Learning Schemas for Unordered XML

Radu Ciucanu Sławek Staworko

University of Lille & INRIA, France

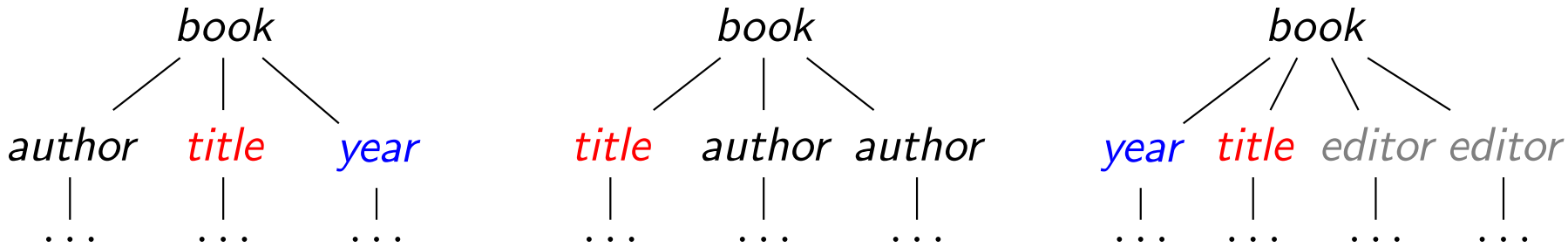
DBPL'13
August 30, 2013

Table of contents

Learning Schemas for Unordered XML

Unordered XML - the relative order of elements is not relevant.

XML is used for **data-centric** applications (as a semi-structured database).



Real world DTD.

$$book \rightarrow (title \mid year \mid author \mid editor)^*.$$

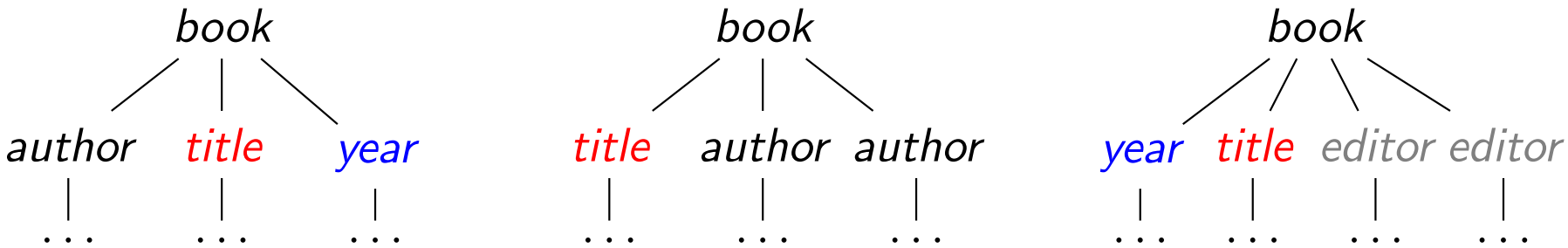
Drawback: over-permissiveness.

- a *book* with many *title*'s or without a *title*.
- a *book* having at the same time *author*'s and *editor*'s...

Learning Schemas for Unordered XML

Unordered XML - the relative order of elements is not relevant.

XML is used for **data-centric** applications (as a semi-structured database).



A schema (DMS) satisfied by the three documents.

$$book \rightarrow title \parallel year^{?} \parallel (author^{+} \mid editor^{+}).$$

Why learning unordered schemas?

- If the schema of a XML collection is incorrect or missing, an inferred schema would allow:
  - **Query** or **modify** the collection by the users,
  - Twig query **minimization** – given a query and a schema, find a smaller yet equivalent query in the presence of the schema,
  - **Learn** better twig queries.
- **Data integration** – needs inferred unordered schemas.
- **Schema evolution** – previous valid documents become invalid (negative examples).

