Rewriting of Queries and Updates across XML Security Views

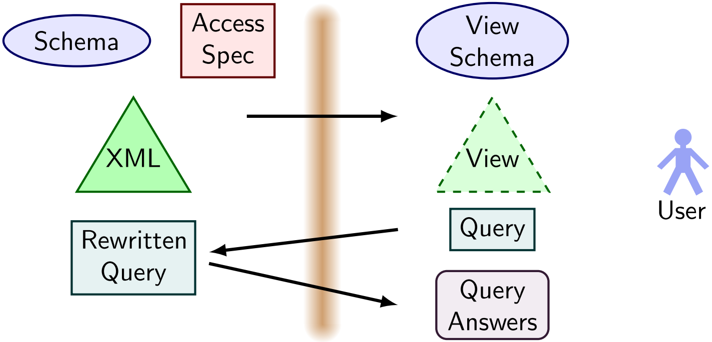
Sławek Staworko
joint work with
Iovka Boneva, Benoît Groz, Anne-Cécile Caron, Yves Roos, and Sophie Tison

Mostrare Project
INRIA Lille - Nord Europe
University of Lille

Dagstuhl Seminar: Security and Rewriting

August 2011

View Based XML Security Framework



- the view schema is derived from Access Spec. and Schema
- the view is virtual (no materialization)
- user queries are rewritten and then evaluated

Overview

This talk includes:

1. Framework (XML, Regular XPath, DTDs)
2. Security Access Specification (SAS) and views
3. Query rewriting
4. Static analysis of SAS
5. Updates and their propagations

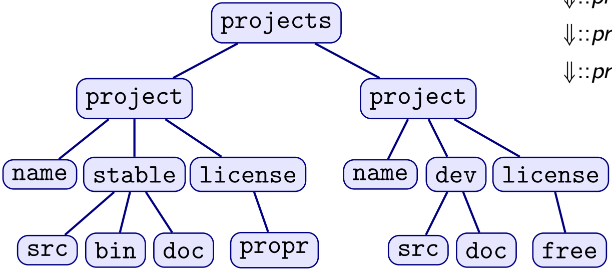
Related to:

- "*XML Security Views Revisited*," DBPL'09
- "*The View Update Problem for XML*," Workshop on XML Updates'10
- "*View Update Translation for XML*," ICDT'11
- "*Static Analysis of XML Security Views and Query Rewriting*," Information & Computation, submitted.

N.B.: Framework introduced and investigated to some extent before.

Basic Notions

XML and XPath



$\Downarrow::project/\Downarrow::name$

$\Downarrow::project[\Downarrow::stable]/\Downarrow::name$

$\Downarrow::project/\Downarrow::*/\Downarrow::src$

Regular XPath ($\mathcal{X}Reg$)

$$\alpha ::\equiv \text{self} \mid \Downarrow \mid \Uparrow \mid \Rightarrow \mid \Leftarrow$$

$$f ::\equiv \text{lab}() = a \mid Q \mid \text{true} \mid \text{not}\, f \mid f \text{ and } f$$

