Project P1: Quick and Dirty XML Diff

Background

In many development environements, one can call a facility named `diff` that takes as input two text files (e.g., code or proper `.txt` text files) and produces as output a description of the differences between the two files. Diff functions are offered, for instance, as part of an operation sytem or in version control systems. Diff algorithms can actually be quite sophisticated. You find a good overview under the keyword “Diff” on Wikipedia.
For detecting differences between XML documents, however, a text diff function will not be very helpful, as different text files may represent the same XML DOM tree. Note for instance that

```
<a>
    <b att1="bla" att2="blabla">
    </b>
</a>
```

and

```
<a><b att2="blabla" att1="bla"/></a>
```

are equivalent XML documents, that is, they stand for the same DOM tree.
Similarly to a `diff` for text, XML specific `xdiff` or `xmldiff` functions are parts of tools for editing and managing XML documents. What exactly they do and how they work is usually not well documented. There have been research papers in the past on this topic, however, and the researchers aimed at a conceptually elegant, comprehensive, and efficient solution.

Goal

The goal of this coursework is far more modest. You are asked to develop a “quick and dirty” application that takes as input two XML documents (without reference to a DTD), say X_1, X_2, and outputs a third XML document, say Y, that describes, in a format chosen by you, the differences between the DOM trees represented by

X_1, X_2. As a special case, your application has to detect when two documents are equivalent.
The task has both a conceptual and an implementation side. Conceptually, you have to come up with a format by which you can describe the differences between two documents. For instance, when receiving as input

```
<a><b/><c/><d/></a>
```

and

```
<a><c/><d/><e></a>
```

you could say that the two docs have the same root element, but different children, or you could say, more specifically, that they docs, have the same root element, have both children `<c/>`, `<d/>`, but the first has a `<b/>` that is missing in the second, and the second has an `<e/>` that is missing in the first.
To build up your intuition, you may want to download trial versions of tools and/or study the literatur. Some relevant publications are

- Gregory Cobena, Serge Abiteboul, Amlie Marian: Detecting Changes in XML Documents. International Conference on Data Engineering, ICDE 2002: 41-52
- Yuan Wang, David J. DeWitt, Jin-yi Cai: X-Diff: An Effective Change Detection Algorithm for XML Documents. International Conference on Data Engineering, ICDE 2003: 519-530
- Adrian Mouat. XML Diff and Patch Utilities. BSc Thesis, Heriot-Watt University, Edinburgh, 2002.

There will be copies of these papers available on the course page.
Practically, you will have to write a system that implements your conceptual solution.
For your work, keep in mind that your task is to create a "quick and dirty" solution. That means, if it finds that two documents are not equivalent, it may *not* perform a fine-grained analysis of which *parts* are still *equivalent*. Pointing to the first nodes in the tree where a difference occurs may be one of the options for your application.

Deliverables

For this coursework, you will work in pairs. Your deliverables will consist of

- a technical report in PDF format with the authors names that

  1. specifies the format in which you describe differences between XML documents,
  2. describes an algorithm that computes such difference descriptions, and
  3. describes the implementation.

  Part 1.) of the report should first explain your concepts of how to describe differences between XML trees and then specify the output format either as DTD or as XML Schema.

- your working and tested Java implementation, including source files and instructions for running it.

Please, submit your work by email to `nutt` AT `inf.unibz.it` no later than

25 November 2010, 23:30 hours.