Approximate Data Exchange

Michel de Rougemont*[1] and Adrien Vieilleribière**[2]

[1] University Paris II & LRI CNRS
[2] University Paris-Sud & LRI CNRS

Abstract. We introduce *approximate data exchange*, by relaxing classical data exchange problems such as Consistency and Typechecking to their approximate versions based on Property Testing. It provides a natural framework for consistency and safety questions, which first considers approximate solutions and then exact solutions obtained with a Corrector.
We consider a model based on transducers of words and trees, and study ε-Consistency, i.e., the problem of deciding whether a given source instance I is ϵ-close to a source I', whose image by a transducer is also ε-close to a target schema. We prove that ε-Consistency has an ε-tester, i.e. can be solved by looking at a constant fraction of the input I. We also show that ε-Typechecking on words can be solved in polynomial time, whereas the exact problem is PSPACE-complete. Moreover, data exchange settings can be composed when they are close.

1 Introduction

Data exchange considers the situation when a source send information to a target and respects a target schema and specific constraints which link the source and the target structures. Fagin et al. [7] consider relational structures with a source schema, a target schema, constraints associated with tuple-generated-dependencies, and define the *Existence-of-Solution* problem. Arenas and Libkin [3] extend the framework to ordered and unordered trees where schemas are regular properties, and constraints are Source to Target dependencies.
We introduce *approximate data exchange* which applies to classical data-exchange but also to the situation where sources may be imperfect. Let Source-consistency be the decision problem, where given a data-exchange setting and an input structure I, we decide if there is a solution J in the target schema. We define the ε-Source-consistency problem by relaxing the source instance I to an I' close to I and a solution J' for I' close to the schema. We consider the special case when a transducer provides the solution and show that this problem is simpler than its exact version. Given an instance source I in the source schema K_S, we also find in linear time a target instance J in the target schema, using a Corrector. If the Source is imperfect because of noise, we still provide a solution which maybe

* mdr@lri.fr, work supported by ACI *Vera* of the French ministry of research.
** vieille@lri.fr

approximate or exact. The ranking of Web data, i.e. to determine if XML documents are close or far from predefined DTDs, is a typical application of this problem. The Transducer may provide a translation from tags in foreign languages into english tags used by the DTDs, and we determine if each document is ε-Source-consistent to the setting defined by the Transducer and a DTD.

Property Testing is a framework for approximate decision problems, which considers classes of finite structures with a distance between structures. Given a parameter $0 \leq \varepsilon \leq 1$, an *$\varepsilon$-tester* [15, 11] 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 is *testable* if for all ε, there exists an ε-tester whose time complexity is independent of the size of the structure and only depends 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. Although we use an approximate version of data-exchange, we may provide an exact target structure (the Target-search problem) in linear time, when the source is ε-consistent.

The main results of the paper use the *Edit Distance with Moves* on words and trees, and a transducer model to link source and target instances. Under these hypotheses, we show that many approximate data exchange problems can be efficiently solved.

- ε-Source-consistency on words and trees, i.e. the problem to decide if a given source instance I is ε-close to a source I' such that its image by the transducer is ε-close to the target schema, has a tester, i.e. can be solved by looking at a constant fraction of the input I.
- ε-Target composition on words and trees can be solved in linear time, for ε-composable settings, i.e. when the target schema of the first setting is closed to the source schema of the second setting.

These results are based on the testers and correctors for words (resp. trees) introduced in [12] for regular words (resp. regular trees) and depend on this specific *Edit Distance with Moves*. We also use the embedding of words and trees into statistical vectors introduced in [9] 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 for ε-Target search and ε-composition.

Although the main motivation is on trees, we first study these problems on words, as unranked trees are coded as binary trees with the Rabin encoding, and then coded as data words. The statistical representation of a tree is the statistical embedding of its encoded data word.

In section 2, we review the basic data exchange models, and define approximate data exchange problems. We study some of these problems on words in section 3, and on trees in section 4, in the special case when the data exchange setting uses transducers.

2 Preliminaries

2.1 Data Exchange

Let $\mathbf{K}$ be a class of finite structures, $\mathbf{K_S}, \mathbf{K_T} \subseteq \mathbf{K}$ two subclasses, and a binary relations $\mathcal{T}$ between structures $I \in K_S$ and $J \in K_T$. A pair of structures (I, J) satisfies a *data exchange setting* $\Delta = (\mathbf{K_S}, \mathcal{T}, \mathbf{K_T})$ if $I \in \mathbf{K_S}$, $J \in \mathbf{K_T}$ and $(I, J) \in \mathcal{T}$. The motivation is to view a source instance I as a finite model of a schema $\mathbf{K_S}$ and study which possible target instances J can be found such that they are models of a target schema $\mathbf{K_T}$ and satisfy a relation $\mathcal{T}$ between the two schemas. Such possible target instances are called solution for I with respect to the data exchange setting and are written $Sol_\Delta(I)$.
Fagin and al. [7] studied the relational case and defined several decision problems, given a *data exchange setting* $\Delta = (\mathbf{K_S}, \mathcal{T}, \mathbf{K_T})$ where $\mathbf{K_S}$ is a source schema, $\mathbf{K_T}$ the target schema and $\mathcal{T}$ is specified by formulae Ψ called tuple- dependencies. Arenas and Libkin [3] studied the case of trees when the schemas are regular tree languages and $\mathcal{T}$ is specified by source-to-target dependencies. In both cases the relation $\mathcal{T}$ is defined as the set of pairs (I, J) such that $(I, J) \models \wedge_{\Psi \in \mathcal{T}} \ \Psi$.
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 unordered trees, $I = (D, Str, \text{Child}, r, L, \lambda)\}$
- On ordered trees, $I = (D, Str, \text{Firstchild}, \text{Nextsibling}, r, L, \lambda)\}$

