

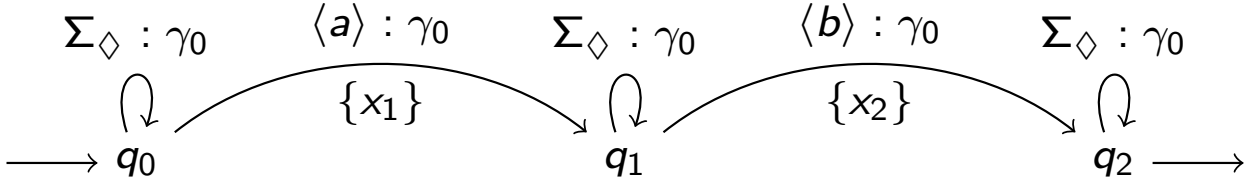
q_0	q_0	q_1	q_1	q_2	q_2
$(\bot, \bot)$	$(\bot, \bot)$	$(l_1, \bot)$	$(l_1, \bot)$	(l_1, l_2)	(l_1, l_2)

Querying with STA



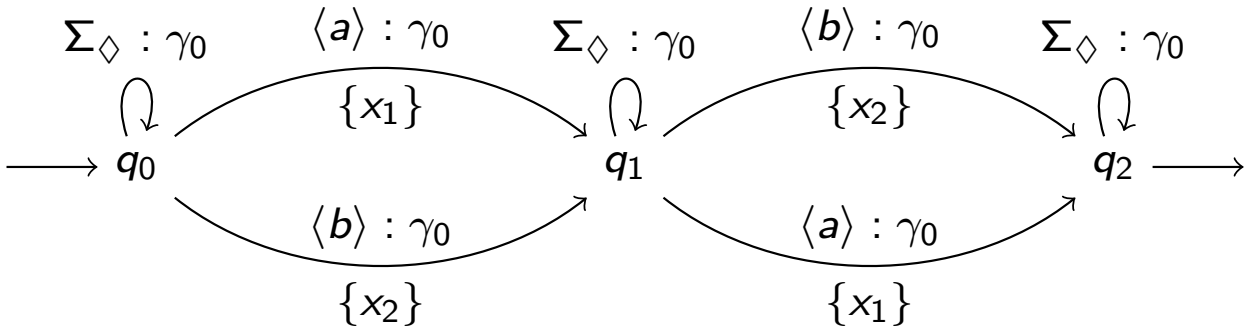
$\langle c \rangle$	$\langle a \rangle$	$\langle /a \rangle$	$\langle b \rangle$	$\langle /b \rangle$	$\langle a \rangle$	$\langle /a \rangle$	$\langle /c \rangle$
l_0	l_1	–	l_2	–	l_3	–	–
q_0	q_0	q_1	q_1	q_2	q_2	q_2	
$(\bot, \bot)$	$(\bot, \bot)$	$(l_1, \bot)$	$(l_1, \bot)$	(l_1, l_2)	(l_1, l_2)	(l_1, l_2)	

Querying with STA



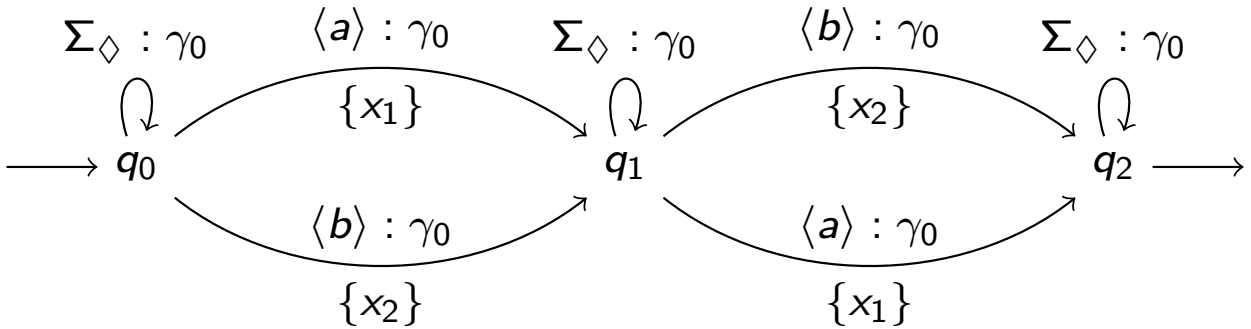
$\langle c \rangle$	$\langle a \rangle$	$\langle /a \rangle$	$\langle b \rangle$	$\langle /b \rangle$	$\langle a \rangle$	$\langle /a \rangle$	$\langle /c \rangle$
l_0	l_1	–	l_2	–	l_3	–	–
q_0	q_0	q_1	q_1	q_2	q_2	q_2	q_2	q_2
$(\bot, \bot)$	$(\bot, \bot)$	$(l_1, \bot)$	$(l_1, \bot)$	(l_1, l_2)	(l_1, l_2)	(l_1, l_2)	(l_1, l_2)	(l_1, l_2)