"We need to **extract good-quality schemas** automatically from existing data and perform **incremental maintenance** of the generated schemas to fulfill the goal of achieving **schema and data independence**." [Florescu '05]

Related work

- Learning DTDs reduces to learning regular expressions,
- [Bex et al. '10] proposed learning algorithms for practical subclasses of regular expressions:
  - SOREs – single occurrence regular expressions,
  - CHAREs – chain regular expressions,
  - k-ores – each alphabet symbol occurs at most k times.
- DMS can be seen as restricted SOREs under commutative closure,
- The algorithms for learning SOREs take ordered input, therefore an additional input that the DMS do not have (the order among labels),
- Learning DMS cannot be reduced to learning SOREs.

Contributions

- Previously, only ordered XML schemas have been investigated for learning (typically restricted classes of regular expressions). We focus on learning unordered schemas.
- Previously, only positive examples have been considered. We also study settings with positive and negative examples.

Schema formalism	*+ examples only*	*+ and - examples*
DMS	**Yes**	No
MS	**Yes**	**Yes**

Table: Summary of learnability results.

Preliminaries

Unordered words

We define **languages of unordered words** (no order on symbols).

$$aab \equiv aba \equiv baa.$$

An unordered word is a **multiset of symbols**.

$$\{a, a, b\}.$$

Multiplicity expressions

Multiplicities

a^+	$a, aa, \ldots$	d^1	d
b^*	$\varepsilon, b, bb, \ldots$	f^0	ε
$c^?$	ε, c		

Multiplicity expressions

Multiplicities

a^+	$a, aa, \ldots$	d^1	d
b^*	$\varepsilon, b, bb, \ldots$	f^0	ε
$c^?$	ε, c		

- Disjunction-free multiplicity expressions use multiplicities and unordered concatenation ("$\|\|$").

$$E_1 = a^+ \,\|\, b^* \,\|\, c^? \,\|\, d$$

$b\,d\,a$ ✓ $\qquad$ $a\,b\,c$ ✗ $\qquad$ $d\,d$ ✗

Multiplicity expressions

Multiplicities

a^+	$a, aa, \ldots$	d^1	d
b^*	$\varepsilon, b, bb, \ldots$	f^0	ε
$c^?$	ε, c		

- Disjunction-free multiplicity expressions use multiplicities and unordered concatenation ("$\|\|$").

$$E_1 = a^+ \,\|\, b^* \,\|\, c^? \,\|\, d$$

$b\,d\,a$ ✓ $\qquad$ $a\,b\,c$ ✗ $\qquad$ $d\,d$ ✗

- Disjunctive multiplicity expressions additionally use disjunction ("$|$").

$$E_2 = (a \mid b)^+ \,\|\, (c^? \mid d^*)$$

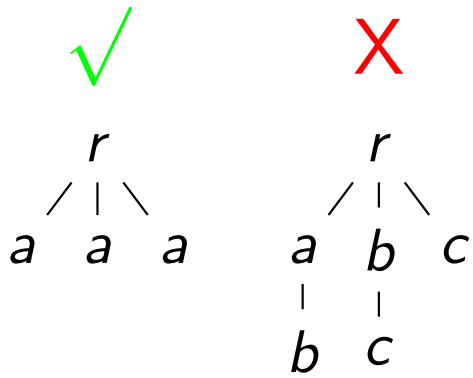
$a\,c\,d$ ✗ $\qquad$ $b\,d\,a$ ✓ $\qquad$ $d\,d$ ✗

Multiplicity schemas

MS Disjunction-free multiplicity schema

A set of disjunction-free multiplicity expressions

$$
\begin{array}{lcl}
r & \rightarrow & a^{+} \parallel b^{*} \parallel c^{?} \\
a & \rightarrow & b^{?} \parallel c^{*} \\
b & \rightarrow & a^{?} \parallel d
\end{array}
$$