$$Q ::\equiv \alpha \mid [f] \mid Q/Q \mid Q \cup Q \mid Q^*$$

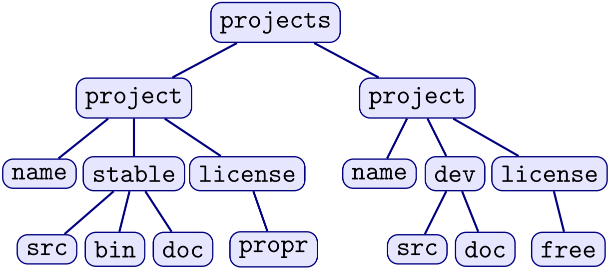
$$\alpha^+ \coloneqq \alpha^*/\alpha$$

$$\alpha::a \coloneqq \alpha[\text{lab}() = a]$$

$$\alpha::* \coloneqq \alpha$$

$$Q[f] \coloneqq Q/[f]$$

SAS = DTD + Annotation



DTD

$projects \rightarrow projects^*$

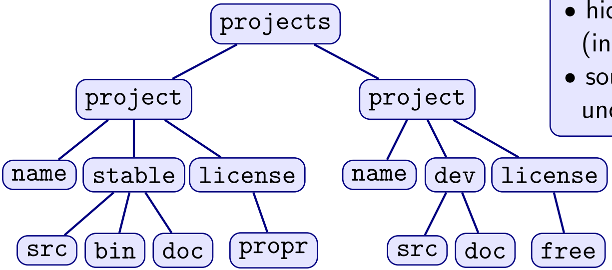
$project \rightarrow name, (stable \mid dev), license$

$stable \rightarrow src, bin, doc$

$dev \rightarrow src, doc$

$license \rightarrow free \mid propr$

SAS = DTD + Annotation



- hide status of project (including binaries)
- source visible only if under free license

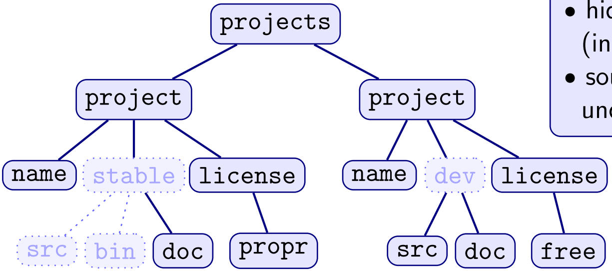
DTD

$projects \rightarrow projects^{*}$
$project \rightarrow name, (stable \mid dev), license$
$stable \rightarrow src, bin, doc$
$dev \rightarrow src, doc$
$license \rightarrow free \mid propr$

Annotation

$A(stable) = \text{false}$
$A(dev) = \text{false}$
$A(doc) = \text{true}$
$A(src) = [\Uparrow::*/\Rightarrow::license/\Downarrow::free]$

SAS = DTD + Annotation



- hide status of project (including binaries)
- source visible only if under free license

Accessibility:

- root always accessible
- if A defined for the node label, then evaluate the filter
- otherwise, accessibility inherited from the parent

View: A(t) = tree obtained from accessible nodes only.

Annotation

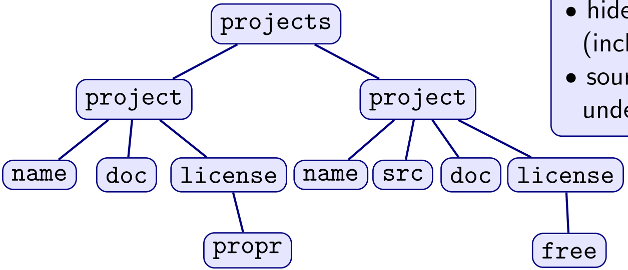
$A(\mathit{stable}) = \text{false}$

$A(\mathit{dev}) = \text{false}$

$A(\mathit{doc}) = \text{true}$

$A(\mathit{src}) = [\Uparrow{::}*/\Rightarrow{::}\mathit{license}/\Downarrow{::}\mathit{free}]$

SAS = DTD + Annotation



- hide status of project (including binaries)
- source visible only if under free license

Accessibility:

- root always accessible
- if A defined for the node label, then evaluate the filter
- otherwise, accessibility inherited from the parent

View: A(t) = tree obtained from accessible nodes only.