where r is the root of the tree, $L : D \to \Sigma$ defines the node label, and $\lambda : D \times A \to Str$ 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. Target schemas $\mathbf{K_{\Sigma,A}}$ may have their value set $Str_T = Str \cup V$ for $V \subseteq Var$ and Var is a fixed infinite set of values distinct from Str, called nulls.

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

(a) Let $\Sigma = \{0, 1\}$, $A = \emptyset$, $D = \{1, ..., n\}$, $Str = \emptyset$, $\mathbf{K_1} = \{I = (D, Str, \text{Child}, r, L, \lambda)\}$ where Child in the natural successor over D and $L : D \to \{0, 1\}$. 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 $\lambda(2, A_1) = a$.

(b) Let $\Sigma = \emptyset$, $A = \{A_1, A_2, A_3\}$, $D = \{1, ..., n\}$, 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.

(c) Let $\Sigma = \{0, 1\}$, $D = \{d_1, ..., d_n\}$, $A = \emptyset$,$Str = \emptyset$. The class of unranked ordered trees with n nodes without attributes is given by

$$\mathbf{K_3} = \{I = (D, Str, \text{Firstchild}, \text{Nextsibling}, L, \lambda)\}$$

2.2 Transformations

Two main approaches can be 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 logical formulas linking source and target structures. In the first case, a deterministic transducer without null transitions associates a unique J, whereas in the second case there may be infinitely many J.

Transducers. 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 [13], 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 $A_S = \{A^1, ..., A^k\}$, 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 $Str \cup Var$. 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 1. *A* tree transducer *between (Σ_S, A_S) trees and (Σ_T, A_T) trees is defined by (Q, q_0, δ, h, μ) where:*

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

The function h extends to a function $h' : \Sigma_S \times Q \times Str \to Str \cup Var$ 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 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:

– $\delta(l,q) = (t_1, t_2, ..t_s)$ 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 [13], where duplications and deletions are allowed.
– Let v a node with attribute values $x_1, ..., x_k \in Str$. The function h extends to a function h' which determines if the value is kept or assigned to a Null value.
– If $\mu(l, q, A_T^1, w) = i$ 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 μ 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.

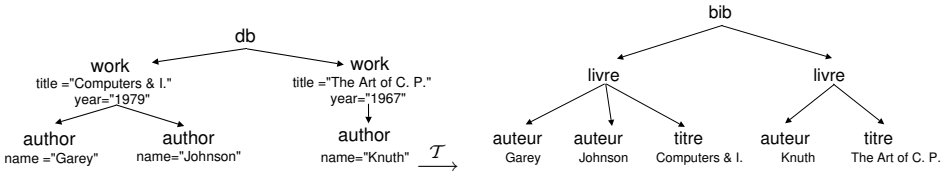
Example 2. Linear transduction on strings with attributes. Let $\Sigma_S = \{0,1\}$, $A_S = \{N, M\}$, $D = \{1, ..., n\}$, $\mathbf{K_S} = \{I = (D, Str, \text{Child}, r, L, \lambda)\}$ where Child in the natural successor over D and $L : D \to \{0,1\}$, as in example 1 of binary words. Let $\Sigma_T = \{a, b, c\}$, $A_T = \{P\}$, and the corresponding $\mathbf{K_T}$, defined by the transducer (Q, q, δ, h, μ):

– $Q = \{q\}$, $\delta(0,q) = c.d[q]$, $\delta(1,q) = b.d[q]$, i.e. words with only one successor,
– for all l, q, $\quad h(l,q,M) = 1$, $h(l,q,N) = V_1$. 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 c of the word $c.d$ with the value $@M$ of the attribute M.

The image of the word on the label set $\{0,1\}$, with attributes N, M defined by $1.0_{[N=a]}.1_{[N=c,M=c]}.0_{[M=c]}.1.0_{[N=a]}$ is a word on the label set $\{a,b,c\}$ with attribute P and attribute values in $Str \cup Var$, i.e. $b.d.c_{[P=a]}.d.b.d.c_{[P=V_1]}.b.d.c.d$

In practice, the transducer is deterministic and defined by an XSLT program π.

Example 3. Consider the following Source and Target XML structures associated with an XSLT transducer.



The transducer takes the *db* tree as input and produces the *bib* tree as output.

