

Approximate Schemas, Source-Consistency and Query Answering

Michel de Rougemont*, Adrien Vieilleribière†

Abstract

We use the Edit distance with Moves on words and trees and say that two regular (tree) languages are ε -close if every word (tree) of one language is ε -close to the other. A transducer model is introduced to compare tree languages (schemas) with different alphabets and attributes. Using the statistical embedding of [8], we show that Source-Consistency and Approximate Query Answering are testable on words and trees, i.e. can be approximately decided within ε by only looking at a constant fraction of the input.

1 Introduction

For a class \mathbf{K} of finite structures, a Schema is a subset $\mathbf{K}_0 \subseteq \mathbf{K}$, i.e. a subclass. We consider classes of strings, ordered and unordered trees and introduce the notion of *Approximate Schemas*, i.e. a notion to compare languages of strings or trees. We use the *Edit distance with Moves* on words and trees and say that two subclasses (languages) are ε -close if every word (tree) of one subclass is ε -close to the other. When the subclasses use the same alphabet, [10, 8] show, based on Property Testing, how to approximate the distance between words (trees) by sampling a number of subword independent of the size n , but only dependent on some ε parameter, and how to test regular properties in a similar way.

We wish to generalize this approach to languages which do not use the same vocabulary, i.e. compare languages such as $\mathbf{K}_1 = 0^*1^*$ and $\mathbf{K}_2 = c(ab)^*ca^*$. In the classical sense $\mathbf{K}_2 = c(ab)^*ca^*$ is close to $\mathbf{K}_3 = (ab)^*a^*$ as every word of length n of \mathbf{K}_2 is at relative distance $\frac{2}{n}$ to \mathbf{K}_3 , by deleting the two c 's, and conversely every word of \mathbf{K}_3 is at relative distance $\frac{2}{n}$ to \mathbf{K}_2 , by inserting two c 's. We say \mathbf{K}_2 and \mathbf{K}_3 are ε -close for any ε , as $\frac{2}{n} \leq \varepsilon$ for large enough n . There is a simple transducer (finite state machine with one state) \mathcal{T} which translates 0 into ab , 1 into a , such that the image of $\mathbf{K}_1 = 0^*1^*$, by \mathcal{T} , i.e. $\mathcal{T}(0^*1^*) = (ab)^*a^*$. In this case, we say that \mathbf{K}_1 and \mathbf{K}_3 are ε -similar for any ε . We introduce a model of transducer which transforms labelled trees with attributes and is well suited for XML, Data-Exchange and Data-Integration. It allows to compare schemas, XML files with different schemas and to also learn schemas from positive and negative examples.

We first present the approach for words, and regular expressions as schemas, but it generalizes naturally to trees and other classes of structures. Given a word w in some alphabet Σ , a regular expression r on a different alphabet Σ' and a deterministic transducer \mathcal{T} , we say that w is ε -close to r via \mathcal{T} if there exists w' ε -close to w such that $\mathcal{T}(w')$ is ε -close to r . In Data Exchange [7, 3], a setting is a triple $(\mathbf{K}_S, \mathcal{T}, \mathbf{K}_T)$ where $\mathbf{K}_S, \mathbf{K}_T$ are the Source and Target Schemas, and \mathcal{T} is a binary relations between Source structures I and target structures J . Given a Data-Exchange setting, *Source-Consistency* takes an input Source I of size n and decides if there exists a J such that $(I, J) \in \mathcal{T}$ and $J \in \mathbf{K}_T$, whereas *Query Answering* takes

*mdr@lri.fr, University Paris-II, CNRS-LRI, France

†vieille@lri.fr, LRI, University Paris-Sud, France

an input Source I and decides if all J such that all $(I, J) \in \mathcal{T}$ are such that $J \in \mathbf{K}_{\mathbf{T}}$. We define the approximate versions when \mathcal{T} is the binary relation associated with a transducer \mathcal{T} . Let $(\mathbf{K}_{\mathbf{S}}, \mathcal{T}, \mathbf{K}_{\mathbf{T}})$, be such a Data-Exchange setting.

- Approximate Source-Consistency takes I and ε as input and decides if I is ε -close to $\mathbf{K}_{\mathbf{T}}$ via \mathcal{T} , i.e. if there exists I' ε -close to I and J' such that $(I', J') \in \mathcal{T}$ and J' is ε -close to $\mathbf{K}_{\mathbf{T}}$.
- Approximate Query answering takes I and ε as input and decides if there exists I' ε -close to I such that all J' such that $(I', J') \in \mathcal{T}$ are ε -close to $\mathbf{K}_{\mathbf{T}}$.

We first consider deterministic \mathcal{T} and generalize to non deterministic transducers and transducers with null-transitions. In the case of XML files, I is a tree and \mathbf{K} is a DTD. We wish to decide if a given file is close or far from a DTD after transformation by a transducer. In many Data-Integration situations, the transducer needs to manage attributes, one of the motivations of our model.

Property Testing is a framework to approximate decision problems, which is well suited on classes of finite structures with a distance. Given a parameter $0 \leq \varepsilon \leq 1$, an ε -tester [14, 9] for a property P decides if a structure satisfies the property P or if it is ε -far from satisfying the property P . A property P is *testable* if there exists a randomized algorithm which is an ε tester for all $\varepsilon > 0$ and whose time complexity is independent of the size of the structure and only dependent on ε . When the structure is ε -close to the property, a *Corrector* finds in linear time a structure which satisfies the property and which is ε -close to the initial structure.

The main results of the paper use the *Edit Distance with Moves* on words and trees, and a transducer model to transform structures from one language into structures of another language. We show:

- Source-Consistency is testable on words and trees, i.e. for all ε there is an ε -tester which decides in time independent of n the size of I if the input I is Source-Consistent or is ε -far from being Source-Consistent.
- Query Answering is similarly testable on words and trees.
- If a word w is ε -close to a regular expression r via \mathcal{T} , we can find w' ε -close to w such that $\mathcal{T}(w')$ is ε -close to r in linear time.

These results are based on the testers and correctors for words (resp. trees) introduced in [10] for regular words (resp. regular trees) and depend on this specific *Edit Distance with Moves*. Other distances do not provide such results. We also use the embedding of words into statistical vectors introduced in [8] which yields natural testers to decide Equality, Membership of a regular language and a polynomial time algorithm to decide if two non-deterministic automata are ε -equivalent. A corrector for XML along this theory presented in [4], is also used. Learning regular properties on words and trees is also closely connected to these problems, using similar techniques.

In section 2, we review the basic data exchange models, the approximation of decision problems provided by the ε -testers. We study approximate data exchange on words in section 3, and on trees in section 4.

2 Preliminaries

A k -ranked ordered tree with n nodes is a structure $T = (D_n, \text{FirstChild}, \text{NextSibling}, \text{root})$ where the domain $D_n = \{1, \dots, n\}$ is the set of nodes with at most k successors. The relations FirstChild , NextSibling are binary relations such that $\text{FirstChild}(u, v)$ if v is the first child of u , $\text{NextSibling}(v, v')$ if v' is the next sibling of

v , and $root$ is a distinguished element of D_n with no predecessors. There are at most k siblings and the graph with Firstchild and Nextsibling edges is a tree. A word w is a 1-ranked tree and the Nextsibling relation is empty. An unranked ordered tree is a structure $T = (D_n, \text{Firstchild}, \text{Nextsibling}, root)$ without a fixed limitation on the number of siblings, which remains less than $n - 1$. An unranked unordered tree is a structure $T = (D_n, \text{Child}, root)$ with no order on the siblings.

We consider labelled trees with attribute values, i.e. nodes have a label (tag) and attributes with values in a set Str . Given a finite alphabet Σ and a finite set A of attributes, we consider a class $\mathbf{K}_{\Sigma, A}$ of (Σ, A) labeled tree structures I which can be ordered or unordered. They have two domains: D is the set of nodes, and Str the set of attribute values.