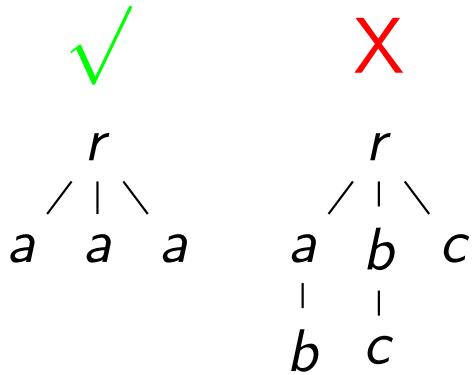


Multiplicity schemas

MS Disjunction-free multiplicity schema

A set of disjunction-free multiplicity expressions

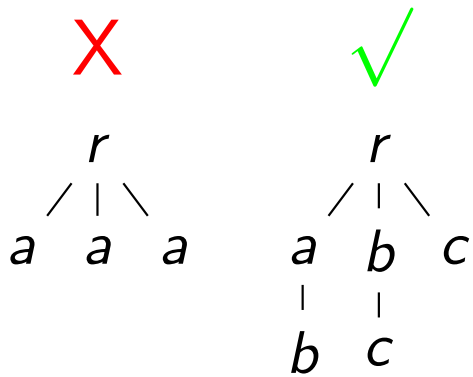
$$
\begin{array}{lcl}
r & \to & a^{+} \parallel b^{*} \parallel c^{?} \\
a & \to & b^{?} \parallel c^{*} \\
b & \to & a^{?} \parallel d
\end{array}
$$



DMS Disjunctive multiplicity schema

A set of disjunctive multiplicity expressions

$$
\begin{array}{lcl}
r & \to & a^{*} \parallel (b \mid c)^{+} \\
a & \to & b^{?} \parallel c^{*} \\
b & \to & (c^{?} \mid d^{?})
\end{array}
$$



Learning framework

- Identification in the limit [Gold '67] – a good learning algorithm is able to infer any concept from a sufficiently rich set of examples.

- A class of schemas is learnable from positive examples if there is an algorithm *learner* that takes a set of examples, returns a schema, and
  1. *learner* is polynomial,
  2. *learner* is sound i.e., returns a schema consistent with the examples.
  3. *learner* is complete i.e., for every schema S there is a set of examples CS_S s.t. for every D that extends CS_S consistently with S, *learner*(D) returns S. CS_S is called the characteristic sample for S w.r.t. *learner*. The cardinality of CS_S is polynomial in the size of the alphabet.

Size vs cardinality

Take $n > 1$, $\Sigma = \{r, a_1, b_1, \ldots, a_n, b_n\}$, and the DMS S:

$$r \to a_1 \parallel b_1,$$
$$a_i, b_i \to a_{i+1} \parallel b_{i+1} \quad (\text{for } 1 \leq i < n),$$
$$a_n, b_n \to \epsilon.$$