Annotation

$A(\mathit{stable}) = \text{false}$

$A(\mathit{dev}) = \text{false}$

$A(\mathit{doc}) = \text{true}$

$A(\mathit{src}) = [\Uparrow{::}{*}/{\Rightarrow}{::}\mathit{license}/{\Downarrow}{::}\mathit{free}]$

Query Rewriting

Query Rewriting

Problem statement

Given:
- source DTD D_S
- annotation A

Input: Regular XPath query Q

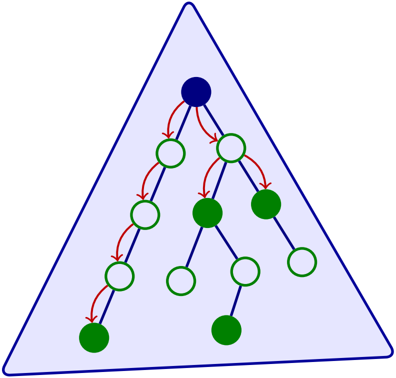
Output: Regular XPath query $Q' = \text{Rewrite}(Q)$ such that for every $t \in L(D_S)$ we have $Q'(t) = Q(A(t'))$

Lemma 1

For any annotation A there exists a filter expression f_{acc} such that

$$\text{a node } n \text{ of a tree } t \text{ is accessible w.r.t. } A \iff (t, n) \models f_{\text{acc}}$$

Query Rewriting: Vertical axes

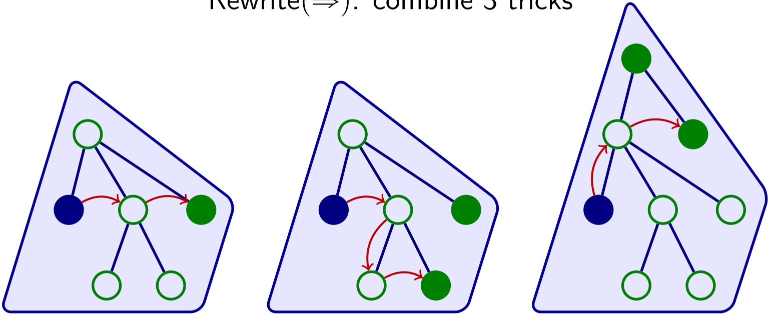


- current node
- accessible node (f_{acc})
- inaccessible node (not f_{acc})

$$\text{Rewrite}(\Downarrow) \coloneqq [f_{acc}]/\Downarrow/\big([\text{not } f_{acc}]/\Downarrow\big)^*/[f_{acc}]$$

Query Rewriting: Horizontal axes

Rewrite(⇒): combine 3 tricks



Query Rewriting: Summary

Theorem

Regular XPath is closed under rewriting over XML views.
The size of the rewritten query is $O(|A| * |Q|)$, where Q is the original query.

Theorem

Boolean and Unary MSO (expressed with tree automata) are closed under rewriting over XML views. Rewriting is polynomial.

Elements of Static Analysis

Static Analysis of SAS: What for?

Scenario: SAS Optimization

- Annotations are replaced with their streamlined versions
- Is the new SAS equivalent to the previous one?

Scenario: SAS Refinement

- SAS is changed to further restrict the access to the document.
- Is the new SAS strictly more restrictive than the previous one?
  Here, **more restrictive** may mean:
  1. Fewer nodes of the source document are visible
  2. Fewer queries can be executed on the source document
  3. Fewer information can be inferred about the source document

Node-based comparison

Equivalence

$$A_1 \equiv^D A_2 \iff \forall t \in L(D).\, Nodes(A_1(t)) = Nodes(A_2(t))$$
$$\iff \forall t \in L(D).\, A_1(t) = A_2(t)$$

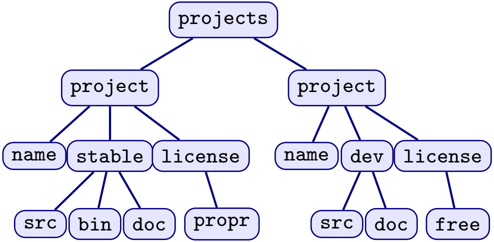
Node-based restriction

$$A_1 \preccurlyeq^D_{NB} A_2 \iff \forall t \in L(D).\, Nodes(A_1(t)) \subseteq Nodes(A_2(t))$$

Theorem

Testing equivalence and node-based restriction is EXPTIME-complete.