Querying with STA



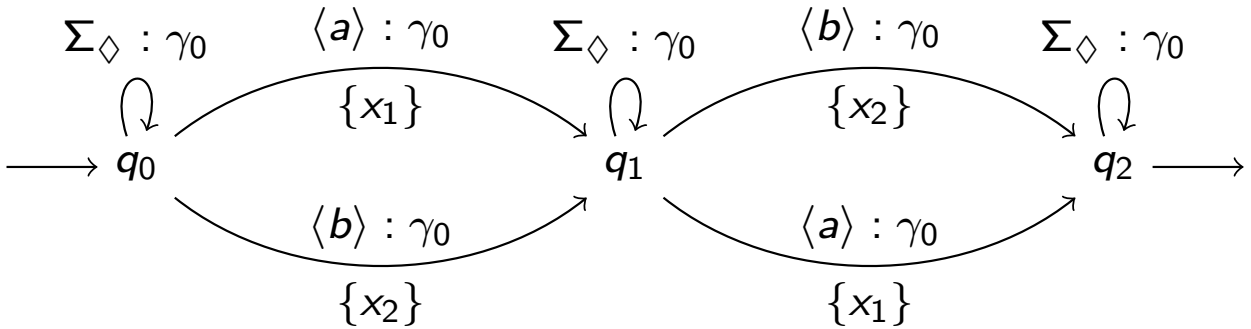
$\langle c \rangle$	$\langle a \rangle$	$\langle /a \rangle$	$\langle b \rangle$	$\langle /b \rangle$	$\langle a \rangle$	$\langle /a \rangle$	$\langle /c \rangle$
l_0	l_1	–	l_2	–	l_3	–	–
q_0	q_0	q_1	q_1	q_2	q_2	q_2	q_2	q_2
$(\bot, \bot)$	$(\bot, \bot)$	$(l_1, \bot)$	$(l_1, \bot)$	(l_1, l_2)	(l_1, l_2)	(l_1, l_2)	(l_1, l_2)	(l_1, l_2)

Querying with STA



$\langle c \rangle$	$\langle a \rangle$	$\langle /a \rangle$	$\langle b \rangle$	$\langle /b \rangle$	$\langle a \rangle$	$\langle /a \rangle$	$\langle /c \rangle$
l_0	l_1	–	l_2	–	l_3	–	–
q_0	q_0	q_1	q_1	q_2	q_2	q_2	q_2	q_2
$(\bot, \bot)$	$(\bot, \bot)$	$(l_1, \bot)$	$(l_1, \bot)$	(l_1, l_2)	(l_1, l_2)	(l_1, l_2)	(l_1, l_2)	(l_1, l_2)
q_0	q_0	q_0	q_0	q_1	q_1	q_2	q_2	q_2
$(\bot, \bot)$	$(\bot, \bot)$	$(\bot, \bot)$	$(\bot, \bot)$	$(\bot, l_2)$	$(\bot, l_2)$	(l_3, l_2)	(l_3, l_2)	(l_3, l_2)

Querying with STA



$\langle c \rangle$	$\langle a \rangle$	$\langle /a \rangle$	$\langle b \rangle$	$\langle /b \rangle$	$\langle a \rangle$	$\langle /a \rangle$	$\langle /c \rangle$
l_0	l_1	–	l_2	–	l_3	–	–
q_0	q_0	q_1	q_1	q_2	q_2	q_2	q_2	q_2
$(\bot, \bot)$	$(\bot, \bot)$	$(l_1, \bot)$	$(l_1, \bot)$	(l_1, l_2)	(l_1, l_2)	(l_1, l_2)	(l_1, l_2)	(l_1, l_2)
q_0	q_0	q_0	q_0	q_1	q_1	q_2	q_2	q_2
$(\bot, \bot)$	$(\bot, \bot)$	$(\bot, \bot)$	$(\bot, \bot)$	$(\bot, l_2)$	$(\bot, l_2)$	(l_3, l_2)	(l_3, l_2)	(l_3, l_2)
q_0	q_0	q_1	q_1	q_1	q_1	q_2	No! :-(	
$(\bot, \bot)$	$(\bot, \bot)$	$(l_1, \bot)$	$(l_1, \bot)$	$(l_1, \bot)$	$(l_1, \bot)$	$(l_3, \bot)$		

Automata, automata, ...