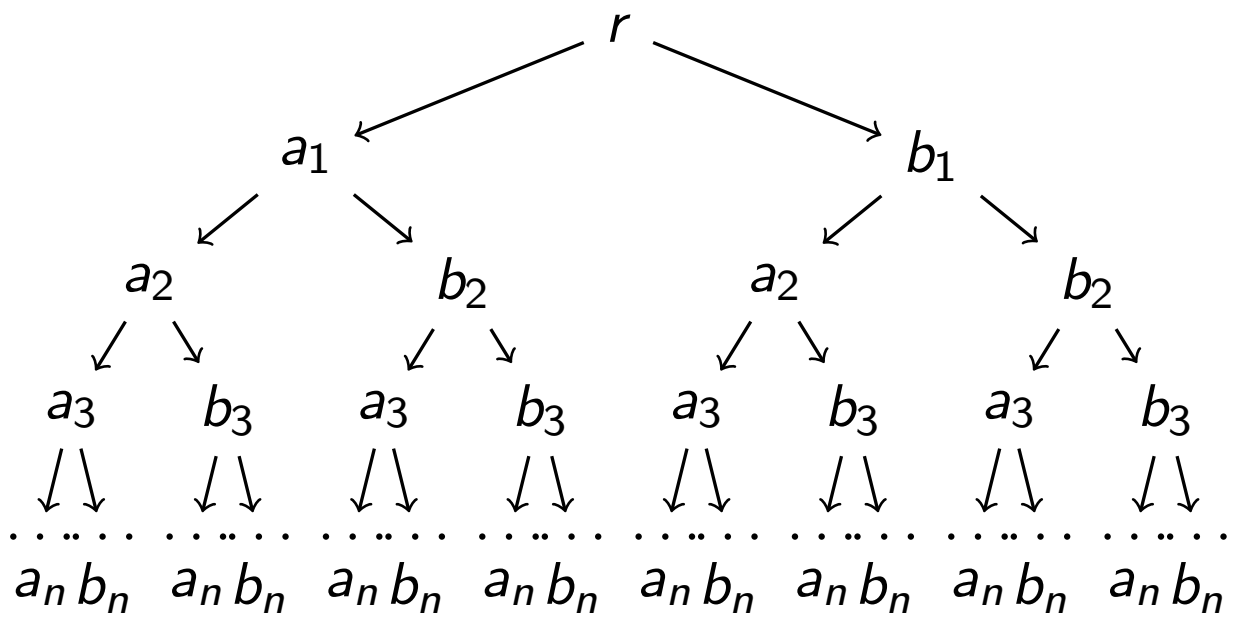
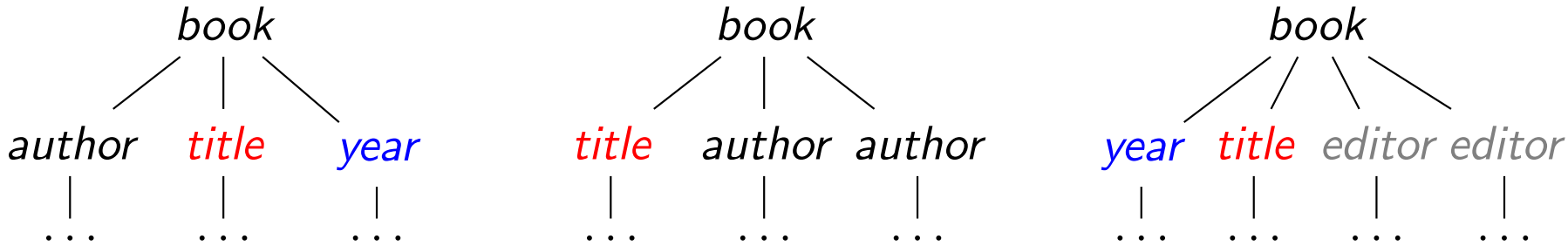


Figure: The unique tree satisfying the schema S.

Minimality

Our algorithms return minimal schemas consistent with the examples.



The minimal (i.e., most specific) consistent schema

$$book \rightarrow title \parallel year^{?} \parallel (author^{+} \mid editor^{+}).$$

Another possible solution (not minimal)

$$book \rightarrow title \parallel year^{?} \parallel author^{*} \parallel editor^{*}.$$

Minimality is perceived as a better fitted learning solution.

Learning from positive examples

Learning DMS reduces to learning DME

- Learning a DTD reduces to learning for each label the corresponding regular expression,
- Similarly, learning a DMS reduces to learning the DME for each label.

Example

Consider $\Sigma = \{a, b, c, d, e\}$, the sample $D = \{aabc, abd, be\}$, and:

$$E_1 = (a^+ \mid e) \parallel b \parallel (c^? \mid d^?),$$
$$E_2 = a^* \parallel b \parallel (c \mid d \mid e).$$

- $D \subseteq L(E_1)$,
- $D \subseteq L(E_2)$,
- $L(E_1) \not\subseteq L(E_2)$ (because of bce),
- $L(E_2) \not\subseteq L(E_1)$ (because of abe),
- Both E_1 and E_2 are minimal DMEs with languages including D.

Normal form of the DME (1)

Any DME E can be captured by its **characterizing triple** (C_E, N_E, P_E).
Take $E_0 = a^+ \parallel (b \mid c) \parallel d^?$:

- C_E *Conflicting pairs of siblings* - pairs of symbols which cannot be present in a word at the same time. $C_{E_0} = \{(b, c), (c, b)\}$
- N_E *Cardinality map* - all the numbers of occurrences for every symbol. $N_{E_0} = \{(b, 0), (b, 1), (c, 0), (c, 1), (d, 0), (d, 1), (a, 1), (a, 2), \ldots\}$
- P_E *Sets of required symbols* - at least one of them should be present in any word. $P_{E_0} = \{\{a\}, \{b, c\}, \ldots\}$

Normal form of the DME (1)

Any DME E can be captured by its **characterizing triple** (C_E, N_E, P_E).
Take $E_0 = a^+ \parallel (b \mid c) \parallel d^?$:

