Student Performance Feedback for COMP60411 exam 2011/12

The exam contained a number of (autograded) multiple choice questions of different difficulties. In addition, Questions 1.1.a-1.1.d, 1.3.c, 1.6, 1.7, 2.14, and 2.15 were short answer questions.

For the differences in difficulty of the multiple choice questions, see the overview chart at the end of this paper. In particular, 2.2j.ii, 2.2, 2.10 and 2.11 were hard questions. In contrast, e.g., 1.2.d-h were easy questions that almost all students got right.

Next, we comment in detail on the short answer questions.

**Question 1.1**

Question 1 was mainly bookwork and relates to core concepts discussed at length in the class. Everybody should have found that easy – but not everybody did.

(a) surprisingly few students got the complete answer. Some claim erroneously that all validating parsers produce a PSVI, or that the parser only validates a document against an associated (!) schema. Many simply forgot to mention PSVIs at all.

(b) surprisingly few students got the complete answer. Often it wasn't made clear where the additional information in the PSVI comes from.

(c) Many students got this one more or less right.

(d) we received and accepted a variety of answers, but I was still surprised that this concept (serialization) was not understood better by more students.

**Question 1.3.c**

This question was designed as a difficult one, and it is not surprising that only 4 students got full marks and not many 1 out of 2: we saw a lot of confusion between a grammar being single-type and between the set of trees described by the grammar being single type.

**Question 1.6**

This question was designed as a difficult one, and it required a multi step line of a reasoning, and only for a correct such line that was brought forward in a coherent way did we give full marks. Most students remembered that single-type had something to do with uniqueness of the PSVI, and some knew that this in turn had something to do with query answering, but didn’t put it all together correctly.
Question 1.6

This question was designed as a modestly difficult one, but students found it very hard: we saw a lot of answers of the form ‘we need a more expressive schema language to say things we can’t say in XML schema…’, for which we didn’t give any marks because it is a vacuously true statement. Some students were able to give examples of constraints that one might want to express, and even example schema languages that can express these. Hence we saw a number of students who gained 1-2 points, but few gave more than a single valid example, and few related these to other schema languages. Finally, quite a number of students wrongly claimed that validation against a schema language more expressive than XSD would be very complex (we learned at length in class that this isn’t the case).

(The general marking scheme for 2.14 and 2.15 was 1 point for coherence/intelligibility, 2 points for basic facts, and 2 points for strength of argument.)

Question 2.14

Facts: Many people did not mention what the problem was that XML tries to solve, to wit, that it attempts to be a (universal) *external representation*. External representations have the requirements of roundtripping and self-describingness. I was generally generous with missing the external representation part. 1 point for each requirement (i.e., that it is, what it is, and how XML breaks it).

Few people discussed whether the problem was hard. While XML fails to solve the problem exactly, it’s still possible to argue that it’s failure is minor (i.e., union types are obscure; leading zeros are pointless). You could lose a point for failing to consider the opposite side (i.e., attack a defence, defend an attack).

In general, it’s not really enough to say "external validation destroys self describingness" rather than "external validation destroys self describingness because some of the information needed to convert the external representation (the XML document) to an internal representation (the desired PSVI) is outside the document (e.g., in a Schema), thus the "description" is not inherent in the document."

Some off topic responses:
- Error handling (esp. draconian)
- No mention of Postel's law was correct
- "representing semi-structured data"
- Communication between people

Most 0s were completely off topic and lacking in coherence. I counted failure to make *any* sort of case for a proposition as a coherence failure.
Most 1s were completely off topic, but reasonably coherent. If you made a sufficient (off topic) case that wasn't utterly off base you could get a 2.

Examples of utterly off base: Labeling as a "problem" the fact that XML doesn't address all use cases; saying that datatypes are a problem that XML solves.

Note that self-describingness does not mean "human intuitable". In this context, it just means we can get to the right internal representation based on the external representation alone.

**Question 2.15**

A key point to discuss is the "degree" of change between versions. Should we have namespace changes for every version, however small?

It's comparatively rare that different versions of the same element need to appear in the same document. Of course, *when* that happens, different namespaces are likely necessary for disambiguation.

Distinct namespaces per version is completely distinct from contained namespaces. There's no connection.

As emphasised in class, deception, confusingness, etc. are features of namespace *declarations*, not of namespaces themselves. There's no necessary connection between having more namespaces even in a document and being more prone to these problems.

Putting version information per se in the namespace isn't a significant advantage. I.e., if you forced them to be random strings most of the benefits (and drawbacks) of the policy would remain.

It's important to note that both documents *and applications* (can) break and need updating with new namespaces.

Change in namespace doesn't necessarily make the difference between versions (or degree of difference) clear absent some policy of "how much" change makes a version. It does let us know which version the document was encoded in. (But one could use a version id attribute.)