A TYPE-BASED TAXONOMY OF ITEMS IN ASSESSMENTS

C. Delgado Kloos, A. Pardo, P. Muñoz Merino, N. Pérez Pérez
Departamento de Ingenería Telemática, Universidad Carlos III de Madrid
{cdk,abel,pedmume,nperper}@it.uc3m.es

Abstract: The IMS Consortium in its QTI (Question & Test Interoperability) specification [1] presents a collection of question types that are useful to be used in an e-learning context. Standardization on the items and test types eases reusing them throughout LMSs, hence its importance. The IMS QTI question catalogue includes some items that are specializations of others. Others just differ in the rendering model. The goal of this paper is to categorize and put all those items in context, thus producing a new taxonomy. We do this by giving a type to each of the items. So, this work could be also understood a defining a type system for items. As a by-product, we include some new kinds of items and re-categorize others.

Key words: E-Learning, Standards, assessment, IMS QTI, type system

1. INTRODUCTION

The IMS Question and Test Interoperability specification [1] defines a catalogue of items that can be used in a variety on e-learning settings for assessment. It is good to have them defined in a standard way, because this permits the reusability of these items and their interoperability across Learning Management Systems. In IMS QTI terminology, an item consists of a question and in some cases a list of possible answers, from which the student has to select one or more. A section can contain one or more sections and one or more items. An assessment contains one or more sections. The OKI initiative [2] (which defines a behavioural model instead of a data model) also uses the same terminology for item, section and assessment. IMS QTI specifies a catalogue of items. In this paper, we study and classify them by finding a clearer taxonomy, and in particular one that separates concerns clearly. We include some additional items that are not covered by IMS QTI, but are used by several tools (some of them are Canvas Learning Course Builder [4], QuestionMark Perception [5], Respondus [6], IMS Assesst Designer [7]. Not all these tools have all types of items). The new taxonomy defined is useful both for better understanding the items, as well as for implementation purposes.

An item can be understood as a function, where the possible answers are the arguments and the evaluation, the result. Therefore, we can give a type to items. For convenience, we give them a type in a domain of polymorphic, dependent types. We use a functional-like notation. Three steps can be identified when processing items answered by a student:

1. Data validation (e.g. is the data inserted in a fill-in-the-blank item of the type expected, integer, string, etc.?). This is type checking. Additional range checking could be needed at run-time. The use of dependent types reduces this additional checking. Dependent types are types that can depend on values. They can be used to assign more accurate types to programs, thus improving error detection at compile-time, and they allow one to assign types to a range of programs that do not type check in traditional type systems but produce a "reasonable" result.
2. There might be data dependent questions, e.g. a question posed may depend on the response given by the student earlier. This notion is also well captured by a theory of dependent types.
3. Evaluation, i.e. interpretation of the function: giving scores and feedback to the item for some particular data collected.

Let **`S`** = type of scores, for example, **`[0..10]`** (in Spain), **`[1..6]`** (in Germany), **`[A..F]`** (in USA), **`B`** (the type of Booleans, passed/not passed) or the number of points given to a correctly responded item. The scores of the items in a section or assessment are combined to give an overall score. Analogously, if we are using this notation to specify a survey we could compute some statistics.

Let **`F`** = type of feedbacks, which would typically be **`String`**, the type of strings. The result type of a question could be **`(SxF)`**, a score and some feedback. Let's call **`R = SxF`**

In an assessment without feedback, we have **`F = U`** (unit type).

In the following sections, we study first choice-based items (section 2), then text-based items (section 3), and then the composition of items (section 4). We finish by discussing the rendering options (section 5) and giving some conclusions (section 6).

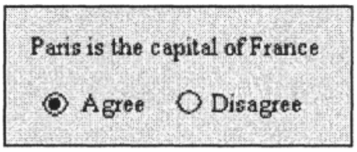
2. CHOICE-BASED ITEMS

2.1 True/false items

A true/false item is a function of type **`B->R`**. We will call it a 1-from-2 item (2 possible answers, of which 1 is correct).

```
parisFrance :: B -> R
parisFrance true  = (1, "right, congratulations!")
parisFrance false = (0, "wrong, travel more!")
```

A typical rendering could make use of radio buttons in an html form.



But we want to separate the value responded from the position the user has to click. We introduce an indirection step by defining a bijective function **`order`** that maps ordinals to Booleans (in QTI this is handled with the **`ident`** attribute of the **`response_label`** tag):

```
order :: [0..1] -> B
order 0 = true
order 1 = false
```

then, the composition of both functions is of type:

```
parisFrance o order :: [0..1]->R
```

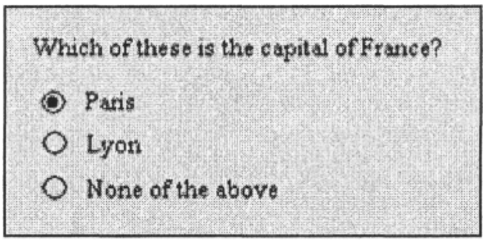
Using a different ordering function produces a reshuffling of the same item answers. There are many more indirection steps in IMS QTI, which we ignore. This item is a special case of the multiple choice item (next).