- C_E *Conflicting pairs of siblings* - pairs of symbols which cannot be present in a word at the same time. $C_{E_0} = \{(b,c),(c,b)\}$
- N_E *Cardinality map* - all the numbers of occurrences for every symbol. $N_{E_0} = \{(b,0),(b,1),(c,0),(c,1),(d,0),(d,1),(a,1),(a,2),\ldots\}$
- P_E *Sets of required symbols* - at least one of them should be present in any word. $P_{E_0} = \{\{a\},\{b,c\},\ldots\}$

A normal form has polynomial representation:

- C_E^* $C^*_{E_0} = \{\{b,c\}\}$
- N_E^* $N^*_{E_0}(a) = +, \quad N^*_{E_0}(b) = N^*_{E_0}(c) = N^*_{E_0}(d) = ?$
- P_E^* $P^*_{E_0} = \{\{a\},\{b,c\}\}$

Normal form of the DME (2)

We can easily construct a DME from its characterizing triple. For example:

$$C^*_{E_1} = \{\{a, e\}, \{c, d\}\},$$
$$P^*_{E_1} = \{\{a, e\}, \{b\}\},$$
$$N^*_{E_1}(a) = *, \quad N^*_{E_1}(b) = 1, \quad N^*_{E_1}(c) = N^*_{E_1}(d) = N^*_{E_1}(e) = ?.$$

Note that they characterize the expression:

$$E_1 = (a^+ \mid e) \parallel b \parallel (c^? \mid d^?).$$

Algorithm for learning a DME from positive examples

We illustrate on the following sample over $\Sigma = \{a, b, c, d, e\}$:

$$D = \{aabc, abd, be\}.$$

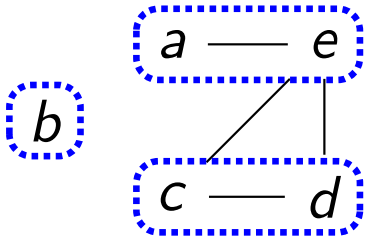
1. Infer cardinality map: $N_E^*(a) = *,\ N_E^*(b) = 1,\ N_E^*(c/d/e) = ?$

Algorithm for learning a DME from positive examples

We illustrate on the following sample over $\Sigma = \{a, b, c, d, e\}$:

$$D = \{aabc, abd, be\}.$$

1. Infer cardinality map: $N_E^*(a) = *,\ N_E^*(b) = 1,\ N_E^*(c/d/e) = ?$
2. Infer conflicting siblings (max-clique partition): $C_E^* = \{\{a, e\}, \{c, d\}\}$

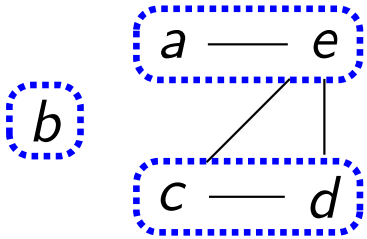


Algorithm for learning a DME from positive examples

We illustrate on the following sample over $\Sigma = \{a, b, c, d, e\}$:

$$D = \{aabc, abd, be\}.$$

1. Infer cardinality map: $N_E^*(a) = *,\ N_E^*(b) = 1,\ N_E^*(c/d/e) = ?$
2. Infer conflicting siblings (max-clique partition): $C_E^* = \{\{a, e\}, \{c, d\}\}$



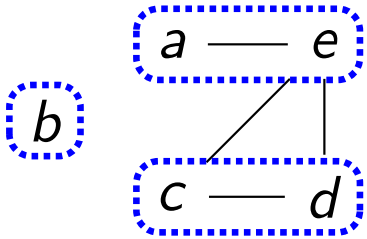
3. Infer sets of required symbols: $P_E^* = \{\{a, e\}, \{b\}\}$

Algorithm for learning a DME from positive examples

We illustrate on the following sample over $\Sigma = \{a, b, c, d, e\}$:

$$D = \{aabc, abd, be\}.$$

1. Infer cardinality map: $N_E^*(a) = *,\ N_E^*(b) = 1,\ N_E^*(c/d/e) = ?$
2. Infer conflicting siblings (max-clique partition): $C_E^* = \{\{a, e\}, \{c, d\}\}$



3. Infer sets of required symbols: $P_E^* = \{\{a, e\}, \{b\}\}$
4. Construct the corresponding DME: $E = (a^+ \mid e) \parallel b \parallel (c^? \mid d^?)$

Theorem

DMS and MS are learnable from positive examples.