Node-based restriction too naïve?



Original annotation

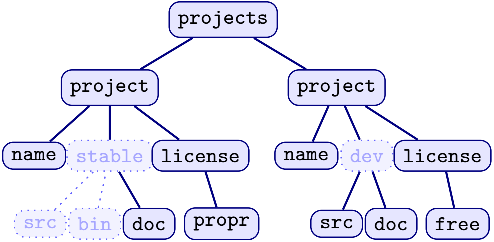
$A_1(\textit{stable}) = \text{false}$

$A_1(\textit{dev}) = \text{false}$

$A_1(\textit{doc}) = \text{true}$

$A_1(\textit{src}) = [\Uparrow::*/\Rightarrow::\textit{license}/\Downarrow::\textit{free}]$

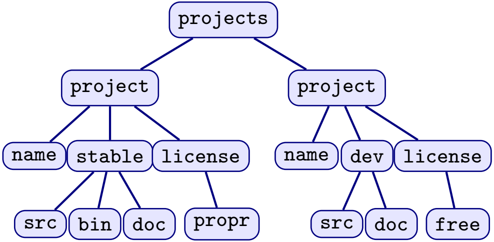
Node-based restriction too naïve?



Original annotation

$A_1(\mathit{stable}) = \text{false}$
$A_1(\mathit{dev}) = \text{false}$
$A_1(\mathit{doc}) = \text{true}$
$A_1(\mathit{src}) = [\Uparrow::*/\Rightarrow::\mathit{license}/\Downarrow::\mathit{free}]$

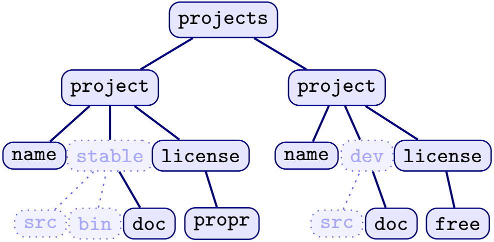
Node-based restriction too naïve?



Original annotation
$A_1(\mathit{stable}) = \text{false}$
$A_1(\mathit{dev}) = \text{false}$
$A_1(\mathit{doc}) = \text{true}$
$A_1(\mathit{src}) = [\Uparrow::*/\Rightarrow::\mathit{license}/\Downarrow::\mathit{free}]$

New annotation (hide sources of projects under development)
$A_2(\mathit{stable}) = \text{false}$
$A_2(\mathit{dev}) = \text{false}$
$A_2(\mathit{doc}) = \text{true}$
$A(\mathit{src}) = [\Uparrow::\mathit{stable}/\Rightarrow::\mathit{license}/\Downarrow::\mathit{free}]$

Node-based restriction too naïve?



Original annotation

$A_1(\textit{stable}) = \text{false}$

$A_1(\textit{dev}) = \text{false}$

$A_1(\textit{doc}) = \text{true}$

$A_1(\textit{src}) = [\Uparrow::*/\Rightarrow::\textit{license}/\Downarrow::\textit{free}]$

New annotation (hide sources of projects under development)

$A_2(\textit{stable}) = \text{false}$

$A_2(\textit{dev}) = \text{false}$

$A_2(\textit{doc}) = \text{true}$

$A(\textit{src}) = [\Uparrow::\textit{stable}/\Rightarrow::\textit{license}/\Downarrow::\textit{free}]$

With the new annotation, the query

$$\Downarrow::\textit{project}[\text{not}(\Downarrow::\textit{src}) \text{ and } \Downarrow::\textit{license}/\Downarrow::\textit{free}]$$

identifies a subset of projects under development which could not be selected before!