Logic based transformations. For a source instance I and a target instance J, a logical specification of source to target dependencies (tgd) is typically

given by formulae Ψ where a notion of satisfaction for $(I, J) \models \Psi$ is defined. In this model, $\mathcal{T}_{STD}(I) = Sol_{\Delta}(I)$ is understood as $\{J : \ (I, J) \models \Psi_i, \ \ \forall \Psi_i \in \mathcal{T}\}$
A Data Exchange setting $\Delta = (\mathbf{K_S}, \mathcal{T}, \mathbf{K_T})$ is specified by a Source schema $\mathbf{K_S}$, a Target Schema $\mathbf{K_S}$, and the relation $\mathcal{T}$.
A *Tree pattern formula* [3] is a formula $\varphi_S(x, y) \rightarrow \psi_T(x, z)$ defined by a conjunctive formula $\varphi_S(x, y)$ in the language of $\mathbf{K_S}$ with free variables for attribute values and a conjunctive formula $\psi_T(x, z)$ in the language of $\mathbf{K_T}$ with free variables for attribute values. Define $(I, J) \models \varphi_S(x, y) \rightarrow \psi_T(x, z)$ if a node v in I is such that $I, v \models \varphi_S(x, y)$ there exists a node v' in J such that $J, v' \models \psi_T(x, z)$. Such formulas Ψ_i are called STDs for Source to Target Dependencies.

Example 4. Some STDs lead to transducers. On a relational setting: $R(x, x, y) \rightarrow T(x, y)$. On an XML setting: $bd[book(@title = x[author(@name = y)]] \rightarrow bib[livre(@auteur = y, @titre = x)]$

In this approach, the settings do not compose, if we strictly rely on first-order formulas [8]. The search for minimal target structures may lead to structures much smaller then the sources, using optimization techniques, and therefore which may be far from target structures obtained by transductions.

Main Data Exchange problems. Let Δ be a data exchange setting where the relation $\mathcal{T}$ is arbitrary and I a source instance.

- *Source-consistency* decides if a source instance I given as input is such that there is $J \in \mathcal{T}(I)$ satisfying $J \in \mathbf{K_T}$.
- *Target-search* takes a source instance I given as input and produces a target structure J as output such that $(I, J) \in \mathcal{T}$ and $J \in \mathbf{K_T}$.
- *Typechecking* or *Safeness* decides if a data exchange Δ given as input is such that for all $I \in \mathbf{K_S}$, $\mathcal{T}(I) \subseteq \mathbf{K_T}$.
- *Boolean Query Answering* decides if an instance I is such that all J such that $(I, J) \in \mathcal{T}$ satisfy a boolean query, i.e. a subclass $\mathbf{K_T^Q}$ on the target structures.

The *Source-consistency*, *Target-search* become simple (linear time) for deterministic transducers, regular words and regular trees, as we only check if $\mathcal{T}(I) \in \mathbf{K_T}$. The typechecking problem is PSPACE complete on words and NEXPTIME complete on trees, as a function of the size of the regular expression or of the DTD [16]. For two settings $\Delta_1 = (\mathbf{K_S^1}, \mathcal{T}^1, \mathbf{K_T^1})$ and $\Delta_2 = (\mathbf{K_S^2}, \mathcal{T}^2, \mathbf{K_T^2})$, we wish to compose the Target-search problems and find a new setting $\Delta = (\mathbf{K_S}, \mathcal{T}, \mathbf{K_T})$ such that $Sol_{\Delta}(I) = \{ \ J \ : \ \exists \ J', J' \in Sol_{\Delta_1}(I) \wedge J \in Sol_{\Delta_2}(J')\}$.

2.3 Property Testing and Approximation

Property Testing has been initially defined in [15] and studied for graph properties in [11]. It has been successfully extended to various classes of finite structures, such as words where regular languages are proved testable [2, 1] for the Hamming distance, and trees where regular tree languages are proved testable

[12] for the Edit Distance with Moves. A tester decides if an input structure satisfies a property or is far from this property by looking at a constant fraction of the input, independent of the global size of the input.
We say that two unary structures $U_n, V_m \in \mathbf{K}$ such as words and trees, whose domains are respectively of size n and m, are *ε-close* if their distance $\mathsf{dist}(U_n, V_m)$ is less than $\varepsilon \times \max(n, m)$. They are *$\varepsilon$-far* if they are not ε-close. The distance of a structure U_n to a class $\mathbf{K}$ is $\mathsf{dist}(U_n, \mathbf{K}) = Min_{V \in \mathbf{K}}\{\mathsf{dist}(U_n, V)\}$. 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, such as binary structures or graphs, one may define the closeness relatively to the representation size (e.g. $\varepsilon.n^2$ for graphs) instead of the domain size.

Definition 2. *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 every sufficiently small $\varepsilon > 0$, there exists an ε-tester whose time complexity depends only on ε.