Streaming Tree Automata (STA) ($M, N, \ldots$)

- Specify structure of the document (expressibility=MSO)
- Transitions: $q \xrightarrow{\langle a \rangle : \gamma} p$ and $q \xrightarrow{\langle /a \rangle : \gamma} p$

Attributed STAs ($M, N, \ldots$)

- Define sets of trees with attribute values
- Opening transitions: $q \xrightarrow[l]{\langle a \rangle : \gamma} p$, where l can be a wildcard $*$

STA Queries ($\Phi, \Psi, \ldots$)

- Query the document (expressiblity=n-ary MSO queries)
- Opening transitions: $q \xrightarrow[X]{\langle a \rangle : \gamma} p$, where X is a (possibly empty) set of variables

Existential and Universal Query Answers

Query Answers

$QA(\Phi, t)$ tuples (without $\perp$) collected by Φ on t.

A naïve approach to query sets of documents

Querying every element of $L(M) = \{t_1, t_2, \ldots\}$ yields $\{QA(\Phi, t_1), QA(\Phi, t_2), \ldots\}$.
Impractical if the set of documents is large (or infinite).

Existential Query Answers

Answers present in some tree

$$QA^{\exists}(\Phi, M) = \bigcup \{QA(\Phi, t) | t \in L(M)\}$$

Universal Query Answers

Answers present in every tree

$$QA^{\forall}(\Phi, T) = \bigcap \{QA(\Phi, t) | t \in L(M)\}$$

Complexity Analysis

Existential (universal) tuple check

Given an attributed STA M, an n-ary STA query Φ, and tuple $\tau \in (\Lambda \cup \{*\})^n$, find if τ is an existential (resp. universal) answer to Φ in M.

Three complexity measures

1. Combined complexity. All components are part of the input.
2. Data complexity. The query is assumed to be fixed.
3. Fixed parametric complexity. Parametrized by the arity n of the query Φ: the time complexity function can have a coefficient $f(n)$ for some f.

Complexity Analysis

Existential (universal) tuple check

Given an attributed STA M, an n-ary STA query Φ, and tuple $\tau \in (\Lambda \cup \{*\})^n$, find if τ is an existential (resp. universal) answer to Φ in M.

Three complexity measures

1. Combined complexity. All components are part of the input.
2. Data complexity. The query is assumed to be fixed.
3. Fixed parametric complexity. Parametrized by the arity n of the query Φ: the time complexity function can have a coefficient $f(n)$ for some f.

	Existential TC	Universal TC
Combined	NP-complete	EXPTIME-complete
Fixed param.	FPT	intractable
Data	PTIME	PTIME

Consistent Query Answers for XML

Edit distance $d(t_1, t_2)$

The minimal cost of transforming t_1 into t_2 by performing standard editing operations: *renaming*, *insertion*, and *deletion*.

Repair of t w.r.t. M

A tree satisfying M minimally different from t, i.e.