Query-based comparison

Identify queries executable on the source

$$Public(D, A) = \{Q \mid \exists Q'.\, \mathrm{Rewrite}(Q', A) \equiv^{D} Q\}$$

Definition (Query-based restriction)

$$A_1 \preccurlyeq^{D}_{QB} A_2 \iff Public(D, A_1) \subseteq Public(D, A_2)$$

Negative results

Testing query-based restriction is **undecidable**.

Positive results

Testing query-based restriction for non-recursive DTDs is in EXPTIME and is PSPACE-hard.

Information-based restriction

What an attacker may suspect?

A well-informed attacker knows: the source DTD D, the annotation A, and the view instance t_V. The source document may be any of:

$$Inv(A, D, t_V) = \{t \in L(D) \mid A(t) = t_v\}$$

What information that can the attacker infer?

$$Certain(D, A, t_S) = \{Q \mid \forall t \in Inv(A, D, A(t_S)).\ t \models Q\}$$

Definition (Information-based restriction)

$$A_1 \preccurlyeq^{D}_{IB} A_2 \iff \forall t \in L(D).\ Certain(D, A_1, t) \subseteq Certain(D, A_2, t)$$

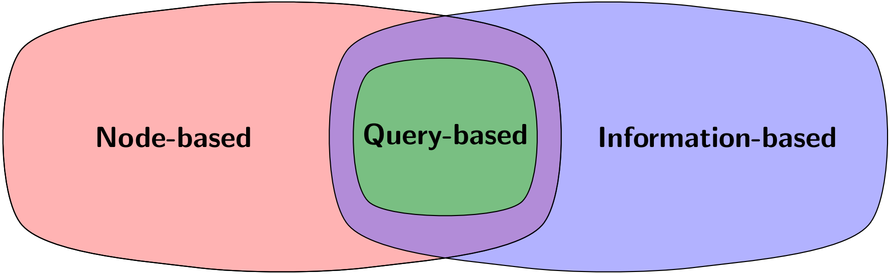
Information-base comparison (cont'd)

Negative results

Testing information-based restriction is **undecidable**.

Positive results

Testing information-based restriction for non-recursive DTDs is in EXPTIME and is PSPACE-hard.



Further results: Interval-bounded SAS

Interval-bounded (IB) SAS

On a descending path in any source document the distance between two consecutive visible nodes is bounded by a fixed constant.

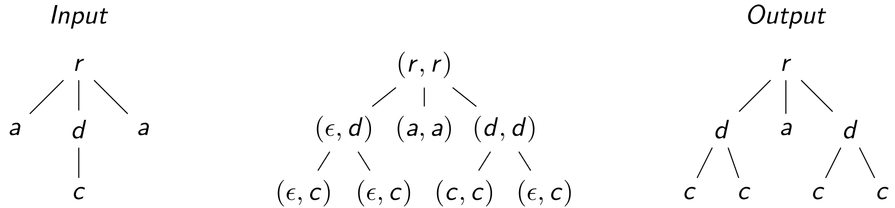
- IB (significantly) generalizes non-recursive DTDs.
- IB pushes the decidability frontier for IB.
- Enables the use of tree automata for a more powerful SAS and more fine-grained comparison of SAS.

Updates and their Rewritings

Alignment trees as Updates

Editing operations

- (ϵ, a) – insert a node
- (a, ϵ) – delete a node
- (a, b) – rename a to b
- (a, a) – do nothing



Editing script

- a tree over $\Sigma \times \{\epsilon\} \cup \Sigma \times \Sigma \cup \{\epsilon\} \times \Sigma$
- downward-closed i.e., delete/insert whole subtrees
- has associated **cost** (number of inserted and deleted nodes)

Update rewriting

View update rewriting (propagation)

Given:
- source DTD D_S
- annotation A (downward-closed)
- view DTD $D_V = A(D_S)$
- source document $t \in L(D_S)$

Input: update of the view $S_o = A(t) \to t_o$ such that $t_o \in L(D_V)$

Output: update of the source document $S = t \to t'$ such that S:
- **side-effect free** i.e., $A(t') = t_o$
- **schema compliant** i.e., $t' \in L(D_S)$
- **optimal** i.e., the cost of S is minimal among all updates of t satisfying the two conditions above

Theorem [Workshop on XML Updates 2010]

An update rewriting can be constructed in polynomial time (DTD is fixed).