Definition 3. *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.*

Let the *Edit distance with moves* between two strings w and w', written $\mathsf{dist}(w, w')$, the minimal number of elementary operations on w to obtain w', divided by $max\{|w|, |w'|\}$. An *elementary operation* on a word w is either an *insertion, a deletion* of a letter, a *modification* of a letter or of an attribute value, or a *move* of a subword of w into another position.
Let the *Edit distance with moves* between two ordered unranked trees T and T', written $\mathsf{dist}(T, T')$, the minimal number of elementary operations on T to obtain T', divided by $max\{|T|, |T'|\}$. An *elementary operation* on a tree T is either an *insertion, a deletion* of a node or of an edge, a *modification* of a letter (tag) or of an attribute value, or a *move* of an entire subtree of T into another position. When an XML file is given by its DOM representation, these operations take unit costs.
Approximate Schemas. We first consider classes of structures on the same language $\mathcal{L}$, i.e. with the same alphabet Σ and attibutes A.

Definition 4. *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 $\equiv_\varepsilon$, to $\mathbf{K_2}$, if both $\mathbf{K_1}$ is ε-contained in $\mathbf{K_2}$ and $\mathbf{K_2}$ is ε-contained in $\mathbf{K_1}$.*

Example 5. 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.

A statistical embedding on strings and trees . For a finite alphabet Σ and a given ε, let $k = \frac{1}{\varepsilon}$. Let the $\mathsf{dist}(w, w')$ be the *Edit distance with moves* 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. Let w be a word of length n, let $\#u$ be the number of occurrences of u of length k in w and:

$$\mathsf{ustat}(w)[u] \stackrel{\text{def}}{=} \frac{\#u}{n-k+1}$$

The vector $\mathsf{ustat}(w)$ is of dimension $|\Sigma|^k$ is also the *probability distribution* that a uniform random subword of size k of w be a specific u, i.e.

$$\mathsf{ustat}(w)[u] = \Pr_{j=1,\ldots,n-k+1}[w[j]w[j+1]\ldots w[j+k-1] = u]$$

This embedding is a generalized Parikh mapping [14] and is also related to [5], where the subwords of length k were called *shingles*. The statistical embedding of [9] associates a statistics vector $\mathsf{ustat}(w)$ of fixed dimension with a string w and a union of polytopes H in the same space to a regular expression r, such that the distance between two vectors (for the L_1 norm) is approximately $\mathsf{dist}(w, w')$ and the distance between a vector and a union of polytopes is approximately $\mathsf{dist}(w, L(r))$. Other embeddings on words [6] and trees [10] depend on the size of the structures.

Example 6. For a lexicographic enumeration of the length 2 binary words, $w = 000111$, $r_1 = 0^*1^*$, $r_2 = (001)^*1^*$, $k = 2$,
$\mathsf{ustat}(w) = \begin{pmatrix} 2/5 \\ 1/5 \\ 0 \\ 2/5 \end{pmatrix}$.*Let* $s_0 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, s_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, s_2 = \begin{pmatrix} 1/3 \\ 1/3 \\ 1/3 \\ 0 \end{pmatrix}$
$H_1 = \text{Convex} - \text{Hull}(s_0, s_1)$ is the polytope associated with r_1 and $H_2 = \text{Convex} - \text{Hull}(s_1, s_2)$ the polytope associated with r_2.

These techniques yield the simple testers of [9] for:

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

– Equivalence tester between two regular expressions r_1 and r_2. Associate the polytopes H_1 and H_2 in the same space as $\mathsf{ustat}(w)$, 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.

The membership tester is polynomial in the size of the regular expression (or non-deterministic automaton) whereas it was exponential in this parameter in [12]. In this paper, we use the Membership tester and the Equivalence tester.

Unranked trees can be coded as binary trees with the Rabin encoding, i.e. ordered trees of degrees at most 2. We define the k-compression of a tree T, as in figure 1. We remove every node whose subtree has size $\leq k = 1/\varepsilon$, and encode the removed subtrees into the labels of their ancestor nodes. The resulting tree has at most $\varepsilon.n$ nodes with two successors, as most of the nodes have only one successor. Using at most $\varepsilon.n$ moves, we end up with a word with labels $w(T)$ that encodes T such that $\mathsf{ustat}(w(T))$ can be approximately sampled from samples on T. The previous results on words are extended to trees trough this encoding.

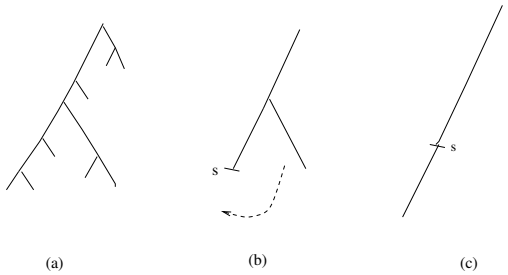


Fig. 1. Binary tree in (a), its k-compressed form in (b) and its word embedding in (c).

Approximate Data Exchange. Consider a distance dist between structures of a class $\mathbf{K}$. We can consider the ε-approximate version of the classical data exchange problems. Let Δ be a data exchange setting, I a source instance and a parameter ε.