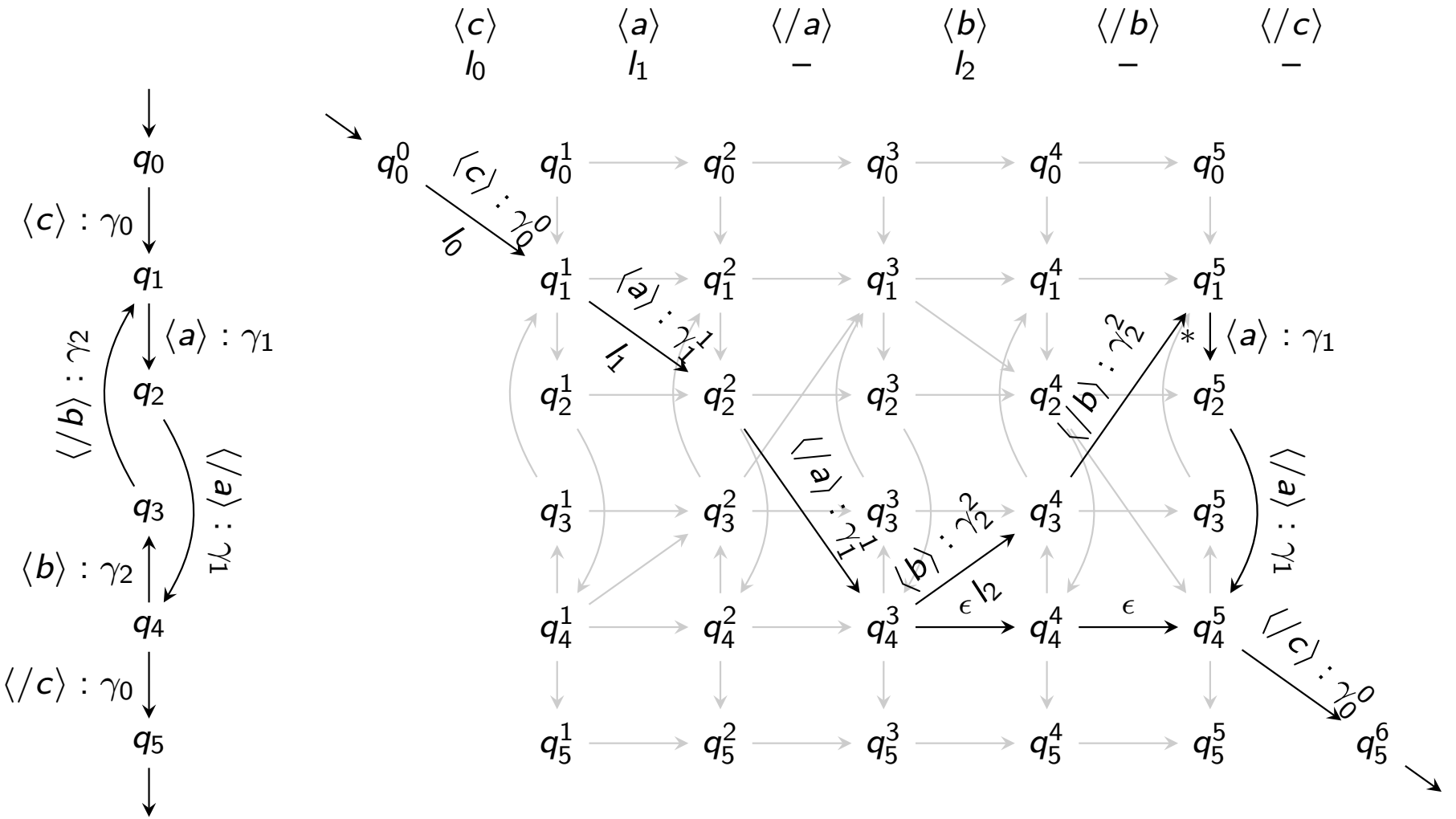
$$Rep(t, M) = \{t' | d(t, t') = \min_{t'' \in L(M)} d(t, t'')\}.$$

Consistent answers to Φ in t w.r.t. M

Answers present in every repair of t. w.r.t. M, i.e.

$$CQA(\Phi, t, M) = \bigcap_{t' \in Rep(t,M)} QA(\Phi, t) = QA^{\forall}(\Phi, Rep(t, M)).$$

Repair Automaton



Computing consistent query answers

Repair automata

Defines the set of all repairs $L(R(t, M)) = Rep(T, M)$.

Consistent query answers

Expressed with universal query answers: $CQA(\Phi, I, M) = QA^{\forall}(\Phi, R(t, M))$

Data complexity

The data complexity of computing consistent query answers is PTIME.

Combined complexity

The combined complexity of computing consistent query answers is Π_p^2-complete if the cost of insertion $w_I > 0$. EXPTIME-complete if $w_I = 0$.

Conclusions

Framework for querying regular sets of XML Documents

- Unifies novel XML database applications
- Based on automata
- Complexity analysis

Consistent Query Answers

- Broader class of schemata: EDTD as compared to DTD
- More general editing operations: working on all nodes not only leaves
- Broader class of queries: n-ary MSO as compared to unary FO_2

Future Work

- Find other applications that can be handled by this framework
  - Data exchange
  - Querying nondeterministic document transformations
  - Validation of access policies
- Improve complexity
  - Automata model enjoying *nicer* pumping properties
  - Restricted classes of queries
- Context-free grammars to represent sets of documents
  - M. Frick, M. Grohe, and C. Koch, *Query Evaluation on Compressed Trees*, LICS 2003
- Automata with value comparison (for querying and constrains in the set of documents)
  - Queries with joins, functional and inclusion dependencies
  - Likely to render things undecidable (M. Bojanczyk et al. PODS 2006)