- On ordered trees, $I = (D, Str, \text{Firstchild}, \text{Nextsibling}, root, L, \lambda)$
- On unordered trees, $I = (D, Str, \text{Child}, root, L, \lambda)$

where $root$ is the root of the tree, $L : D \rightarrow \Sigma$ defines the node label, and $\lambda : D \times A \rightarrow Str$ is a partial function which defines the attributes values of a node, when they exist. On unordered trees Child is the edge relation of an unranked tree, whereas on ordered trees Firstchild defines the first child of a node and Nextsibling defines the successor along the siblings.

Example 1: Strings, Relations and Trees as $\mathbf{K}_{\Sigma, A}$ classes.

- Let $\Sigma = \{0, 1\}$, $A = \emptyset$, $D = \{1, \dots, n\}$, $Str = \emptyset$, $\mathbf{K}_1 = \{I = (D, Str, \text{Child}, r, L, \lambda)\}$ where Child is the natural successor over D and $L : D \rightarrow \{0, 1\}$. This class represents binary words of length n . If $A' = \{A_1, A_2\}$, $Str = \{a, c\}$, we have binary words with two attributes, and values in Str . For example $1.0_{[A_1=a]}.1_{[A_1=c, A_2=c]}.0_{[A_1=a]}.1.0_{[A_1=a]}$ is a word where certain letters have attribute values. For example $L(2) = 0$ and $\lambda(2, A_1) = a$.
- Let $\Sigma = \emptyset$, $A = \{A_1, A_2, A_3\}$, $D = \{1, \dots, n\}$, and Str an arbitrary set of string values, $\mathbf{K}_2 = \{I = (D, Str, \text{Child}, r, L, \lambda)\}$, such that Child is the edge relation of an unranked tree of depth 1 whose leaves have attributes A_1, A_2, A_3 and values in Str . This class represents ternary relations with $n - 1$ tuples having values in Str .
- Let $\Sigma = \{0, 1\}$, $D = \{1, \dots, n\}$, $A = \emptyset, Str = \emptyset$, $\mathbf{K}_3 = \{I = (D, Str, \text{Firstchild}, \text{Nextsibling}, L, \lambda)\}$. This class represents unranked ordered trees with n nodes without attributes.

We consider regular schemas, i.e. classes defined by automata. For unranked ordered trees, the automata have been defined by [17] and DTDs are special cases. XML schemas define regular properties of ordered and unordered trees.

2.1 Distances

The Edit distance has been introduced in [18] for comparing strings and generalized in [16] for trees. The Edit distance between two words is the minimal number of insertions, deletions and substitutions of a letter required to transform one word into the other. The best known algorithm for computing the edit distance between two words is in $O(n^2/\ln n)$ [12]), for the time complexity. Several approximation algorithms have been proposed but no algorithm achieves an approximation with a constant error for this distance.

Consider an extension of the distance: the *Edit distance with moves*, where arbitrary substrings can be moved in one step, as in [6, 15]. Let the *relative Edit distance with moves* between two strings w and w' , written $\text{dist}(w, w')$, the minimal number of elementary operations (insertions, deletions, substitutions and moves) on w to obtain w' , divided by $\max\{|w|, |w'|\}$.

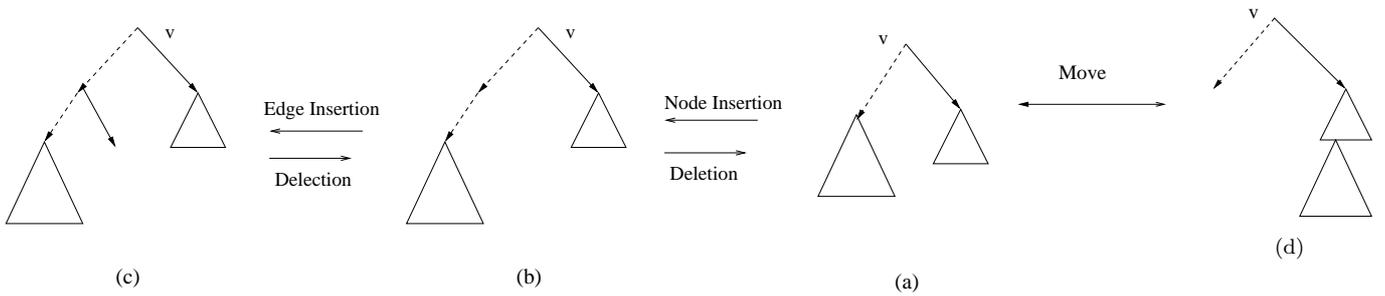


Figure 1: Tree-Edit operations for binary trees in (a),(b),(c), and a Move operation in (d).

The Tree Edit distance consider basic operations on ordered labelled trees: *updating* a label or an attribute, *inserting* a node or an edge, and *deleting* a node or an edge. An attribute value is considered as a label: it can be modified, entirely removed or added, at a unit cost, independent of the size of the attribute value. On ordered binary trees without labels, operations must preserve the binary tree and certain edge deletions are prohibited. Hardness results and polynomial algorithms for ordered and unordered trees, are presented in [2]. In figure 1, we obtain the tree (b) from the tree (a) with a node insertion and the tree (c) from the tree (b) with an edge insertion.

The *Tree Edit Distance with Moves* generalizes the Edit distance with a new operation, the Move of an entire subtree from one node to another node (e.g. between (a) and (d) in figure 1).

Definition 1. *The Edit distance with moves between two ordered unranked trees T and T' , written $\text{dist}(T, T')$, is the minimal number of elementary operations on T to obtain T' , divided by $\max\{|T|, |T'|\}$. The distance between T and a language L , noted $\text{dist}(T, L)$ is the minimum distance between $\text{dist}(T, T')$ for $T' \in L$*

We say that two trees T and T' are ε -close if their relative distance is less than ε where the relative distance is the Edit distance with moves divided by $\text{Max}\{|T|, |T'|\}$.

2.2 Transductions

Two main approaches are used to specify transformations between a source instance and possible target instances. Let \mathcal{T} be a binary relation defined by some pairs of structures (I, J) where $I \in \mathbf{K}_S$ and $J \in \mathbf{K}_T$ for a target schema \mathbf{K}_T . The transformation \mathcal{T} can be defined by regular transductions or by formulas linking sources to targets.

A transduction transforms a source structure I into a target structure J in the language of the target. It does not change the basic structure of I but transforms the tags from the source language to tags of the target language. There is a large literature on tree transducers and our model is close to the top-down tree transducers of [11], but also handles attributes values.

Let \mathbf{K}_S be a set of (Σ_S, A_S) trees and \mathbf{K}_T be a set of (Σ_T, A_T) trees. A transducer associates in a top-down manner with a node v with label in Σ_S and attribute values along attributes in $A_S = \{A^1, \dots, A^k\}$, a local new finite (Σ_T, A_T) subtree and new attribute values for each node of that tree. In particular, a node v with attribute values $A_1 = a$ can generate a child node with label A_1 and data-value a , and conversely. This setting is motivated by the XSLT language where this feature is standard.

Let H_{Σ_T, A_T} be the set of finite sequences of *finite trees* (hedges) with attributes in A_T and values in $\text{Str} \cup \text{Var}$. The values of the attributes may be defined, i.e. in Str , or unknown. In this last case, we use a

new variable in Var . Let $H_{\Sigma_T, A_T}[Q]$ be the set of finite sequences of finite trees where one leaf of each tree is a distinguished element labelled by a sequence of states in Q , which is possibly empty.

The transducer is defined by three functions. The function δ defines the local tree transformation at each node, the function h defines the transformation of attribute values (into possibly null values) and the *partial* function μ defines the positions of the new attribute values in the new finite tree t introduced by δ .