- ε-*Source-consistency* decides if a source instance I given as input is ε-close to a source I' such that there exists J' ε-close to $\mathbf{K_T}$ and $(I', J') \in \mathcal{T}$.
- ε-*Target search* computes, given a source instance I as input, a target instance $J \in \mathbf{K_T}$ which is ε-close to $\mathcal{T}(I)$.
- ε-*Typechecking* or ε-*Safeness* decides if a data exchange Δ given as input is such that for all $I \in \mathbf{K_S}$, $\mathcal{T}(I)$ is ε-close to $\mathbf{K_T}$.
- ε-*Boolean Query Answering* decides if an instance I is ε-close to a source I' such that all J' such that $(I', J') \in \mathcal{T}$ are ε-close to a subclass $\mathbf{K_T^Q}$ on the target structures.

We took the most liberal definitions where we allow approximations on the source instance I and on the solution J. We could restrict these definitions and allow asymetric approximations. We consider the *special case of Transducers*, as a first

step, and show that ε-*Source-consistency* can be decided in $O(1)$, whereas the exact version is decided in $O(n)$. We give a condition to compose such data exchange settings, based on similar approximations: only close schemas can be composed.

Definition 5. *Let $\varepsilon \geq 0$. Two settings $\Delta^1 = (\mathbf{K_S^1}, \mathcal{T}^1, \mathbf{K_T^1})$ and $\Delta^2 = (\mathbf{K_S^2}, \mathcal{T}^2, \mathbf{K_T^2})$, are ε-composable is they are ε-safe and if $\mathbf{K_T^1}$ is ε-close to $\mathbf{K_S^2}$.*

We will show how to compose settings, using correctors. We will nest the transducers with the correctors and obtain an ε-composition.

3 Approximate data exchange on strings

In this section a data exchange setting is defined with a deterministic transducer and we consider the approximate Data Exchange problems when Schemas are regular.

3.1 ε-Source-consistency.

We present an ε-Tester for ε-Source-consistency, first for the case of a transducer with one state, and generalize it in a second step.

Example 7. Let Δ^1 be defined as in example 3. The setting Δ^1 is not consistent as $\mathcal{T}^1$ does not output the character c. For a given instance such as 0001111, $\mathcal{T}^1(I) =$ *ababab.aaaa* is at distance $\frac{1}{5}$ from the target schema $c(ab)^*ca^*$. A corrector for $\mathbf{K_T}$ will transform *ababab.aaaa* into *c.ababab.c.aaaa* in linear time.

Notice that we cannot apply a direct approach where we would test if I is ε-close to $\mathcal{T}^{-1}(\mathbf{K_T}) \cap \mathbf{K_s}$, as the distances are not kept by the inverse transformation. The Tester follows a simple *sampling approach*. It samples I to obtain a random subword u and estimate $\mathsf{ustat}(I)$. We then look at $\mathcal{T}(u)$ and obtain a subword v of $\mathcal{T}(I)$, which we will sample with specific probabilities. As the transducer produces words of different lengths, we have to adjust the sampling probabilities to guarantee a uniform distribution on $\mathcal{T}(I)$.

Transducer with one state. Let $k = 1/\varepsilon$, $\alpha = \min_{a \in \Sigma_s} |\mathcal{T}(a)| > 0$, $\beta = LCM_{a \in \Sigma_s} |\mathcal{T}(a)|$, i.e. the Least Common Multiplier of the lengths $\mathcal{T}(a)$.

Tester$_1(w, k, N)$:
1. Repeat until N outputs are generated.
{ Choose $i \in_r \{1, ..., n\}$ uniformly, let $a = w[i]$ and $\gamma = |\mathcal{T}(a)|$,
choose $b \in_r \{0, 1\}$ whith $Prob(b = 1) = \frac{\gamma}{\beta}$,
If (b=1) { Choose $j \in_r \{0, ..., \gamma - 1\}$ uniformly and
output the subword of length k begining at position j in $\mathcal{T}(a)$ and
continuing with k letters on the right side }
}.
2. If the geometrical distance between $\widehat{ustat}$ and $H(\mathbf{K_T})$ is greater than ε then reject else accept.

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

Lemma 1. *For any data exchange setting Δ, $\varepsilon > 0$ and $N \geq N_0$, Tester$_1(w, k, N)$ is an ε-tester which decides if a source word w is ε-Source consistent with respect to Δ.*

General Transducer. Let $\mathcal{A}$ be a deterministic transducer with m states. We generalize the previous tester, and sample subwords u of w which yield subwords of $\mathcal{T}(w)$. We do not know however the state q in which the transducer is, on sample u. We will consider all the possible states $S_u \subseteq Q$ but need to make sure that the possible states of the samples, i.e. $S_{u_1}, ...S_{u_N}$ are compatible, i.e. can be obtained by paths which meet only a set Π of connected components of the automaton. We will enumerate all possible such Π in the acyclic graph defined by the connected components, and apply a Tester which tests if w is close to a word w' such that a run on w' starting in some state of Π meets only the connected components of Π.
A connected component of $\mathcal{A}$ is a set S of states such that for each pair (s, s') of states of S there is a path from s to s' in $\mathcal{A}$. Let $\mathcal{G}$ be the directed acyclic graph (DAG) associated with the connected components of $\mathcal{A}$, i.e. a node is a connected component C_i and there is an edge from C_i to C_j if there is a path in $\mathcal{A}$ from a state $s \in C_i$ to a state $s' \in C_j$