Characteristic sample for learning DMS

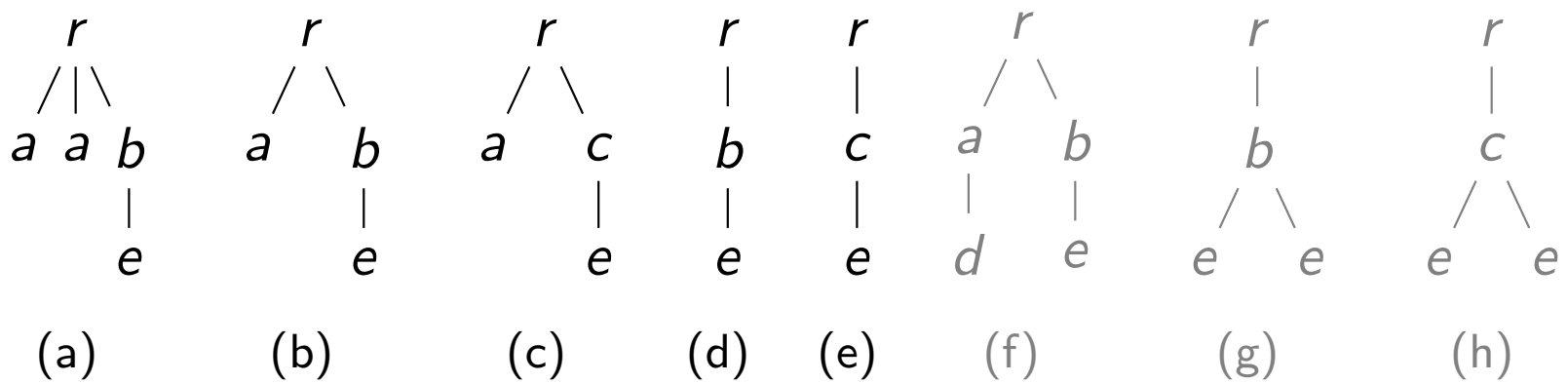
Take the following schema S over $\Sigma = \{a, b, c, d, e\}$:

$$r \to a^* \parallel (b \mid c), \qquad a \to d^?, \qquad b, c \to e^+, \qquad d, e \to \epsilon.$$

We construct the characteristic sample for each DME from the rules of S:

- $CS_{R_S(r)} = \{aab, ab, ac, b, c\}$,
- $CS_{R_S(a)} = \{\varepsilon, d\}$,
- $CS_{R_S(b)} = CS_{R_S(c)} = \{e, ee\}$,
- $CS_{R_S(d)} = CS_{R_S(e)} = \{\varepsilon\}$.

We construct the characteristic sample for S:



Impact of negative examples

Consistency checking

A sound algorithm needs to return *null* iff there is no concept consistent with the input.

Consistency checking

Given a set of positive and negative examples, decide whether there exists a concept consistent with the input sample.

Consistency checking

A sound algorithm needs to return *null* iff there is no concept consistent with the input.

Consistency checking

Given a set of positive and negative examples, decide whether there exists a concept consistent with the input sample.

Given a set of positive examples, there is an unique minimal consistent MS.

Example: $D^+ = \{aabc, abd, be\}$

- The minimal ME consistent with D^+ is $E = a^* \parallel b \parallel c^? \parallel d^? \parallel e^?$,
- If $D^- = \{abbe\}$, the sample is consistent,
- If $D^- = \{abe\}$, the sample is **not** consistent.

Consistency checking

A sound algorithm needs to return *null* iff there is no concept consistent with the input.

Consistency checking

Given a set of positive and negative examples, decide whether there exists a concept consistent with the input sample.

Given a set of positive examples, there is an unique minimal consistent MS.