Definition 2. A tree transducer \mathcal{T} between (Σ_S, A_S) trees and (Σ_T, A_T) trees is defined by (Q, q_0, δ, h, μ) where:

- $\delta : \Sigma_S \times Q \rightarrow H_{\Sigma_T, A_T}[Q]$
- $h : \Sigma_S \times Q \times A_S \rightarrow \{1\} \cup Var$,
- $\mu : \Sigma_S \times Q \times A_T \times D_T \rightarrow \{1, 2, \dots, k\}$, where D_T is the set of nodes of the sequence of trees (hedge) defined by δ .

The function h extends to a function $h' : \Sigma_S \times Q \times Str \rightarrow Str \cup Var$ as follows. For label $l \in L_S$, state $q \in Q$, if $h(l, q, A_S^i) = 1$ then $h'(l, q, x_i) = x_i$. If $h(l, q, A_i) = V \in Var$ then $h'(l, q, x_i) = V$. Notice that this model is precisely what XSLT allows, but some attribute values may be kept in some state, i.e. when $h(l, q, A_S^i) = 1$, and assigned Null values (variables in Var) in some other states.

A top-down run starts with the root node in state q_0 and transforms each node in a top-down manner. A node v with label $l \in L_S$, state q , attributes in A_S and attribute values in Str is replaced by a finite subtree with labels in L_T , attributes in A_T and attribute values in $Str \cup Var$, through the transformation $\mathcal{T}(l, q)$ defined by:

- $\delta(l, q) = (t_1, t_2, \dots, t_s)$ a set of finite trees with a distinguished leaf element labelled by a sequence of states. The trees t_i are inserted below the node v as siblings, as defined in [11], where duplications and deletions are allowed.
- Let v a node with attribute values $x_1, \dots, x_k \in Str$. The function h extends to a function h' which determines if the value is kept or assigned a Null value.
- If $\mu(l, q, A_T^i, w) = i$ then the value of the first attribute of node $w \in D_T$ is the value $h'(x_i)$. The function sets the value of the attribute of w as the image through h' , defined by h of the i -th value of the node v .

Notice the \mathcal{T} is a finite object as it only depends on finite trees and finitely many attributes. The set Str is not finite as it depends on arbitrary trees but the set Var of Null values is finite and determined by the labels and states.

At each node, we apply the transformation $\mathcal{T}(l, q)$ for label l and state q and we obtain a tree T' with labels in Σ_T , attributes in A_T , and attribute values in $Str \cup Var$. In the case of strings, if each $\delta(a, p) = u[q]$ where u is a finite word, we obtain the classical transducer which replaces in state p a letter a with the word u and goes to state q . The transducer is *linear* when no duplication is allowed.

Example 2 : transductions on strings and trees, with attributes.

- (a) Let $\Sigma_S = \{0, 1\}$, $A_S = \{N, M\}$, $D = \{1, \dots, n\}$, $\mathbf{K}_S = \{I = (D, Str, Child, r, L, \lambda)\}$ where Child in the natural successor over D and $L : D \rightarrow \{0, 1\}$, as in example 1 of binary words. Let $\Sigma_T = \{a, b, n\}$, $A_T = \{P\}$, and the corresponding \mathbf{K}_T , defined by the transducer (Q, q, δ, h, μ) :

- $Q = \{q\}$, $\delta(0, q) = n.d[q]$, $\delta(1, q) = b.d[q]$, i.e. words with only one successor,
- for all l, q , $h(l, q, M) = 1$, $h(l, q, N) = V_1$. Hence $h'(l, q, a) = a$, $h'(l, q, c) = V_1 \in Var$,
- μ sets the value of the attribute M on the node n of the word $n.d$ with the value $@M$ of the attribute M .

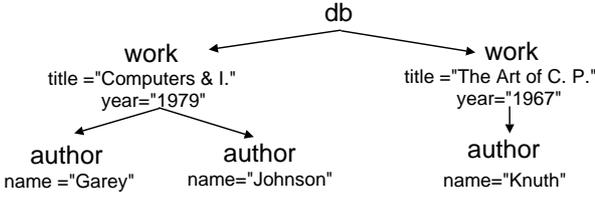


Figure 2.1 : An ordered tree with attributes.

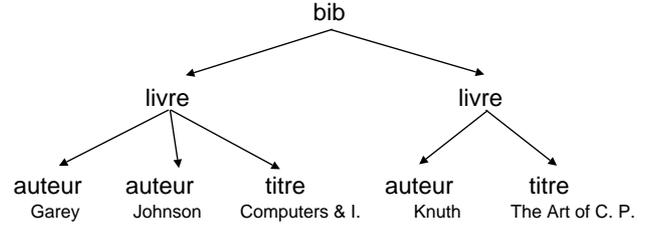


Figure 2.2 : The result of the transducer.

The image of the word on the label set $\{0, 1\}$, with attributes N, M defined by

$$1.0_{[N=a]} \cdot 1_{[N=c, M=c]} \cdot 0_{[M=c]} \cdot 1.0_{[N=a]}$$

is a word on the label set $\{a, b, n\}$ with attribute P and attribute values in $Str \cup Var$, i.e.

$$b.d.n_{[P=a]} \cdot d.b.d.n_{[P=V_1]} \cdot b.d.n.d$$

(b) Let I be Source Instance described by the figure 2.1

Let \mathcal{T} be the transducer given by the XSLT program :

```
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
  <bib>
    <xsl:apply-templates/>
  </bib>
</xsl:template>
<xsl:template match="work">
  <livre>
    <xsl:apply-templates/>
    <titre>
      <xsl:value-of select="@title" />
    </titre>
  </livre>
</xsl:template>
<xsl:template match="author">
  <auteur>
    <xsl:value-of select="@name" />
  </auteur>
</xsl:template>
```

The transducer \mathcal{T} is defined by :

$\Sigma_S = \{ db, work, author \}$
 $A_S = \{ title, name \}$
 $\Sigma_{ST} = \{ bib, livre, auteur \}$
 $A_T = \{ name \}$.
 $Q = \{ q \}$, $\delta(db, q) = bib[q]$
 $\delta(work, q) = (t_1, t_2)$, where $t_1 = livre[q]$ and $t_2 = titre$
 $\delta(author, q) = auteur$
 $h(work, q, titre) = 1$, $h(author, q, titre) = 1$.
 The function μ sets the attribute value of *title* as default attribute (PCDATA) to the node *titre* of the subtree t_2 , and the attribute value of *author* as default attribute (PCDATA) to the node *auteur* of $\delta(author, q)$.

After the XSLT transduction the structure is given by the figure 2.2.

In practice, the transducer is deterministic and defined by an XSLT program π . It tries to translate tags, modify attributes names and values to fit a target schema. In data exchange [7, 3], there are other models to express relationships between source and target schemas, and there is a close connection between Approximate Query Answering and Approximate Data Exchange.

2.3 Property Testing and Approximation

Property Testing has been initially defined in [14] and studied for graph properties [9] in conjunction with learning methods. It has been successfully extended to various classes of finite structures, such as words where regular languages are proved testable [1] for the Hamming distance, and trees where regular tree languages are proved testable [10] for the Edit Distance with Moves. A tester approximates a property by looking at a constant fraction of the input, independent of the global size of the input.

We say that two structures $U_n, V_m \in \mathbf{K}$, whose domains are respectively of size n and m , are ε -close if their distance $\text{dist}(U_n, V_m)$ is less than $\varepsilon \times \max(n, m)$. They are ε -far if they are not ε -close. The distance of a structure U_n to a class \mathbf{K} is $\text{dist}(U_n, \mathbf{K}) = \text{Min}_{V \in \mathbf{K}} \{\text{dist}(U_n, V)\}$. In this paper, we consider this notion of closeness for words and trees since the representation of their structure is of linear size. For other classes of structures, such as binary relations or graphs, one may define the closeness relatively to the representation size (e.g. εn^2 for graphs as there are at most n^2 edges) instead of the domain size.

Definition 3. Let $\varepsilon \geq 0$ be a real. An ε -tester for a class $\mathbf{K}_0 \subseteq \mathbf{K}$ is a randomized algorithm A such that:

- (1) If $U \in \mathbf{K}_0$, A always accepts;
- (2) If U is ε -far from \mathbf{K}_0 , then $\Pr[A \text{ rejects}] \geq 2/3$.

