XmlCorrector
User's Guide
Version 1.0

http://www.info.univ-tours.fr/~savary/English/xmlcorrector.html

Joshua Amavi
LIFO - Université d'Orléans, Orléans, France

Béatrice Bouchou
Université François Rabelais Tours, Blois Campus, France

Agata Savary
Université François Rabelais Tours, Blois Campus, France

February 1, 2012

Contents

1 Introduction

XMLCorrector is an implementation of an algorithm allowing to correct an XML document with respect to schema constraints expressed as a DTD. Namely, given a well-formed XML document t seen as a tree, a schema S and a non negative threshold th the algorithm finds every tree t' valid with respect to S such that the edit distance between t and t' is no bigger than th. The algorithm is based on a recursive exploration of the finite-state automata representing structural constraints imposed by the schema, as well as on the construction of an edit distance matrix storing edit sequences leading to correction candidate trees. The algorithm is an extension of ideas announced in [2, 3], and is fully described in [1].

XMLCorrector is a free software. It can be used under the terms of the GNU LGPL v3 license. The current distribution consists of:

- XmlCorrector.jar: the executable file for launching the application,
- src/: the directory containing the Java source codes,
- lib/: the directory containing the external librairies used by XmlCorrector,
- doc/: the directory containing the Javadoc files (technical documentation),
- examples/: the directory contains some easy examples of DTDs and XML Documents (trees to be corrected),
- results/: the directory containing the result XML files,
- logs/: the directory containing the log files,
- COPYING, COPYING.LESSER, LICENSE.txt: the license files,
- XMLCorrector-users-guide.pdf: this user'guide.

The next section will detail the user's interface.

2 Operations on an XML tree and distance between two trees

For correcting an XML tree, we allow three kind of elementary operations on nodes:

1. Insertion of a node at a position in the tree
2. Deletion of a leaf of the tree
3. Relabeling of a node in the tree

A sequence of node operations transforms a tree t into another tree t'. Figure 1 shows an example of an initial tree t and of a tree t' resulting from t by the application of an operation sequence.

Each node operation has a cost. The cost of the node operation sequence is equal to the sum of the costs of each operation in the sequence. The distance between t and t' is defined as the minimal cost of all operation sequences allowing to transform t into t'. For instance, if we admit cost 1 for each node operation, the operation sequence in Fugure 1 has cost 3. Note that there exists non operation sequence transforming t into t' with a lower cost, thus the distance between t and t' is 3.

The aim of XMLCorrector, for a given well-formed XML document, a DTD and a threshold, is to find all correction candidates, i.e. all valid documents whose distance from the original document does not exceed the threshold. The program provides the list of all operation sequences leading to such correction candidates, as well as the resulting XML documents themeselves.

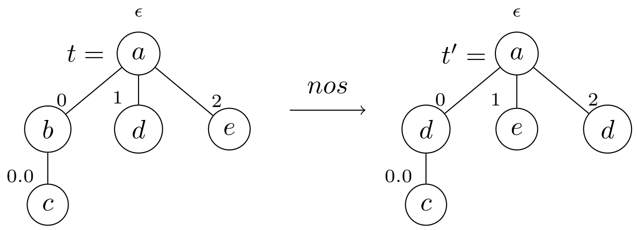
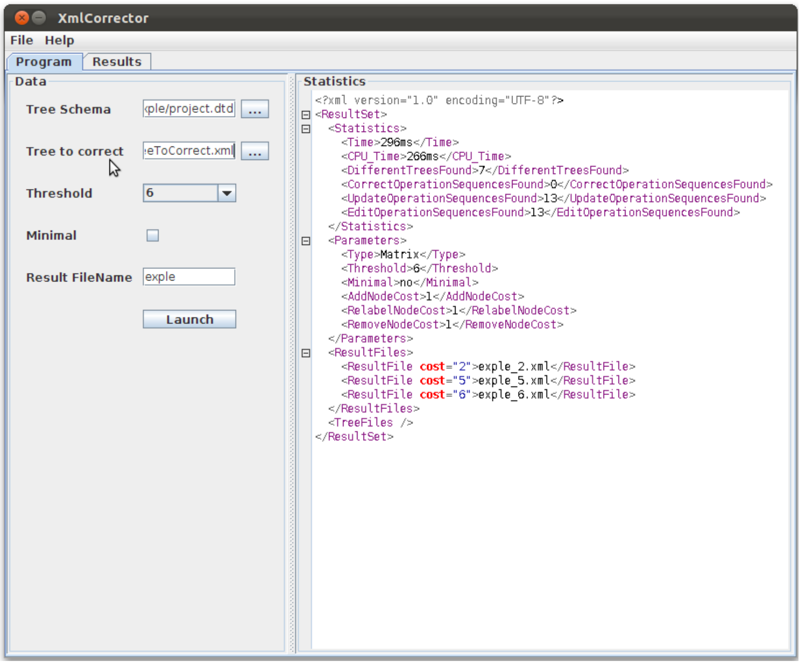