Definition 6. *A set Π of connected components is admissible for $\mathcal{A}$ if there exists a word x and a state q in one of the connected components of Π such that the run from q on x meets exactly the connected components of Π. Such a word x is called Π-compatible from q.*

A word w is ε-source consistent along Π, if there is an ε-close w' which is Π-compatible from the initial state q_0 such that $\mathcal{T}(w')$ is ε-close to $\mathbf{K}$. Let H_Π be the polytope associated to the automaton reduced to Π.

Tester$_2(w, k, N, \Pi)$:
Generate $u_1...u_N$ words of length k in w.
Estimate $\widehat{\mathsf{ustat}}(w)$: if it is ε-far from H_Π, reject.
If it is ε-close to H_Π, associate a set of states S_{u_i} with each u_i from which u_i is Π-compatible.
Apply Tester$_1(w, k, N)$ for each possible states of S_{u_i}, as $\mathcal{T}(u_i)$ is well defined.
Accept if there is choice of states such that Tester$_1(w, k, N)$ accepts, else reject.

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

Lemma 2. *For $N \geq N_0$, $\varepsilon > 0$, Tester$_2(w, k, N, \Pi)$ is an ε-tester which decides if a source word w is ε-source consistent along Π.*

Tester$_3(w, k, N)$:
Generate all Π and apply Tester$_2(w, k, N, \Pi)$.
If there is a Π such that Tester$_2(w, k, N, \Pi)$ accepts, then accept, else reject.

Theorem 1. *If $N \geq \beta \ln(1/\varepsilon) \ln(|\Sigma_T|)/(\alpha\varepsilon^3)$, $\varepsilon > 0$, Tester$_3(w, k, N)$ is an ε-tester which decides if a source word w is ε-Source-consistent.*

Intrepretation with the word Embedding. Let $\mathsf{ustat}(w)$ be the statistics vector of dimension $|\Sigma_S|^k$, associated with w. There are two tasks to perform: find a w' close to w readable by $\mathcal{T}$ and find the image $\mathcal{T}(w')$ close to $\mathbf{K_T}$. Let H_T be the target union of polytopes associated with $\mathbf{K_T}$ by the embedding. Consider the automaton associated with $\mathcal{T}$ where all states accept, when we ignore the output. Let $G_{\mathcal{T}}$ be the union of polytopes associated with the language accepted by this automaton. Each polytope corresponds to a Π.
All the polytopes of $G_{\mathcal{T}}$ which are ε-close to $\mathsf{ustat}(w)$ indicate a possible w' given by its statistics vector $\mathsf{ustat}(w')[u]$ along a fixed Π. Define the image $\mathsf{ustat}(\mathcal{T}(w'))$ relative to Π as the set of statistics vector $\mathsf{ustat}[v]$ of dimension $|\Sigma_T|^k$ which are possible images of w' by $\mathcal{T}$. This set is defined as the convex-hull C of the $\mathsf{ustat}[v]$ *limit* vectors such that there exist states $s_1.....s_p$ in Π such that for some i $\mathcal{T}(s_i, u_i) = v_j$, and $\mathsf{ustat}[v_j] = \mathsf{ustat}(w')[u_i]$ if $|v_j| = k$. If $|v_j| > k$,then the weight of $\mathsf{ustat}(w')[u_i]$ is uniformly distributed over all subwords v of length k of v_j. The set of i for which there is no s_i must be of density less than ε. The Tester is simply: If C is ε-close to H_T, then Accept, else Reject.

3.2 ε-Typechecking

The Typechecking (or Safety) problem is hard in general, as it involves the comparison of schemas, a PSPACE problem for regular schemas on strings, but an undecidable problem on context-free schemas. In [9], it is shown that ε-equivalence and ε-containment are PTIME for regular schemas and EXPTIME for context-free schemas. We then obtain:

Theorem 2. *Given $\epsilon > 0$, a transducer $\mathcal{T}$ and regular schemas $\mathbf{K_S}$ and $\mathbf{K_T}$, there exists an ε-tester which decides ϵ-safety of $(\mathbf{K_S}, \mathcal{T}, \mathbf{K_T})$, in Polynomial time.*