The *query complexity* is the number of boolean queries to the structure U of \mathbf{K} . The *time complexity* is the usual time complexity where the complexity of a query is one and the time complexity of an arithmetic operation is also one. A class $\mathbf{K}_0 \subseteq \mathbf{K}$ is *testable* if for all $\varepsilon > 0$, there exists an ε -tester whose time complexity depends only on ε .

Definition 4. An ε -corrector for a class $\mathbf{K}_0 \subseteq \mathbf{K}$ is a (randomized) algorithm A which takes as input a structure I which is ε -close to \mathbf{K}_0 and outputs (with high probability) a structure $I' \in \mathbf{K}_0$, such that I' is ε -close to I .

When an XML file is given by its DOM representation, the operations associated with the Edit distance with moves take unit costs. The move requires only the modifications of a few pointers. Testers for regular properties of words and trees have been presented in [10, 8] for this distance and a corrector for regular trees presented in [4].

2.3.1 Approximate Schemas

We first consider classes of structures on the same alphabet Σ and attributes A .

Definition 5. Let $\varepsilon \geq 0$. Let $\mathbf{K}_1, \mathbf{K}_2$ be two classes of structures. We say that \mathbf{K}_1 is ε -contained in \mathbf{K}_2 , if all but finitely many words of \mathbf{K}_1 are ε -close to \mathbf{K}_2 . \mathbf{K}_1 is ε -equivalent, written \equiv_ε , to \mathbf{K}_2 , if both \mathbf{K}_1 is ε -contained in \mathbf{K}_2 and \mathbf{K}_2 is ε -contained in \mathbf{K}_1 .

Notice that the image of regular properties on strings is also regular, so we need to efficiently distinguish approximately equivalent schemas.

Example 3 :

- (a) Let $\mathbf{K}_1 = O^*1^*$ and $\mathbf{K}_2 = c(ab)^*ca^*$ be two regular expressions. There is a transducer with one state which replaces the letter 0 by ab and the letter 1 by a . The transducer \mathcal{T} is specified by $0 : ab$ and $1 : a$. The image of O^*1^* by \mathcal{T} is $\mathcal{T}(O^*1^*) = (ab)^*a^*$, which is ε -close to $c(ab)^*ca^*$ for any ε . Any word $w \in (ab)^*a^*$ of length n is at distance $2/n$ from a word of $c(ab)^*ca^*$, as two insertions of c are required.

- (b) Let $\mathbf{K}_S^1 =$

```

<!ELEMENT db (work*)>
<!ELEMENT work (author*)>
  <!ATTLIST work
    title CDATA #REQUIRED
    year CDATA>
<!ELEMENT author (EMPTY)>
  <!ATTLIST author
    name CDATA #REQUIRED>

```

and $\mathbf{K}_T^1 =$

```

<!ELEMENT bib (livre*)>
<!ELEMENT livre (auteur+, titre , annee)>
  <!ELEMENT auteur #PCDATA>
  <!ELEMENT titre #PCDATA>
  <!ELEMENT annee #PCDATA>

```

be two DTDs, special regular schemas on trees. If we use the the transducer of example 2, then $\mathcal{T}(\mathbf{K}_S^1)$ is also ε -close to \mathbf{K}_T^1 .

A more general definition of closeness, called *similarity* considers structures in different languages and allows for a transducer to compare the schemas.

Definition 6. Let $\varepsilon \geq 0$ and $\mathbf{K}_1, \mathbf{K}_2$ be two classes of structures. We say that \mathbf{K}_1 is ε -included in \mathbf{K}_2 , if there exists a transducer \mathcal{T} such that all but finitely many words of $\mathcal{T}(\mathbf{K}_1)$ are ε -close to \mathbf{K}_2 . \mathbf{K}_1 is ε -similar, written \simeq_ε , to \mathbf{K}_2 , if both \mathbf{K}_1 is ε -included in \mathbf{K}_2 and \mathbf{K}_2 is ε -included in \mathbf{K}_1 .

This definition is motivated by Data-Integration where documents and DTDs use tags in different languages and we need to efficiently decide ε -inclusion and ε -similarity. Notice that the image of regular properties of strings by transducers is also regular, but this is not the case on trees.

2.3.2 A statistical embedding on strings.

For a finite alphabet Σ and a given ε , let $k = \frac{1}{\varepsilon}$. Let the $\text{dist}(w, w')$ be the *Edit distance with moves* as defined in Example 3 and the embedding of a word in a vector of k -statistics, describing the number of occurrences of all subwords of length k in w . For all words u of length k , let $\#u$ be the number of occurrences of u in w and $\mathbf{u}\text{-stat}(w)[u] \stackrel{\text{def}}{=} \frac{\#u}{n-k+1}$.

The vector $\mathbf{u}\text{-stat}(w)$ is of dimension $|\Sigma|^k$ is also the *probability distribution* that a uniform random subword of size k of w be a specific u :

$$\mathbf{u}\text{-stat}(w)[u] = \Pr_{j=1, \dots, n-k+1} [w[j]w[j+1] \dots w[j+k-1] = u]$$

This vector is a generalized Parikh mapping [13], is related to the previous work of [5] where the subwords of length k were called *shingles*, and is also called a k -gram in statistics. The statistical embedding [8] associates a statistics vector $\mathbf{u}\text{-stat}(w)$ with a string w and a union of polytopes $H = \{\mathbf{u}\text{-stat}(w) : w \in r\}$ in the same space, to a regular expression r , such that the distance between two vectors (for the L_1 norm) is approximately $\text{dist}(w, w')$ (for w and w' of approximately the same length) and the distance between a vector and a union of polytopes is approximately $\text{dist}(w, L(r))$.

Example 4 : Let $w = 000111$ of length $n = 6$, $k = 2$, $n-k+1 = 5$ and $\Sigma = \{0, 1\}$. The number of occurrences of $u = 00$, $u = 01$, $u = 10$ and $u = 11$ in w are respectively 2, 1, 0, 2. For a lexicographic enumeration of the length 2 binary words, the $\mathbf{u}\text{-stat}$ vector associated with the word w is : $\mathbf{u}\text{-stat}(w) = \begin{pmatrix} 2/5 \\ 1/5 \\ 0 \\ 2/5 \end{pmatrix}$. Let $s_0 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$, $s_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$, $s_2 = \begin{pmatrix} 1/3 \\ 1/3 \\ 1/3 \\ 0 \end{pmatrix}$, $k = 2$ and consider the regular expressions $r_1 = 0^*1^*$, $r_2 = (001)^*1^*$, $r_3 = 0^*(001)^*1^*$. The convex hull associated with r_1 is $H_1 = \text{Convex} - \text{Hull}(s_0, s_1)$, the one associated with r_2 is $H_2 = \text{Convex} - \text{Hull}(s_1, s_2)$ and the one associated with r_3 is $H_3 = \text{Convex} - \text{Hull}(s_0, s_1, s_2)$. The representation of the polytope H_2 in the space of vectors of size $\Sigma^k = 2^2$ is given in the figure 3.

These techniques yield several basic testers [8] which we will use as Black Box:

- **Equality tester** between two words w and w' of approximately the same length. Sample the words with at least $N \in O\left(\frac{(\ln|\Sigma|)|\Sigma|^{2/\varepsilon}}{\varepsilon^3}\right)$ samples, define $\widehat{\mathbf{u}\text{-stat}}(w)$ and $\widehat{\mathbf{u}\text{-stat}}(w')$ as the $\mathbf{u}\text{-stat}$ of the samples. Reject if $|\widehat{\mathbf{u}\text{-stat}}(w) - \widehat{\mathbf{u}\text{-stat}}(w')|_1 \geq \varepsilon$.
- **Membership tester** between a word and a regular expressions r . Compute $\widehat{\mathbf{u}\text{-stat}}(w)$ as before and the polytope H associated with r in the same space. Reject if the geometrical distance from the point $\widehat{\mathbf{u}\text{-stat}}(w)$ to the polytope H is greater then ε .