Update programs

Update program for a DTD D (cf. XQuery Update Facility)

A set of updates $\mathcal{U}$ that is

- **schema compliant** i.e., $\forall S \in \mathcal{U}$ the input and output of S satisfy D
- **functional** i.e., $\forall t \in L(D)$ there is exactly one $S \in \mathcal{U}$ matching t

$\mathcal{U}$ is **regular** if it is defined with a tree automaton

Constrained update program rewriting

Given: source DTD D_S, annotation A, view schema D_V, a set of allowed updates Ω of the source D_S

Input: view update program $\mathcal{U}_o \subseteq \Omega$

Output: source update program $\mathcal{U}$ such that $\forall t \in L(D_S).\, \mathcal{U}_o(A(t)) = A(\mathcal{U}(t))$

. . . and their rewritings

Unconstrained case (Ω allows all updates)

Rewritings of general (regular) update programs can be easily constructed.

Constrained case

Constrained rewritings of (regular) update programs cannot be constructed.

Synchronized updates

On a descending path in an alignment tree the distance between two consecutive node that are preserved (not deleted nor inserted) is bounded by a constant.

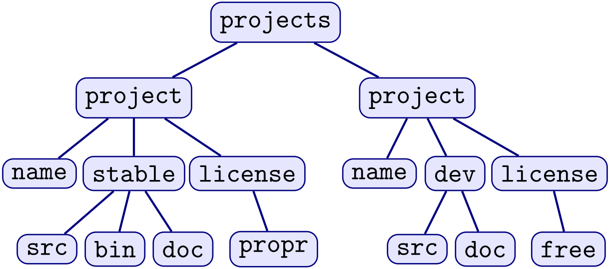
Constrained case

Rewritings of synchronized regular update programs can be constructed.

Thank you

And if there is more time...
Constructing View Schema

Deriving view schema



Annotation

$A(stable) = \text{false}$

$A(dev) = \text{false}$

$A(doc) = \text{true}$

DTD

$projects \rightarrow projects^*$

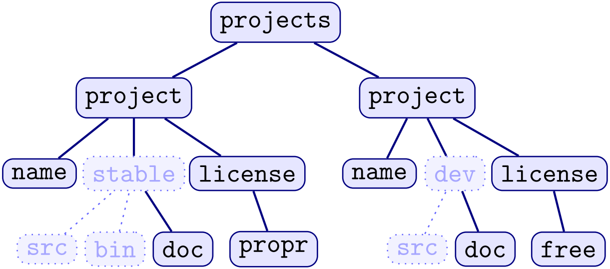
$project \rightarrow name, (stable \mid dev), license$

$stable \rightarrow src, bin, doc$

$dev \rightarrow src, doc$

$license \rightarrow free \mid propr$

Deriving view schema



Annotation

$A(\mathit{stable}) = \text{false}$

$A(\mathit{dev}) = \text{false}$

$A(\mathit{doc}) = \text{true}$

DTD

$\mathit{projects} \to \mathit{projects}^*$

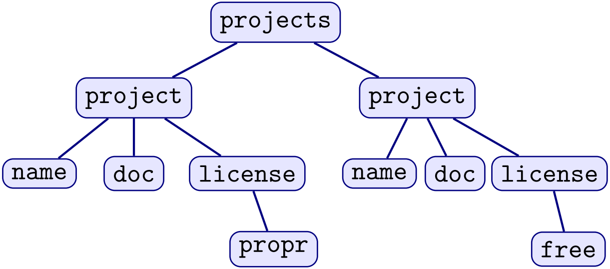
$\mathit{project} \to \mathit{name}, (\mathit{stable} \mid \mathit{dev}), \mathit{license}$