Example: $D^+ = \{aabc, abd, be\}$

- The minimal ME consistent with D^+ is $E = a^* \parallel b \parallel c^? \parallel d^? \parallel e^?$,
- If $D^- = \{abbe\}$, the sample is consistent,
- If $D^- = \{abe\}$, the sample is **not** consistent.

Theorem

1. Consistency checking for MS is in PTIME,
2. MS are learnable from both positive and negative examples.

Consistency checking

A sound algorithm needs to return *null* iff there is no concept consistent with the input.

Consistency checking

Given a set of positive and negative examples, decide whether there exists a concept consistent with the input sample.

Given a set of positive examples, there may be many minimal consistent DMS.

Example: $D^+ = \{aabc, abd, be\}$ and $D^- = \{abe\}$

There are two minimal DME consistent with D^+:

- $E_1 = a^* \parallel b \parallel (c \mid d \mid e)$, but $abe \in L(E_1)$,
- $E_2 = (a^+ \mid e) \parallel b \parallel (c^? \mid d^?)$, $abe \notin L(E_2)$, so the sample is consistent.

In fact, there may exist exponentially many minimal consistent DMS.

Consistency checking for DME is NP-complete

For $\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_3 \vee \neg x_4)$, take the sample:

$$
\begin{array}{ll}
(t_1 f_1 t_2 f_2 t_3 f_3 t_4 f_4, +), & (\varepsilon, -), \\
(t_1 f_1, +), & (t_1 t_1 f_1 f_1, -), \\
(t_2 f_2, +), & (t_2 t_2 f_2 f_2, -), \\
(t_3 f_3, +), & (t_3 t_3 f_3 f_3, -), \\
(t_4 f_4, +), & (t_4 t_4 f_4 f_4, -), \\
 & (f_1 f_1 t_2 t_2 f_3 f_3, -), \\
 & (t_1 t_1 f_3 f_3 t_4 t_4, -).
\end{array}
$$

A consistent DME is $E_V = (t_1 \mid f_2 \mid t_3 \mid f_4)^+ \parallel f_1^? \parallel t_2^? \parallel f_3^? \parallel t_4^?$.
Note that a valuation $V \models \varphi$ iff E_V is consistent with the sample.

Consistency checking for DME is NP-complete

For $\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_3 \vee \neg x_4)$, take the sample:

$$(t_1 f_1 t_2 f_2 t_3 f_3 t_4 f_4, +), \quad (\varepsilon, -),$$
$$(t_1 f_1, +), \quad (t_1 t_1 f_1 f_1, -),$$
$$(t_2 f_2, +), \quad (t_2 t_2 f_2 f_2, -),$$
$$(t_3 f_3, +), \quad (t_3 t_3 f_3 f_3, -),$$
$$(t_4 f_4, +), \quad (t_4 t_4 f_4 f_4, -),$$
$$(f_1 f_1 t_2 t_2 f_3 f_3, -),$$
$$(t_1 t_1 f_3 f_3 t_4 t_4, -).$$

A consistent DME is $E_V = (t_1 \mid f_2 \mid t_3 \mid f_4)^+ \parallel f_1^? \parallel t_2^? \parallel f_3^? \parallel t_4^?$.
Note that a valuation $V \models \varphi$ iff E_V is consistent with the sample.

Theorem

1. *Consistency checking for DMS is NP-complete,*
2. *DMS are not learnable from both positive and negative examples.*

Conclusions and future work

Conclusions

- We have studied the problem of learning unordered XML schemas from examples given by the user,
- We have investigated the impact of negative examples.

Schema formalism	*+ examples only*	*+ and - examples*
DMS	**Yes**	No
MS	**Yes**	**Yes**

Table: Summary of learnability results.

- For the learnable cases, we have proposed learning algorithms that return a minimal schema consistent with the examples,
- We have shown the intractable case by proving the intractability of the consistency checking.

Future work

- Characteristic sample of polynomial size (instead of cardinality):
  - Use compressed representation of the XML documents with DAGs.
- Boost the existing learning algorithms for twig queries:
  - Investigate the problem of query minimization in the presence of DMS,
  - Propose a twig learning algorithm which takes into account the schema.
- Extend the learning algorithms for more expressive unordered schemas e.g., which allow numeric occurrences of the form $a^{[n,m]}$.