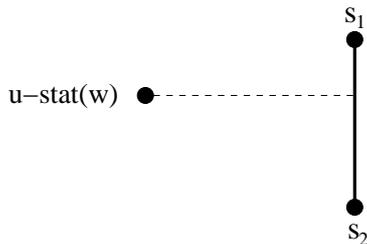


Figure 3: $\mathbf{u-stat}(w)$ and H_2 .

- **Equivalence tester** between two regular expressions r_1 and r_2 . Associate the polytopes H_1 and H_2 in the same space as $\mathbf{u-stat}(w)$, represented by the nodes $H_{1,\varepsilon}$ and $H_{2,\varepsilon}$ on a grid of step ε . If $H_{1,\varepsilon} \neq H_{2,\varepsilon}$ then r_1 and r_2 are ε far.
- **Proximity tester** between two regular expressions r_1 and r_2 . Associate the polytopes H_1 and H_2 , represented by the nodes $H_{1,\varepsilon}$ and $H_{2,\varepsilon}$ on a grid of step ε . If a node of $H_{1,\varepsilon}$ is ε -close to a node of $H_{2,\varepsilon}$ then accept else reject.

The membership tester is polynomial in the size of the regular expression (or non-deterministic automaton) whereas it was exponential in this parameter in [10]. In this paper, we use the Membership tester, the Equivalence tester and the Proximity tester as Black Box.

Construction of H . Every word w close to a regular language L is close to a word w' which follows some basic small loops, i.e. $w \equiv_k w' = v_1 u_1^{a_1} . v_2 . u_2^{a_2} \dots v_l . u_l^{a_l} . v_{l+1}$, where $|v_i|, |u_i| \leq m$, $a_i \in \mathbb{N}$ and m is the size of an automaton A for L . $\{u_1, u_2, \dots, u_l\}$ is a compatible set of loops, i.e. there is a run along w' in A . We can indeed regroup the loops in w with *moves* operations and obtain w' .

Example 5: Let $L = L(r_2) = (001)^*1^*$ and $w = 000111$ of length $n = 6$, $k = 2$ as in example 4. $\text{dist}(w, L) = 1/6$ as we can just remove the first 0 to obtain $w' = (001)^11^2$. If $w_1 = w.w = 000111000111$ then $\text{dist}(w_1, L) = 3/12$ as we can just remove the first and fourth 0 and move (001) from the 6-th position to the first and obtain $w'_1 = (001)^21^4$.

Associate to each u_i its statistics vector $(u_i) = \mathbf{u-stat}(u_i.u_i\dots u_i)$. By definition $\mathbf{u-stat}(w)$ is ε close to $\sum_{i=1,\dots,l} \lambda_i(u_i)$, i.e. is close to a point in the simplex of the $\{(u_1), (u_2), \dots, (u_l)\}$. In order to construct H from A , first construct the Directed Acyclic Graph G_A of connected components C_i of A . A node is a connected component $C_i = \{q_{i_1}, q_{i_2}, \dots, q_{i_p}\}$ such that there is a run in A from any two states in C_i . There is an edge between C_i and C_j if there a run between one state of C_i and one state of C_j . Suppose the initial state $q_0 \in C_0$ and all states accept. Enumerate all paths π from C_0 to a sink in G_A : for each such path there is a polytope H_π whose summits are all (u_i) such that u_i is a loop of length less than m from a state q_i in a connected component of π . All these constructions are polynomial in the size m of A . Finally H is the union of the H_π .

3 Source-consistency, Query answering on words

Consider a deterministic transducer \mathcal{T} and a target schema $\mathbf{K}_\mathcal{T}$ defined by a regular expression r . Approximate Source-Consistency considers a word w as input and a parameter ε , and decides:

- (1) If $\mathcal{T}(w) \in r$ or if

(2) If w is ε -far from any w' such that $\mathcal{T}(w')$ is ε -close to r .

An ε -tester for Approximate Source-Consistency is a randomized algorithm which decides with high probability the conditions 1 and 2. For simplicity, we consider a transducer as a finite state machine where transitions are labelled by a tuple $(\alpha : \beta)$ where α is a letter for the Source alphabet and β is a word for the target alphabet.

Example 6: Let $\mathcal{T}, \mathbf{K}_2$ be defined as in example 3(a). For a given instance $I = 0001111$, $\mathcal{T}^1(I) = ababab.aaaa$ is at distance $\frac{1}{5}$ from the target schema $c(ab)^*ca^*$. A corrector for \mathbf{K}_2 will transform $ababab.aaaa$ into $c.ababab.c.aaaa$ in linear time.

3.1 Tester for Approximate Source-Consistency

We present an ε -Tester for Approximate Source-Consistency, first for the case of a transducer with one state, and generalize it in a second step. Let \mathbf{K}_T be the regular expression r on the target alphabet Σ , for which we build its equivalent polytope H_T .

The Tester uniformly samples $I = w$ to obtain random subwords u of length k to obtain $\widehat{\mathbf{u-stat}}(w)$ which is close to $\mathbf{u-stat}(w)$. In the case of a transducer with one state we may also directly estimate $\widehat{\mathbf{u-stat}}(\mathcal{T}(w))$.

3.1.1 Transducer with one state

If u is a random sample, consider $\mathcal{T}(u)$ and obtain a subword v of $\mathcal{T}(I)$, on which we will construct an approximate statistics. As the transducer produces words of different lengths, we have to adjust the sampling probabilities to guarantee the approximation. Let $\alpha = \min_{a \in \Sigma_s} |\mathcal{T}(a)| > 0$, $\beta = LCM_{a \in \Sigma_s} |\mathcal{T}(a)|$, i.e. the Least Common Multiplier of the lengths $\mathcal{T}(a)$ and $k = 1/\varepsilon$.

Tester₁(w, k, N) :

1. Repeat until N outputs are generated.

Choose $i \in_r \{1, \dots, n\}$ uniformly, let $a = w[i]$ and $\gamma = |\mathcal{T}(a)|$,

Choose $b \in_r \{0, 1\}$ with $Prob(b = 1) = \frac{\gamma}{\beta}$,

If (b=1) { Choose $j \in_r \{0, \dots, \gamma - 1\}$ uniformly and let $X_{i,j}$ be the subword of length k beginning at position j in $\mathcal{T}(a)$ and continuing with k letters on the right side. }

2. Let $\widehat{\mathbf{u-stat}}_T(w, k, N)$ be the $\mathbf{u-stat}$ vector of the $X_{i,j}$.

3. If the geometrical distance between $\widehat{\mathbf{u-stat}}(w, k, N)$ and H_T is greater than ε then reject else accept.

Let $N_0 = O(\beta \ln(1/\varepsilon) \ln(|\Sigma|)/(\alpha \varepsilon^3))$.

Lemma 1. For any w, \mathcal{T} and regular \mathbf{K}_T , $\varepsilon > 0$ and $N \geq N_0$, $\text{Tester}_1(w, k, N)$ is a $2.\varepsilon$ -tester for Approximate Source Consistency.