Figure 1: Application of $nos = \langle (add, 1, e), (delete, 2, /), (relabel, 0, d) \rangle$ on the tree t, $cost(nos) = 3$

3 How to use the Graphical User Interface

In order to launch the application click on the jar file `XmlCorrector.jar` or open a terminal and enter the following command: `java -jar XmlCorrector.jar`. The main window opens as shown in Figure 2.



Tree Schema: ple/project.dtd
Tree to correct: eToCorrect.xml
Threshold: 6
Minimal: ☐
Result FileName: exple

```
<?xml version="1.0" encoding="UTF-8"?>
<ResultSet>
  <Statistics>
    <Time>296ms</Time>
    <CPU_Time>266ms</CPU_Time>
    <DifferentTreesFound>7</DifferentTreesFound>
    <CorrectOperationSequencesFound>0</CorrectOperationSequencesFound>
    <UpdateOperationSequencesFound>13</UpdateOperationSequencesFound>
    <EditOperationSequencesFound>13</EditOperationSequencesFound>
  </Statistics>
  <Parameters>
    <Type>Matrix</Type>
    <Threshold>6</Threshold>
    <Minimal>no</Minimal>
    <AddNodeCost>1</AddNodeCost>
    <RelabelNodeCost>1</RelabelNodeCost>
    <RemoveNodeCost>1</RemoveNodeCost>
  </Parameters>
  <ResultFiles>
    <ResultFile cost="2">exple_2.xml</ResultFile>
    <ResultFile cost="5">exple_5.xml</ResultFile>
    <ResultFile cost="6">exple_6.xml</ResultFile>
  </ResultFiles>
  <TreeFiles />
</ResultSet>
```

Figure 2: Main Window

Fill out the left hand tab form and click on the button ”Launch”. The following data are needed to fill out the form fields[1]:

---

[1] The running example, provided in `example/projects`, has been inspired by [4].

- Tree Schema – the DTD file which contains the description of the schema. The tree schema selected in Figure 3 is the file `project.dtd`, shown in Fig. 3, provided in the directory `example/projects`. According to this DTD a project description requires: the name of the project, the manager (name and salary), and the list of employees (names and salaries). A project can have a list of sub-projects.

```
<!ELEMENT projs (proj*)>
<!ELEMENT proj (name,emp,proj*,emp*)>
<!ELEMENT emp (name,salary)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT salary (#PCDATA)>
```

Figure 3: The DTD of the XML file used for the example

- Tree to correct – the XML document to be corrected with respect to the schema mentioned above. The document selected in Figure 3 is the file `projectTreeToCorrect.xml`, shown in Fig. 4, provided in the directory `example/projects`. Note that this document is incorrect: the `<salary>` is missing for the manager of the project "Cooking Pierogies", and there is an additional `<address>` tag for the manager of the project "Preparing Stuffing".
  In order to correct an empty tree, the selected XML file should contain the tag `<EmptyTree />` only.
- Threshold – the maximum value for the edit distance between the tree to correct and the possible candidate trees. Only trees within this threshold are proposed.
- Minimal – the type of search for the correction algorithm. If the box is checked then only the trees which have the minimal edit distance are proposed. Otherwise all trees within the threshold are proposed.
- Result Filename – prefix for the name of the result files which are stored in the directory `results/`.

When the correction terminates, statistics are shown in the right hand panel of the window in Figure 2:

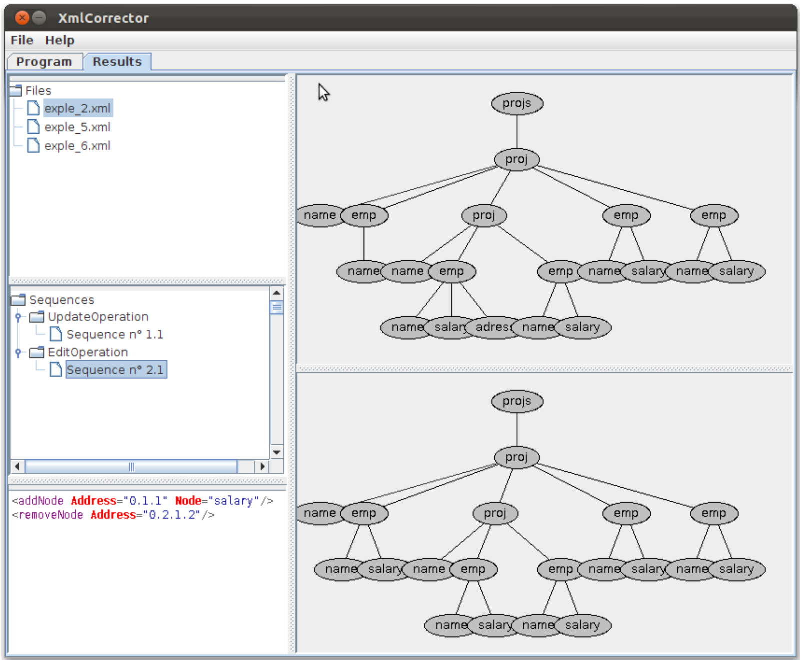
- execution time,
- number of correction candidates found,
- parameters used (the threshold value, the Boolean value for the minimal search, the cost of adding, removing or relabeling a node)
- names of result files `prefix_i.xml` (where `prefix` is the word entered for the field `Result Filename`).

```xml
<?xml version="1.0" encoding="UTF-8"?>
<projs>
   <proj>
      <name>Cooking Pierogies</name>
      <emp>
         <name>Smith</name>
      </emp>
      <proj>
         <name> Preparing Stuffing</name>
         <emp>
            <name>John</name>
            <salary>80K</salary>
            <address>London</address>
         </emp>
         <emp>
            <name>Mary</name>
            <salary>40K</salary>
         </emp>
      </proj>
      <emp>
         <name>Peter</name>
         <salary>30K</salary>
      </emp>
      <emp>
         <name>Steve</name>
         <salary>50K</salary>
      </emp>
   </proj>
</projs>
```

Figure 4: XML document which is correct

The result file `prefix_i.xml` contains all operation sequences with cost equal to i, as well as the resulting XML trees obtained by applying these sequences on the initial XML tree. We can visualize theses sequences and trees in the second tab `Results`. The upper left hand panel in Figure 5 shows the list of result files, and the XML tree to correct is displayed in upper right hand panel. After selecting one result file (here `exple_2.xml`) the list of operation sequences contained in this file is displayed below. After selecting one operation sequence in this list its detailed content is shown in the lower left hand panel and the resulting correct XML tree is displayed in the lower right hand panel.

It is possible to modify the cost of each of the three node operations (add, relabel and remove) via the menu option *File* → *Options*.



```
<addNode Address="0.1.1" Node="salary"/>
<removeNode Address="0.2.1.2"/>
```

Figure 5: Content of the `Results` tab

Acknowledgements

XMLCorrector includes the following open source libraries:

- dk.brics.automaton v1.6 under the BSD 2-Clause license, developed by Anders Møller at Aarhus University et al. (`http://www.brics.dk/automaton/`)
- jdom v1.1 under an open source license derived from the BSD 3-Clause License, developed by the JDOM Project. (`http://www.jdom.org/`)
- bounce v0.18 under the BSD 3-Clause License, developed by Edwin Dankert. (`http://www.edankert.com/bounce/`)
- DTDParser v1.21 under the GNU Lesser General Public License v3, developed by Mark Wutka. (`http://mvnrepository.com/artifact/com.wutka/dtdparser/1.21/`)

References

[1] Joshua Amavi, Béatrice Bouchou, and Agata Savary. On Correcting XML Documents With Respect to a Schema. Technical Report 301, LI, Université François Rabelais Tours, 2012. Submitted to The Computer Journal.

[2] Béatrice Bouchou, Ahmed Cheriat, Mirian Halfeld Ferrari Alves, and Agata Savary. Integrating Correction into Incremental Validation. In *Proceeding of Bases de Données Avancées (BDA 2006)*, Lille, 2006.

[3] Béatrice Bouchou, Ahmed Cheriat, Mirian Halfeld Ferrari Alves, and Agata Savary. XML Document Correction: Incremental Approach Activated by Schema Validation. In *Tenth International Database Engineering and Applications Symposium (IDEAS 2006)*, pages 228–238, 2006.

[4] Slawomir Staworko and Jan Chomicki. Validity-Sensitive Querying of XML Databases. In *EDBT Workshops*, pages 164–177, 2006.