2.2 Multiple choice items

A multiple choice item has type **Enum(α) => α->R**, where α is a type variable. α should be an enumeration type. We will call this a 1-from-n item (n possible answers, of which 1 is correct).

```
parisLyonNoneFrance :: (Paris, Lyon, None) -> R
parisLyonNoneFrance Paris = (1, "well done!")
parisLyonNoneFrance Lyon  = (0, "wrong, try again!")
parisLyonNoneFrance None  = (0, "wrong, try again!")
```

Again applying an ordering function, in this particular example of type `[0..2]->String`, would give us an item with ordered responses. A typical rendering could make use of radio buttons in an html form:



Which of these is the capital of France?

(●) Paris
( ) Lyon
( ) None of the above

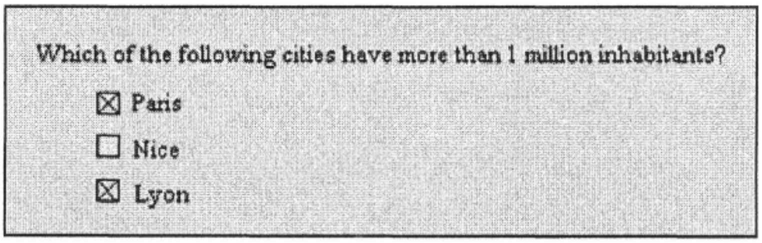
This item is a special case of the multiple response item (next).

2.3 Multiple response items

We call a multiple response item an m-from-n item (n possible answers, of which m are correct). Its type should be **℘(α)->R** (where ℘ is the powerset constructor), because the user has to get the set of correct answers.

```
inh1MFrance :: ℘(Paris, Nice, Lyon) -> R
inh1MFrance {Paris,Lyon}= (10, "very good!")
inh1MFrance {Paris}     = (5, "got one!")
inh1MFrance {Lyon}      = (5, "got one!")
inh1MFrance else        = (0, "not quite!")
```

A typical rendering could make use of checkboxes in an html form. An ordering function would map response positions to response values.



Which of the following cities have more than 1 million inhabitants?

☒ Paris
☐ Nice
☒ Lyon

2.4 Summary

The choice-based items are related as follows:

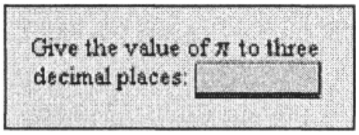
{1-from-2} ⊆ {1-from-n} ⊆ {m-from-n}

After applying ordering functions, the true/false items are a subset of the multiple-choice items, and these a subset of the multiple-response items.

3. TEXT-BASED ITEMS

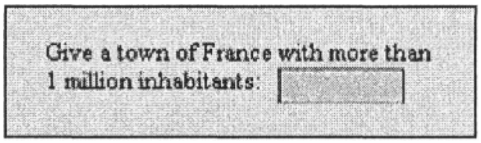
3.1 Fill-in-the-blank items

A fill-in-the-blank item is of type **α->R.** A typical rendering could make use of input elements of type text in an html form.


Give the value of π to three decimal places:

```
pi3Decimals :: Num -> R
pi3Decimals 3,141 = (1, "very good!")
pi3Decimals else  = (0, "not quite!")
```

In general there might be several correct answers (if there is logically just one, one might allow different spellings or upper/lower case alternatives).


Give a town of France with more than 1 million inhabitants:

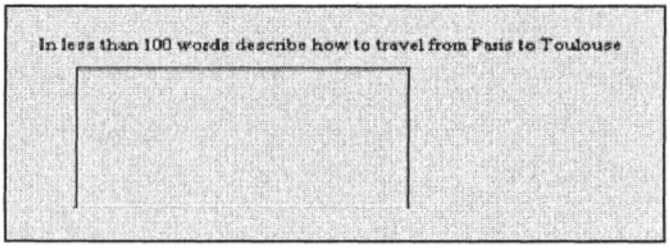
```
inhab1MFrance :: String -> R
inhab1MFrance "Paris "    = (1, "very good!")
inhab1MFrance "Lyon"      = (1, "very good!")
inhab1MFrance "Marseille" = (1, "very good!")
inhab1MFrance "Toulouse"  = (1, "very good!")
inhab1MFrance else        = (0, "not quite!")
```

3.2 Short answers

A short answer is a kind of item whose evaluation cannot be fully automated in general. Whereas in a fill-in-the-blank item, one can list all the possible valid answers, this is normally not the case for short answers or full essays in general. Thus, this type of items needs intervention by a grader. We model this with an additional argument that reflects the (sometimes not formalizable) criteria of the grader. Let's call its type **c**:

```
short-answer :: (C -> String -> R)
```

A rendering could make use of a text-area element in an html form.


