Programming Project
DTD Validator for XML

Web Data Models

October 16th, 2017

The objective of this programming assignment is to be able to implement a program that is able to check for well-formedness and validity of a XML that is given in serialized form, using a simple DTD.

1 Preliminaries

This assignment will use the streaming format for the XML. An XML document will be given as input to the implemented algorithms. The file is a whitespace delimited file having the following format:

```
0|1 <whitespace> element1
0|1 <whitespace> element2
...
```

where `0|1` is a bit indicating a startElement or endElement event respectively, and `element` is the name of the element. For instance, the following file `example.xml` contains the document `<a><b></b><c></c><d></d></a>`:

```
0 a
0 b
1 b
0 c
1 c
0 d
1 d
1 a
```

A simplified DTD the second input to the program, with the following format, on each line and for each element in the XML:

```
element_name <whitespace> content_regular_expression
```

where `element_name` is the name of the element, and `content_regular_expression` is the regular expression that the content formed of the children nodes must match.

For example, `example.dtd` is the following:

```
a (bc)*d
b _
c _
d _
```

The parameters of the implemented executables will be the following:

```
./<program_name> <xml_file> <dtd_file>
```

The program will output two lines, one for the well-formedness check and one for the validity check. For instance:

```
./<program_name> example.xml example.dtd
```

will return the following output:

```
well-formed
valid
```

2 Assignments

2.1 Algorithm Implementation

Implement an algorithm that given an XML file and its DTD file in the formats defined above, checks whether the XML is well-formed and the DTD is valid.

The program should output on the first line either the string "well-formed" if the XML represented in the input XML is well-formed, or the string "not well-formed" otherwise. On the second line, the string "valid" should be present if the XML is valid w.r.t. the input DTD, and "not valid" otherwise. No other output should be present.

The DTD specified will conform to the following simple DTD regular expression rules:

1. the allowed symbols are the element names, `?`, `*`, `+` and `_` and the grouping `()`,
2. an element name appears only once in the content specification for an element,
3. the `_` symbol represents the empty element.

For the implementation, any external library for XML, DTD validation or automata building and validation is forbidden. You must create your own data structures and algorithms. This will be checked in the source file, and is part of the evaluation criteria.

2.2 Analysis & Experimental Evaluation

For the two algorithms implemented in the above assignment, write a document containing an implementation analysis and an experimental section.

The *implementation* analysis section will contain a write-up of the implementation choices. For example, it should contain an analysis of the data structures used, but also a discussion of the optimizations used in the implementation.

The *experimental evaluation* analysis will contain an empirical evaluation of the algorithm in the form of plots evaluating the execution time and memory space for a selection of document and DTD sizes. For instance, a possible plot has the size of the document on the x-axis and the execution time on the y-axis. The above are just suggestions: other evaluations are welcome and appreciated – if they are relevant to the implementations.

3 Submission & Evaluation

Send your submissions by email to Silviu Maniu (`silviu.maniu@lri.fr`) by **Friday, November 17th 2017, 11:59pm**, for full credit. Submissions sent by Saturday, November 18th 2017, 11:59pm will incur 5 points of penalty out of 20. Submissions received after this date will receive no credit.

Your submission should contain two files:

1. an archive (.zip, .tar, .rar, etc.) consisting of the source code and instructions on how to compile (if necessary) and execute the code, and

2. a PDF document containing the analysis and experiments document.

Your submission will be evaluated based on private tests run on your algorithms for correctness, on the evaluation of the code and implementation quality, and the quality of your evaluation document. The implementation and documentation will have equal weight in the final grade.