Proof. If w is consistent, the tester accepts with high probability. Let's focus on the second condition in the definition of an ε -tester. Let $|w| = n$ and $X_{i,j}$ a succesful output with indices i and j , i.e. a subword of length k . For $u \in |\Sigma_T|^k$ we define

$$\widehat{\mathbf{u-stat}}_T(w, k, N)[u] = \sum_{i=1}^N \frac{\#u}{N}, \quad u \in \{X_{i,j}\}$$

Any couple (i, j) is chosen with probability $\frac{1}{n\beta}$, hence all $X_{i,j}$ are obtained with equal probability and the expectation of \widehat{ustat}_T is $\mathbf{u}\text{-stat}(\mathcal{T}(w))$. We can apply a Chernoff bound to each component u :

$$\Pr(|\widehat{ustat}_T(w, k, N)[u] - \mathbf{u}\text{-stat}(\mathcal{T}(w))[u]| > \varepsilon) \leq 2 \exp(-2\varepsilon^2 N)$$

We apply a union bound to all components and replace N by its lower bound:

$$\Pr(|\widehat{ustat}_T(w, k, N) - \mathbf{u}\text{-stat}(\mathcal{T}(w))| > \varepsilon) \leq 1/6$$

If we repeat $\Theta(N_0, \beta \ln(1/\delta)/\alpha)$ times the outer loop (1), we obtain N samples where the error is bounded by δ i.e. arbitrarily small, instead of $1/6$. We then apply a membership tester to $\widehat{ustat}_T(w, k, N)$ and H_T . We accept if the distance is less than 2ε , else we reject. We conclude that $\text{Tester}_1(w, k, N)$ is a 2ε -tester. \square

3.1.2 Transducer with m states

Let \mathcal{T} be a deterministic transducer with m states. We could generalize the previous tester, and sample subwords u of w which would yield possible subwords of $\mathcal{T}(w)$, if we knew the states in which \mathcal{T} was. As we don't, we use instead $\widehat{\mathbf{u}\text{-stat}}(w)$ which approximates $\mathbf{u}\text{-stat}(w)$ with $N = O(f(\varepsilon))$ samples to determine which π , i.e. sequences of compatible connected components of \mathcal{T} may yield feasible outputs along π .

Let A be the automaton defined from \mathcal{T} where we ignore outputs and all states accept. Recall the definitions of the graph G_A of the connected components C_i of A , and of the set π of compatible connected components given in section 2.3.2, when we described the construction of $H = \bigcup_{\pi} H_{\pi}$. Because G_A is a DAG, we can enumerate all the π , and there at most $|\Sigma_T|^m$ of them. Let H_{π} the polytope describing $\{\mathbf{u}\text{-stat}(w) : w \in L(A) \text{ along } \pi\}$. It has (u_i) ($\mathbf{u}\text{-stat}$ vector associated with the simple loop u_i) as summits where u_i is a simple loop for π , i.e. is a loop from a state q in a connected component of π . If w is ε -close to $L(A)$, then $\mathbf{u}\text{-stat}(w)$ must be close to some H_{π} , i.e. close to $\sum_i \lambda_i(u_i)$ such that $\sum_i \lambda_i = 1$. We just decompose a point in a polytope and say that w is close to $\sum_i \lambda_i(u_i)$ along π .

There may be several simple loops u_i from different states along π which may provide different outputs, or there may be different u_i, u_j with the same statistics, i.e. $(u_i) = (u_j)$ which may also provide different outputs. Let us write $(q) u_i : v_i$ if the transduction of the Source word u_i in a state q is the Target word v_i . We distinguish between *ambiguous loops* $u_{i'}$ such that several possible v_j are possible along π , i.e. $(q) u_{i'} : v_1$ and $(q') u_{i'} : v_2$ and *non ambiguous loops* where there is a unique $(q) u_i : v_i$. For non ambiguous loops, we replace the density $\lambda_i(u_i)$ in the Source space by $\lambda_i(v_i)$ in the Target space as u_i is transduced in v_i . For non ambiguous loops, we replace the density $\lambda_{i'}(u_{i'})$ in the Source space by a polytope $\sum_{\mu_1 + \dots + \mu_j = \lambda_{i'}} \mu_j(v_j)$ in the Target space, i.e. $\mu_1(v_1) + \mu_2(v_2)$ where $\mu_1 + \mu_2 = \lambda_{i'}$ in our example. We can now define the polytope $H_{w,\pi}$ in the target space, which captures the statistics of the images by \mathcal{T} of words close to w along π .

Definition 7. Given w, π, \mathcal{T} such that w is close to $\sum_i \lambda_i(u_i)$ along π , let $H_{w,\pi}$ be the set

$$\sum_{\text{non ambiguous } i} \lambda_i(v_i) + \sum_{\text{ambiguous } i'} \sum_{j \text{ s.t. } \sum_j \mu_j = \lambda_{i'}} \mu_j(v_j)$$

Lemma 2. If $\mathbf{u}\text{-stat}(w)$ is ε close to H_{π} and w' is ε close to w and accepted by A along π , then $\mathcal{T}_{\pi}(w')$ is ε close to the polytope $H_{w,\pi}$.

Proof. If $\mathbf{u}\text{-stat}(w)$ is ε close to H_{π} , then $\mathbf{u}\text{-stat}(w)$ is close to $\sum_i \lambda_{i=1,\dots,p}(u_i)$ along π . Because w' is ε close to w , $\mathbf{u}\text{-stat}(w') = \sum_i \lambda'_{i=1,\dots,p}(u_i)$ where $\sum_{i=1,\dots,p} |\lambda_i - \lambda'_i| \leq \varepsilon$. Because w' is accepted by A along π , $\mathcal{T}_{\pi}(w')$ is ε -close to $H_{w,\pi}$. \square

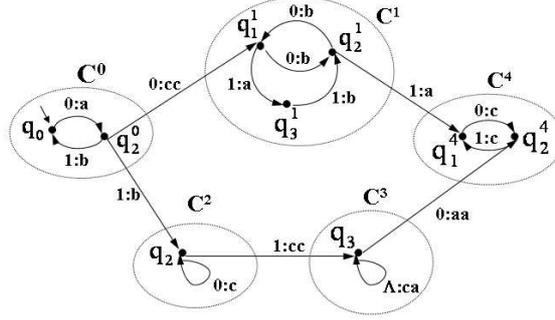


Figure 4: The transducer \mathcal{T} , its DAG G_A with the connected components.

Let $k = 1/\varepsilon$ the approximation parameter, \mathcal{T} the transducer, and H_T the polytope associated with the target schema \mathbf{K}_T be fixed. We can now describe the Tester for Source-Consistency which takes a (large) w as input.

Tester₂(w) :

Compute $Y = \widehat{\mathbf{u}\text{-stat}}(w)$ with $N = O(f(\varepsilon))$ samples.

Generate all π associated with \mathcal{T} .

If there is a π such that $H_{w,\pi}$ is ε close to H_T , then accept.

If no π is such that $H_{w,\pi}$ is ε close to H_T , then reject.

Theorem 1. *Source-Consistency is testable.*

Proof. We have to show that for all ε Tester₂ is an ε -tester. Because we use $N = O(f(\varepsilon))$ samples, independent of $n = |w|$, we satisfy the complexity requirement. If w satisfies Source-consistency ($\varepsilon = 0$), there is a π such that $\mathcal{T}(w) \in \mathbf{K}_T$ and the tester finds it and accepts. If w is ε far from Source-consistency then w is ε far from any w' such that $\mathcal{T}(w')$ is ε close to \mathbf{K}_T . Either $\mathbf{u}\text{-stat}(w)$ is far from any H_π , or $\mathbf{u}\text{-stat}(w)$ is close to some H_π but $H_{w,\pi}$ is far from H_T . The first condition is decided with high probability with the Membership tester of A , and the second condition is implied by lemma 2 and the Proximity tester. \square

Notice that the construction of $H_{w,\pi}$ is similar if \mathcal{T} is non deterministic or admits null transitions and therefore gives a general construction.

3.2 Example

Suppose $K_S = (01)^*0^*1^*(01)^*$, $K_T = (ab)^*cb^*a^+$, $w = 00001000100100000000$ and $k = 2$. The Directed Acyclic Graph G_A of connected components of \mathcal{T} is given by Figure 4. G_A has five connected components and two paths: $\pi_1 : \{C^0, C^1, C^4\}$ and $\pi_2 : \{C^0, C^2, C^3, C^4\}$. The transition $(q_0) 0 : a (q_0^2)$ indicates that on input 0 \mathcal{T} outputs a and goes to state (q_0^2) . The transducer is not deterministic and has null transitions $(q_3) \Lambda : ca (q_3)$, where Λ is the empty string, i.e. may output arbitrarily many ca in state q_3 . Observe that:

- w is close to K_S ,
- w is close to w' such that $\mathcal{T}(w')$ along π_1 is close to H_T ,
- all w' close to w are such that $\mathcal{T}(w')$ along π_2 is far from H_T .