3.3 ε-Composition

Let $\Delta^1 = (\mathbf{K_S^1}, \mathcal{T}^1, \mathbf{K_T^1})$ and $\Delta^2 = (\mathbf{K_S^2}, \mathcal{T}^2, \mathbf{K_T^2})$ be two settings. Let $\Delta^{2\circ 1} = (\mathbf{K_S^1}, \mathcal{T}^{2\circ 1}, \mathbf{K_T^2})$ such that $\mathcal{T}^{2\circ 1} = \mathcal{C}_T^2 \circ \mathcal{T}^2 \circ \mathcal{C}_S^2 \circ \mathcal{C}_T^1 \circ \mathcal{T}^1$ where $\mathcal{C}_X^i$ is a corrector for $\mathbf{K_X^i}$. Recall that a corrector for a regular schema $\mathbf{K_X^i}$ is a deterministic linear time program wich takes a word w ε-close to $\mathbf{K_X^i}$ as input and produces a word $w' \in \mathbf{K_X^i}$ which is ε-close to w. An **ε-solution** of $I \in \mathbf{K_S^1}$ with respect to $\Delta^{2\circ 1}$ is an instance $J_{2\circ 1} \in \mathbf{K_T^2}$ such that $\exists J_T^1 \in \mathbf{K_T^1}, I_S^2 \in \mathbf{K_S^2}, \varepsilon_1, \varepsilon_2, \varepsilon_3,\ \ \varepsilon_1 + \varepsilon_2 + \varepsilon_3 \leq \varepsilon$ such that:

$$\mathsf{dist}(\mathcal{T}^1(I), J_T^1) \leq \varepsilon_1 \wedge \mathsf{dist}(J_T^1, I_S^2) \leq \varepsilon_2 \wedge \mathsf{dist}(\mathcal{T}^2(I_S^2), J_{2\circ 1}) \leq \varepsilon_3$$

Theorem 3. *Let Δ_1 and Δ_2 be two ε-composable settings. Then $J = \mathcal{T}_{2\circ 1}(I)$ is an ε-solution for I with respect to $\Delta^{2\circ 1}$.*

Proof. Let $J_1 = \mathcal{T}^1(I)$. There is a $J_T^1 \in \mathbf{K_T^1}$ which is ε_1-close to J_1. There is a $J_S^2 \in \mathbf{K_S^2}$ which is ε_3-close to J_T^1. Let $J_2 = \mathcal{T}^2(J_S^2)$. There is a $J^{2\circ 1} \in \mathbf{K_T^2}$ which is ε_2-close to J_2. Using the triangular inequality, we conclude that $J^{2\circ 1}$ a ε-Solution for I with respect to $\Delta^{2\circ 1}$.

Example 8. Let Δ^1 be defined as in example 3(a) and let $\Delta^2 = (\mathbf{K_S^2}, \mathcal{T}^2, \mathbf{K_T^2})$ such that $\mathbf{K_S^2} = a^*d(ab)^*$, $\mathbf{K_T^2} = 0^+(022)^*3$ and $\mathcal{T}^2 : a|0, b|22, d|3$. Let us apply the transformations:

$$\begin{aligned} I = 00011 &\xrightarrow{\mathcal{T}^1} ababababaa \xrightarrow{\mathcal{C}_T^1} cababababcaa \xrightarrow{\mathcal{C}_S^2} aadababababab \\ &\xrightarrow{\mathcal{T}^2} 003022022022 \xrightarrow{\mathcal{C}_T^2} 000220220223 = J_{2\circ 1} \end{aligned} \tag{1}$$

The second corrector $\mathcal{C}_S^2$ makes one move (aa moves to the end) and two deletions (c are removed) and $\mathcal{C}_T^2$ makes one move ("3" moves to the end). Δ^1 and Δ^2 are ϵ-composable and therefore this sequence of operations guarantees that the result is in the target shema and is obtained by few corrections.

4 Approximate data exchange on trees

The basic results on approximate data exchange on words generalize to unranked ordered trees via a coding of a tree T as 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. 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 Edit distance with moves on unranked trees is equivalent to the same distance on these extended 2-ranked trees, on which we apply the k-compression and obtain a word $w(T)$.

4.1 ε-Source-consistency via the tree embedding

Let $\mathsf{ustat}(T)$ be the statistical vector of T defined by the tree embedding, i.e. the statistics vector of the word $w(T)$, obtained in the compression described in section 2.

From the $\mathsf{ustat}(T)$ vector, we have to find a close $\mathsf{ustat}'(T)$ vector whose image by the transducer has a statistics vector close to the polytope H_T of the target schema $\mathbf{K_T}$ defined by the embedding. Consider a grid of step ε on each dimension, and the set I_ε of $\mathsf{ustat}'(T)$ points on the grid which are ε-close to $\mathsf{ustat}(T)$ and to some polytope of $H_\mathcal{T}$ as in the case of words.

For each of the points of I_ε, construct its image C by $\mathcal{T}$ with set of connected component Π, determined with one of the polytopes of $H_\mathcal{T}$. C is the convex-hull of the $\mathsf{ustat}[v]$ *limit* vectors in the target space, such that there exist states $s_1.....s_{|\Sigma_S|^k}$ in Π such that: (i) for most i, $\mathcal{T}(s_i, u_i) = v_j$, and $\mathsf{ustat}[v_j] = \mathsf{ustat}(w')[u_i]$ if $|v_j| = k$. If $|v_j| > k$,then the weight of $\mathsf{ustat}(w')[u_i]$ is uniformly distributed over all subwords v of length k of v_j. (ii) the fraction of i for which there is no such s_i is of density less than ε, i.e. $\sum_i \mathsf{ustat}[u_i] \leq \varepsilon$.