In less than 100 words describe how to travel from Paris to Toulouse

Some item generation tools have another kind called "essay". We subsume it under the same kind as "short answer".

3.3 File upload

When the text to be delivered is even lengthier, it is more reasonable to have the student create it using some editor and them save it in a file. The grader criteria again are needed in order to evaluate the results. This type (which is not included in the IMS QTI specification) is similar to the one of the short answer, except, that the string is interpreted as a file name. There is an indirection step to get the submitted text:

```
file-upload :: (C -> String -> R)
```

A rendering could make use of input elements of type text in an html form in which one could write the name of the file to be uploaded. It would be useful to have a window in which to browse through the local directory.

3.4 Summary

The text-based items are related as follows:

$$\{\text{fill-in-the-blank}\} \subseteq \{\text{short-answer}\} \approx \{\text{file-upload}\}$$

Fill-in-the-blank is a subset of short-answer, where the length is limited. Short-answer and file-upload are equivalent. The difference is whether the text has to be typed in online or stored previously in a file. Choice-based items are a special case of text-based items. Instead of typing in the answers, they are selected from a list of possible values. We move from a finite set of possible answers to a (theoretically) infinite set for the text-based items.

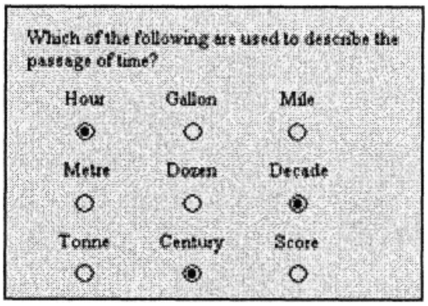
4. COMPOSITION OF ITEMS

4.1 Simple sequential composition of items

We can compose items sequentially, by just putting one after the other in the same way as we compose two sentences in an imperative programming language (by using the ";" symbol): one sentence is executed after the other. There the result is a composite sentence and here, a composite item. Its type is the product of the types of the individual items, but some type operations can be performed. A item composed sequentially by two ordered true/false items with the same result type would be of type **`([0..1]->R) x ([0..1]->R)`**, which is equivalent for our purposes to **$([0..1]^2\text{->}R^2)$**.

There should be a function the converts the result of type **R^2** of the individual items to an overall result of type **R**. This kind of sequentially composed item is a special case of the one presented in the next section, in which the second item depends on the answer of the first item.

IMS QTI considers what they call a Matrix-based Multiple Response a composite item type. This is an example:



Which of the following are used to describe the passage of time?

Hour	Gallon	Mile
●	○	○
Metre	Dozen	Decade
○	○	●
Tonne	Century	Score
○	●	○

We consider this a sequential composition of items, in this case, multiple choice items that share the questions, and therefore can be skipped. We use the name Matrix-based for the two-dimensional composition (see below).

4.2 Dependent sequential composition of items

In this type of sequential composition, the second item may be one or another depending on the user's behaviour. Here is a simple example with an initial true/false item and a second one that depends on the value given to the first one by the user. We use a functional notation to capture the result of the item and an ad-hoc notation for dependent types, to make the point clearer:

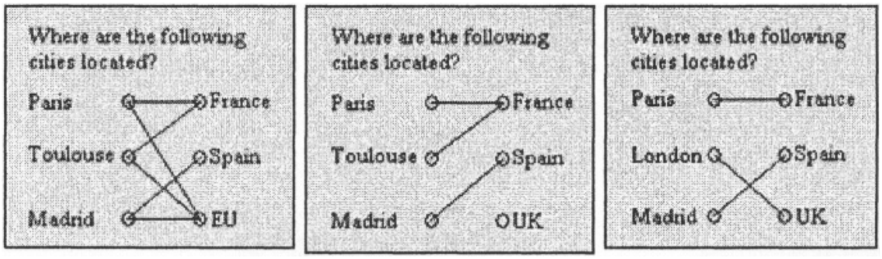
```
depSeqQuest (x,y) =
let (s,f) = (lyonFrance(x)) in
  let (s',f') in if s then (londonUK (y))
                  else (parisLyonNoneFrance (y))) in
    (scorecomb(s,s'), feedbcomb(f,f'))
```