For each π , we compute the statistical representation H_π of the automaton A restricted to π . Each summit of H_π corresponds to an elementary loop of the automaton, noted $C^i(q_1 \dots q_p)$ for a loop in the component C^i along the states $q_1 \dots q_p$. The path π_1 contains 4 elementary loops and two of them are ambiguous loops $C^0(q_0 q_2^0)$ and $C^4(q_1^4 q_2^4)$, i.e. have the same $\mathbf{u}\text{-stat}$ vectors. The path π_2 contains the same 2 ambiguous loops, a non ambiguous one, and a loop

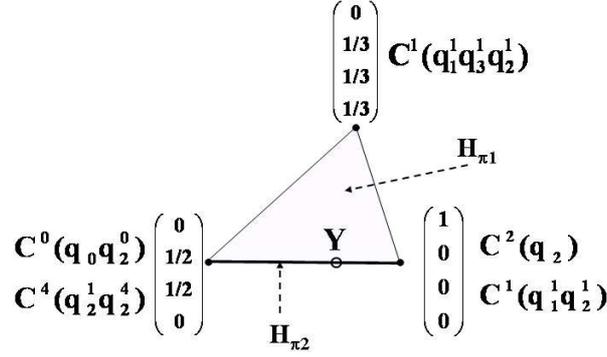


Figure 5: The source statistics: H_{π_1}, H_{π_2} and Y in dimension 4.

$C^3(q_3)$ on the empty word.

We first describe the statistics in dimension 4 of the source ($\Sigma_S = \{0, 1\}$), then the statistics in dimension 9 of the target ($\Sigma_T = \{a, b, c\}$), and finally describe the corrections on w to obtain w' such that $\mathcal{T}(w')$ along π_1 is close to H_T . Distances in the figures are approximate because of the projection.

3.2.1 The source's statistics

- H_{π_1} is the triangle in Figure 5 whose summits correspond to the statistics of its 4 loops, as two of them have the same statistics.
- H_{π_2} is a segment which is also the triangle's base.
- $\mathbf{u}\text{-stat}(w)$ can be approximated by the $\mathbf{u}\text{-stat}$ vector on few 6 samples, the vector $Y = \begin{pmatrix} 2/3 \\ 1/6 \\ 1/6 \\ 0 \end{pmatrix} \approx \mathbf{u}\text{-stat}(w)$.¹
- For each π , we decompose Y over H_π , if it is possible. Y is in H_{π_2} and H_{π_1} , therefore Y is decomposable over both H_π as:

$$Y = \begin{pmatrix} 2/3 \\ 1/6 \\ 1/6 \\ 0 \end{pmatrix} = \frac{1}{3} \begin{pmatrix} 0 \\ 1/2 \\ 1/2 \\ 0 \end{pmatrix} + \frac{2}{3} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

3.2.2 The target's statistics

For succinctness, we use a compact representation of the $\mathbf{u}\text{-stat}$ vectors of dimension 9, where only non-zero values are present.

- The polytope H_T associated with $(ab)^*cb^*a^+$ has three summits associated with the simple loops ab , bb , aa . It is the left triangle in the Figure 6.
- Transduction along π_1 . The decomposition of Y can also be written: $Y = \frac{1}{3} \cdot (01) + \frac{2}{3} \cdot (00)$ as the base is determined by the simple loops 00 and 01. The loop 00 is non-ambiguous and translates to bb . The density $\frac{2}{3}$ over 00 is translated into the same density over bb . The loop 01 is ambiguous: it can translate into ab in C^0 or cc in C^4 . The density $\frac{1}{3}$ can be decomposed as $\lambda_0 \cdot \begin{pmatrix} ab : 1/2 \\ ba : 1/2 \end{pmatrix} + \lambda_1 \cdot (cc : 1)$ such that $\lambda_0 + \lambda_1 = \frac{1}{3}$. We can then represent the polytope $H_{w, \pi_1}(Y)$ as $\lambda_0 \cdot \begin{pmatrix} ab : 1/2 \\ ba : 1/2 \end{pmatrix} + \lambda_1 \cdot (cc : 1) + \frac{2}{3} \cdot (bb : 1)$ where $\lambda_0 + \lambda_1 = \frac{1}{3}$, in the Figure 6.

¹The coordinates follow the lexicographic enumeration 00, 01, 10, 11.

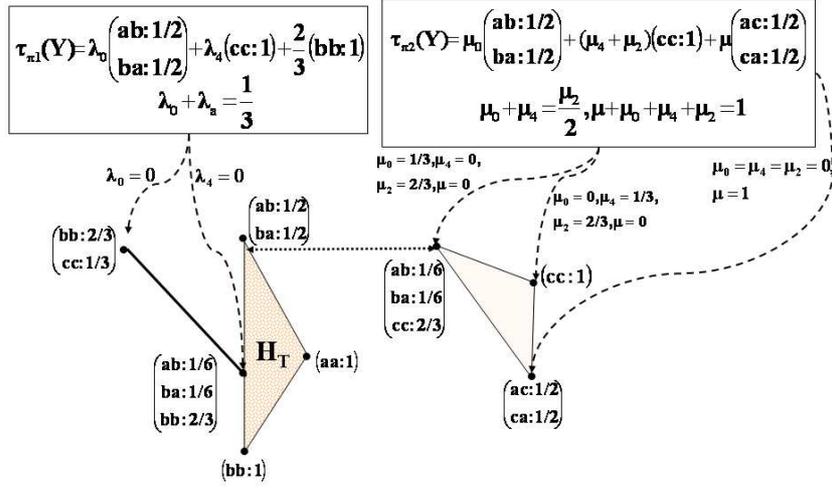


Figure 6: The target statistics

- Transduction along π_2 . The Λ -transition in C^3 allows arbitrary densities of the form $\mu \begin{pmatrix} ac : 1/2 \\ ca : 1/2 \end{pmatrix}$. The 00 simple loop is non ambiguous and translates to cc whereas the 01 loop is ambiguous as before. We can then represent the polytope $H_{w,\pi_2}(Y)$ as $\mu_0 \cdot \begin{pmatrix} ab : 1/2 \\ ba : 1/2 \end{pmatrix} + \mu_4 \cdot (cc : 1) + \mu_2 \cdot (cc : 1) + \mu \cdot \begin{pmatrix} ac : 1/2 \\ ca : 1/2 \end{pmatrix}$, where $\mu_0 + \mu_4 = \mu_2/2$ and $\mu_0 + \mu_4 + \mu_2 + \mu = 1$, as the right triangle in the Figure 6.

3.2.3 Correction on w

If the statistics $H_{w,\pi}$ is close to H_T , we may find w' such that $\mathcal{T}(w')$ is ε -close to \mathbf{K}_T . H_{w,π_1} has an intersection with H_T . We may choose the point where $\lambda_1 = 0$, i.e. 01 occurs in C^0 and translates into ab . We perform three moves on w to put 01's at the beginning. We obtain $w' = 01010100000000000000$, which is translated into a word close to the target schema. In fact, $\mathcal{T}(w') = abababaccbbbbbbbbb$. If we remove ac and add an a at the end, we obtain $w_T = abababcbbbbbbbbbba \in \mathbf{K}_T$.

H_{w,π_2} is far from H_T (the minimum l_1 norm is $2/3$) because w is ε -far for being source consistent along this path. To minimize the distance to the target, we would have to remove almost all the zeros (14 of them) to obtain the word $w' = 010110$, whose image $ababccaa$ is close to \mathbf{K}_T .

3.3 Tester for Approximate Query Answering

We can now generalize the previous tester to Query Answering.

Tester₃(w) :

Compute $Y = \widehat{\text{u-stat}}(w)$ with $N = O(f(\varepsilon))$ samples.