Given a point $\mathsf{ustat} \in I_\varepsilon$ the test is: If C is ε-close to H_T, then Accept, else Reject this point. We need the following lemmas:

Lemma 3. *If there is a point in I_ε such that its C is ε close to H_T, there exists a T' ε-close to T such that its image is ε-close to $\mathbf{K_T}$.*

Lemma 4. *If C is ε far to H_T for all points of I_ε, T is not ε-close to a T' whose image is ε-close to $\mathbf{K_T}$.*

Tree Consistency Tester(T, k) :
Generate all points of I_ε.
{For each point I_ε and compatible Π, compute C.
If C is close to H_T accept,}
Else reject.

Theorem 4. *For all $\varepsilon, k = 1/\varepsilon$, Tree Consistency Tester(T, k) is an ε-tester for the ε-Source consistency problem on trees.*

4.2 ε-Source-consistency via sampling

The Tester follows an approach, similar to the one presented for strings and uses the Membership ε-Tester presented in [12]. It samples I to obtain a random subtree t with a uniform distribution. We look at $\mathcal{T}(t)$ and obtain a subtree t' of $\mathcal{T}(I)$. As the transducer produces trees of different sizes, we have to adjust the sampling probabilities to guarantee the near-uniform distribution of t'. A random subtree of size k in an unranked tree is defined through the encoding of an unranked tree as as an extended 2-ranked tree.

Theorem 5. *For any data exchange setting Δ, and $\varepsilon > 0$, there is an ε-tester which decides if a source tree T is ε-Source consistent with respect to Δ.*

4.3 ε-Typechecking ε-Composition

The image of a regular by a transducer is regular for linear transducers but not in general. In this case, we generalize the methods introduced in section 3.

Theorem 6. *For any data exchange setting Δ with a linear transducer, and $\varepsilon > 0$, ε-Safety with respect to Δ can be decided in time exponential in the representation.*

Theorem 7. *Let Δ_1 and Δ_2 be ε-composable settings $J = \mathcal{T}_{2\circ1}(I)$ is an ε-Solution for I with respect to $\Delta^{2\circ1} = (\mathbf{K_S^1}, \mathcal{T}^{2\circ1}, \mathbf{K_T^2})$.*

5 Conclusion

We introduced a framework for approximate data exchange and considered the *ε-Source-consistency*, *ε-Typechecking* and *ε-Composition* problems in the special case of deterministic transducers.

We showed that for the *Edit distance with Moves*, ε-*Source-consistency* is testable on words and trees, and that ε-*Typechecking* is polynomial on words.
We need to generalize this approach to transformations which may generate an infinite set of solutions, for example with transducers with null transitions and more generally with Logic-based transformations.

References

1. N. Alon, E. Fischer, M. Krivelevich, and M. Szegedy. Efficient testing of large graphs. *Combinatorica*, 20:451–476, 2000.
2. 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.
3. M. Arenas and Leonid Libkin. Xml data exchange: Consistency and query answering. In *ACM Principles on Databases Systems*, 2005.
4. U. Boobna and M. de Rougemont. Correctors for XML data. In *XML Database Symposium*, pages 97–111, 2004.
5. A. Broder. On the resemblance and containment of documents. In *Compression and Complexity of Sequences*, pages 21–29, 1997.
6. G. Cormode and S. Muthukrishnan. The string edit distance matching problem with moves. In *Symposium on Discrete Algorithms*, pages 667–676. Society for Industrial and Applied Mathematics, 2002.
7. R. Fagin, P. G. Kolaitis, R. J. Miller, and Lucian Popa. Data exchange: Semantics and query answering. In *International Conference on Database Theory*, pages 207–224, 2002.
8. R. Fagin, P. G. Kolaitis, L. Popa, and W. C. Tan. Composing schema mappings: Second-order dependencies to the rescue. In *ACM Principles on Databases Systems*, pages 83–94, 2004.
9. E. Fischer, F. Magniez, and M. de Rougemont. Approximate satisfiability and equivalence. In *Proceedings of 21st IEEE Symposium on Logic in Computer Science*, 2006.
10. M. Garofalakis and A. Kumar. Xml stream processing using tree-edit distance embeddings. *ACM Transactions on Database Systems*, 30(1):279–332, 2005.
11. 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.
12. F. Magniez and M. de Rougemont. Property testing of regular tree languages. In *International Colloquium on Automata Languages and Programming*, pages 932–944, 2004.
13. W. Martens and F. Neven. Frontiers of tractability for typechecking simple XML transformations. In *ACM Principles on Databases Systems*, pages 23–34, 2004.
14. R. J. Parikh. On context-free languages. *Journal of the ACM*, 13(4):570–581, 1966.
15. R. Rubinfeld and M. Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):23–32, 1996.
16. L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time(preliminary report). In *ACM Symposium on Theory of Computing*, pages 1–9, 1973.