$\mathit{stable} \to \mathit{src}, \mathit{bin}, \mathit{doc}$

$\mathit{dev} \to \mathit{src}, \mathit{doc}$

$\mathit{license} \to \mathit{free} \mid \mathit{propr}$

Deriving view schema



Annotation

$A(stable) = \text{false}$

$A(dev) = \text{false}$

$A(doc) = \text{true}$

DTD

$projects \to projects^*$

$project \to name, (stable \mid dev), license$

$stable \to src, bin, doc$

$dev \to src, doc$

$license \to free \mid propr$

View DTD

$projects \to projects^*$

$project \to name, doc, license$

$license \to free \mid propr$

One problem: Size

DTD (annotated)

$r \to a_n$

$a_n \to a_{n-1}, a_{n-1}$

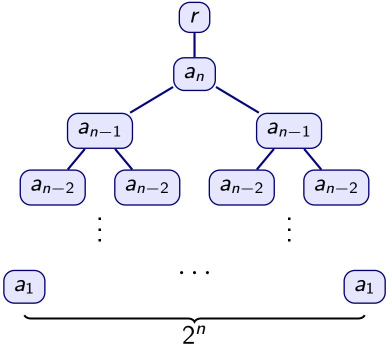
$a_{n-1} \to a_{n-2}, a_{n-2}$

$\dots$

$a_1 \to \text{empty}$

$A(a_n) = \text{false}$

$A(a_1) = \text{true}$



View DTD

$r \to \underbrace{a_1, \dots, a_1}_{2^n}$

$a_1 \to \text{empty}$

Observation

The view DTD may be of *exponential* size!

And another one: Regularity

DTD (annotated)

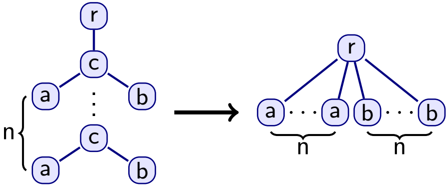
$r \rightarrow c$

$c \rightarrow (a, c?, b)$

$A(c) = \text{false}$

$A(a) = \text{true}$

$A(b) = \text{true}$



Observation

The view schema needs not be regular (in particular may not have a DTD)

Proposition

It is **undecidable** to test if the view schema can be captured with a DTD.

Approximation: Optimality criterion

Definition (Indistinguishability)

Two sets of trees L_1 and L_2 are *indistinguishable* by a class of queries $\mathcal{C}$ iff

$$\forall Q \in \mathcal{C}. \left[(\exists t_1 \in L_1.\, t_1 \models Q) \iff (\exists t_2 \in L_2.\, t_2 \models Q) \right].$$

Approximation

A DTD D^* is a **good approximation** of the view schema of D and A if $L(D^*)$ and $\{A(t) \mid t \in L(D)\}$ are indistinguishable by a relatively large class of queries.

Three approximations

DTD (annotated)

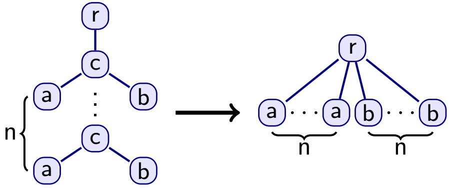
$r \to c$

$c \to (a, c?, b)$

$A(c) = \text{false}$

$A(a) = \text{true}$

$A(b) = \text{true}$



Parikh	Subword	Subset
$r \to (a, b)^*$	$r \to a^*, b^*$	$r \to (a \mid b)^*$
$\mathcal{X}Reg(\Downarrow, \Uparrow, [\,], \text{not})$	$\mathcal{X}Reg(\Downarrow, \Uparrow, \Rightarrow^+, \Leftarrow^+, [\,])$	$\mathcal{X}Reg(\Downarrow)$

Further results