Generate all π associated with \mathcal{T} .

If there is a π such that $H_{w,\pi}$ is ε included in H_T , then accept.

If no π is such that $H_{w,\pi}$ is ε included in H_T , then reject.

Theorem 2. *Query Answering is testable.*

Proof. The argument is similar to the one used for Source-Consistency but we replace the Proximity tester by the Inclusion tester (implicit in the Equivalence tester), i.e. $H_{w,\pi}$ must be ε -included in H_T . \square

In the previous example, the input w was ε -far from Query Answering, as the polytopes $H_{w,\pi}$ were not ε -included in H_T . If we replace \mathbf{K}_T by $\mathbf{K}_T' = (c(ab)^*(bb+abb)^*c^+) + ((ba)^+(ac)^*c^+)$, then w satisfies Query Answering.

3.4 Target Correction

Approximate Source Consistency decides w is ε -close to a w' such that $\mathcal{T}(w')$ is ε -close to \mathbf{K}_T . We may want to find a target word w_T in \mathbf{K}_T and close to $\mathcal{T}(w')$. The corrector introduced in [4], finds such a w_T for words and trees. It is a valuable tool which generalizes error-correction, relative to a schema.

4 Source-Consistency and Query answering on trees

Source Consistency and Query Answering generalize to trees. Let ε , a source DTD (or Schema) \mathbf{K}_S , a transducer \mathcal{T} and a target schema \mathbf{K}_T be fixed. Given a tree T of size n , we ask the same questions and prove similar results. The technique uses the tree embedding introduced in [8] which associates a \mathbf{u} -stat vector to an unranked labelled tree.

Every unranked tree T can be coded as a binary tree $e(T)$. Consider the classical Rabin encoding where each node v of the unranked tree is a node v in the binary encoding, the left successor of v in the binary tree is its first successor in the unranked tree, the right successor of v in the binary tree is its first sibling in the unranked tree.

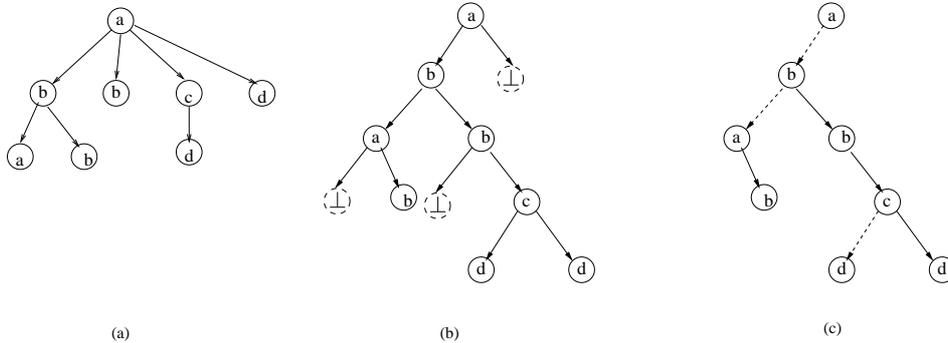


Figure 7: Encoding of an unranked tree in (a) as a binary tree with \perp in (b) and as an extended 2-ranked tree in (c).

New nodes with labels \perp are added to complete the binary tree when there are no successor or no sibling in the unranked tree. If we remove the leaves labelled with \perp , we consider trees with degree at most 2, where nodes may only have a left or a right successor and call these structures *extended 2-ranked trees*. The advantage of this representation is that the extended 2-ranked tree has the *same set of nodes* as the unranked tree.

The k -compression of a tree T , first transforms it into a 2-extended tree, then removes every node whose subtree has size $\leq k$, and finally encodes the removed subtrees into the labels of their ancestor nodes, as in figure 8. This compression leads naturally to a word $w(T)$ that encodes T such that $\mathbf{u}\text{-stat}(w(T))$ can be approximately sampled from samples on T . The basic testers on words such as Membership and Equivalence can be extended to trees, but the construction of the polytopes are more complex. We then obtain:

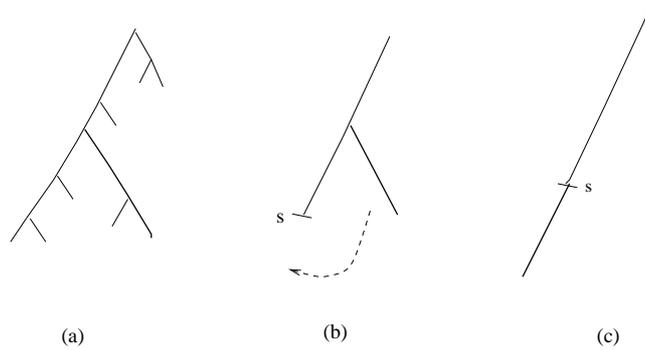


Figure 8: k -compression and word embedding.

Theorem 3. *Source Consistency and Approximate Query Answering are testable on trees.*

5 Conclusion

We introduced a model of transducer for trees with attributes which allows to compare schemas over different alphabets. We defined *Approximate Source Consistency* and *Approximate Query Answering* for this setting, using the Edit distance with Moves. We used the basic testers for Membership and Equivalence, i.e. randomized algorithms which approximately decide these problems by only looking at a constant fraction of the inputs, and proved that Source Consistency and Query Answering are testable on words and trees.

References

- [1] N. Alon, M. Krivelich, I. Newman, and M. Szegedy. Regular languages are testable with a constant number of queries. *SIAM Journal on Computing*, 30(6), 2000.
- [2] A Apostolico and Z. Galil. Pattern matching algorithms, chapter 14: Approximate tree pattern matching. *Oxford University Press*, 1997.
- [3] Marcelo Arenas and Leonid Libkin. Xml data exchange : Consistency and query answering. In *Principles of Database Systems*, pages 13–24, 2005.
- [4] U. Boobna and M. de Rougemont. Correctors for XML data. In *XSym*, pages 97–111, 2004.
- [5] A. Broder. On the resemblance and containment of documents. In *Compression and Complexity of Sequences*, page 21, 1997.
- [6] G. Cormode and S. Muthukrishnan. The string edit distance matching problem with moves. In *Symposium On Discrete Algorithms*, pages 667–676, 2002.
- [7] Ronald Fagin, Phokion G. Kolaitis, Renee J. Miller, and Lucian Popa. Data exchange: Semantics and query answering. In *International Conference on Database Theory*, pages 207–224, 2003.
- [8] E. Fischer, F. Magniez and M. de Rougemont. Approximate Satisfiability and Equivalence. In *Logic in Computer Science*, pages 421–430, 2006.
- [9] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45(4):653–750, 1998.

- [10] F. Magniez and M. de Rougemont. Property testing of regular tree languages. In *International Conference on Automata Languages and Programming (ICALP)*, pages 932–944, 2004.
- [11] W. Martens and F. Neven. Frontiers of tractability for typechecking simple xml transformations. In *Principles of Database Systems*, pages 23–34, 2004.
- [12] W. Masek and M. Paterson. A faster algorithm for computing string edit distance. *Journal of Computer and System Sciences*, 20(1):18–31, 1980.
- [13] Rohit J. Parikh. On context-free languages. *Journal of the ACM (JACM)*, 13(4):570–581, 1966.
- [14] R. Rubinfeld and M. Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):23–32, 1996.
- [15] D. Shapira and J. Storer. Edit distance with move operations. In *Proceedings of Symposium on Combinatorial Pattern Matching*, volume 2373 of *Lecture Notes in Computer Science*, pages 85–98. Verlag, 2002.
- [16] K. C. Tai. The tree-to-tree correction problem. *Journal of the Association for Computing Machinery*, 26:422–433, 1979.
- [17] J. W. Thatcher. Characterizing derivation trees of context-free grammars through a generalization of finite automata theory. *Journal of Computer and System Sciences*, 1:317–322, 1967.
- [18] R. Wagner and M. Fisher. The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21:168–173, 1974.