`scorecomb` and `feedbcomb` combine the scores and feedback of the individual items to the result of the composed item. Its type happens to be dependent. Here are the types of all the items:

```
lyonFrance :: B->R
londonUK :: B->R
parisLyonNoneFrance :: (Paris, Lyon, None) -> R
depSeqQuest ::
    (B∃b x (if b then B else (Paris, Lyon, None)))->R
```

4.3 Matching item

This type of item is not included in IMS QTI, but is available in several assessment systems. Therefore, it is of interest to include it in this taxonomy. In a matching item we have two sets of elements (A and B) that can be of different cardinality (|A| and |B|). The student has to link each element in A with one or several of the elements in B. There are several variants now:


Where are the following cities located?
Paris
Toulouse
Madrid
France
Spain
EU
Where are the following cities located?
Paris
Toulouse
Madrid
France
Spain
UK
Where are the following cities located?
Paris
London
Madrid
France
Spain
UK

1. M_1: each element in A can be linked to one or more in B, without restriction. This is like having |A| multiple response items that share the response set (B). Type: $\wp(\wp\ (\alpha)\text{->}\mathbf{R})$.
2. M_2: each element in A can be linked to one and only one in B, where those in B can be reused (the inverse is a partial function). This is like having |A| multiple choice items that share the response set (B). Type: $\wp\,(\alpha\text{->}\mathbf{R})$
3. M_3: each element in A can be linked to one and only one in B, without the possibility of reusing those in B (we have a bijection, if

|A|=|B|). This can be understood as an extension of the true/false item to cardinality |A|. Type: $\wp\,(\alpha\text{->}\mathbf{R})$

The matching items are related to the choice-based items as follows:

$$\begin{array}{ccccc} \{\text{1-from-2}\} & \subseteq & \{\text{1-from-n}\} & \subseteq & \{\text{m-from-n}\} \\ & & \in & & \in \\ M_3 & \subseteq & M_2 & \subseteq & M_1 \end{array}$$

4.4 Two-dimensional composition of items

A two-dimensional item is a tabular or matrix-based arrangement of items. An example is a crossword puzzle. This is also not a type defined in IMS QTI, but is a nice example of composite item. A crossword puzzle is a two-dimensional composition of fill-in-the-blank items. Type: $\wp\,(\alpha\text{->}\mathbf{R})$

	A.	B.	C.
1.	Q	■	A
2.	T	I	M
3.	I	M	S

Horizontal:
1. A. Question
   C. Answer
2. A. Berners-Lee
3. A. Instructional Metadata Specification

Vertical:
A. 1. Question and Test Interoperability
B. 2. Instant Messaging
C. 1. American Mathematical Society

5. RENDERING

We separate the type of item from the way it is presented or rendered. A true/false item can be rendered with radio buttons or with an image map, conceptually it is the same. Not all renderings are possible for all types of items. Here is a table that captures the reasonable possibilities:

RENDERING	Radio	Checkbox	Text	TextArea	Slider	Image
True/False	X		X		X	X
Multiple Choice	X		X		X	X
Multiple Response		X				X
Fill-in-the-Blank			X			
Short Answer				X		
File Upload			X			

6. CONCLUSION

We have used a powerful type system when giving types to items for convenience. This exercise just shows the complexity that is inherent in this topic. When implementing a system that handles these item types, it is clear that a polymorphic, dependent type system is not needed; instead more variants of functions have to be implemented.

We have separated elements which are conceptually orthogonal on the one hand, and identified those which are related on the other one. The rendering issue has to be clearly separated from the basic types of items. We have distinguished basic items from those that can be obtained through composition. For instance, matrix-based multiple response items or matching items are really composite ones. A section is just a composite item with some selection and ordering functions. We think that with the definition of this taxonomy, the different types of items can be better understood and put into context. While defining the taxonomy we have identified some additional types of items, which are not part of IMS QTI, but are of interest.

ACKNOWLEDGEMENTS

This work has been partially sponsored by the E-LANE project [3] (EU @LIS Programme) and the SIEMPRE project (Spanish Ministry of Science and Technology, TIC2002-03635). In the E-LANE project, work is being carried out to extend the .LRN Learning Management System [8] (based on the OpenACS platform [9]) to cope with IMS QTI based assessments.

REFERENCES

[1] IMS QTI, **http://www.imsglobal.org/question**
[2] OKI, Open Knowledge Initiative, **http://web.mit.edu/oki**
[3] E-LANE project, **http://www.e-lane.org**
[4] Canvas Learning, **http://www.canvaslearning.com**
[5] QuestionMark, **http://www.questionmark.com**
[6] Respondus, **http://www.respondus.com**
[7] IMS Assesst Designer, **http://www.xdlsoft.com/ad**
[8] .LRN, **http://www.dotlrn.org**
[9] OpenACS, **http://www.openacs.org**