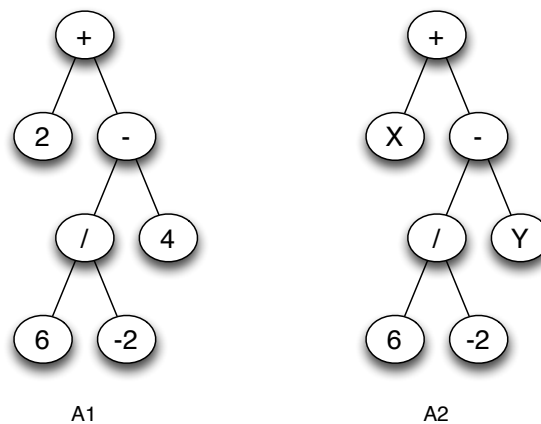


LABD

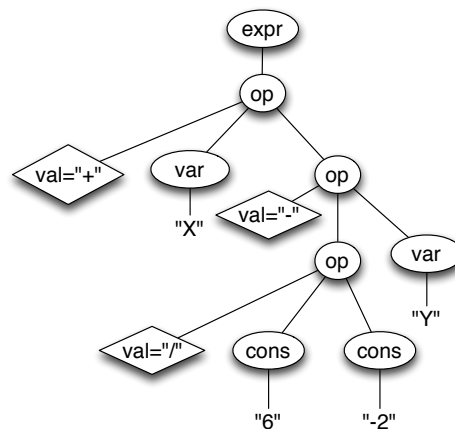
mars 2015

Programmer avec XQuery (2)

Pour ce TP, on reprend la représentation en XML d’expressions arithmétiques du TP 3 en s’appuyant sur leur forme arborescente comme rappelée ci-dessous : Il s’agit d’expressions arithmétiques entières dans lesquelles peuvent apparaître des variables. Par exemple, les expressions arithmétiques $E1 = (2 + ((6 / -2) - 4))$ et $E2 = (X + ((6 / -2) - Y))$ ont comme représentation arborescente respectives les arbres A1 et A2 ci-dessous :



L’expression $E2$ précédente peut être représentée par l’arbre XML :



Ce qui donne au bout du compte le fichier `expression.xml` suivant disponible dans l’archive à télécharger sur le portail :

```
<?xml version="1.0" encoding="utf-8"?>
<expr xsi:schemaLocation="http://www.expression.org →
    ↪ expression.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://www.expression.org">
  <op val="+">
    <var>X</var>
    <op val="-">
      <op val="/"><cons>6</cons><cons>-2</cons></op>
      <var>Y</var>
    </op>
  </op>
</expr>
```

Dans le TP 3, il s'agissait de définir un schéma `expression.xsd` précisant la structure de ces documents XML. Un tel schéma est disponible dans l'archive, ce schéma définit l'espace de nom `http://www.expression.org`. Dans la suite, on dira qu'un fichier est de type `expression` si celui-ci décrit une expression arithmétique, c'est à dire si le fichier est validé par le schéma `expression.xsd`.

L'objectif du TP est de définir un module de bibliothèque de nom `expression.xq` permettant de manipuler des fichiers de type `expression`. Les différentes questions du TP permettront d'enrichir progressivement ce module. À chacune de ces étapes, vous testerez la nouvelle fonctionnalité en créant une requête exécutable XQuery important le module. Il est fortement conseillé d'utiliser la bibliothèque **Saxon** en ligne de commande pour évaluer vos requêtes afin de profiter de messages d'erreurs plus explicites que dans le logiciel `editix`.

Pour tester, vous trouverez dans l'archive un `jar` exécutable de nom `gen.jar` qui permet de générer aléatoirement des fichiers de type `expression`. Ce `jar` exécutable dispose d'un certain nombre d'options permettant de définir la taille de l'expression, les variables pouvant apparaître dans l'expression, Dans un terminal, tapez la commande `java -jar gen.jar -h` pour afficher une documentation sommaire des différentes options possibles.

Question 1 : Créez le module `expression.xq` et définissez dans ce module une fonction XQuery de signature `print($name as xs:string) as xs:string` qui transforme tout fichier de type supposé¹ `expression`, dont le nom est passé en paramètre à la fonction, en un texte correspondant à l'écriture infixe complètement parenthésée de l'expression arithmétique correspondante. Par exemple, cette transformation appliquée au fichier XML correspondant à l'expression *E2* de l'exemple doit sortir le texte `(X + ((6 / -2) - Y)) ^ 2`. Testez sur plusieurs instances à l'aide de

1. On ne dispose pas en salle de TP d'une version de logiciel ou de bibliothèque XQuery permettant de vérifier que le fichier passé en paramètre est bien de type `expression`

2. Pour ne pas avoir l'entête `<?xml version="1.0" encoding="UTF-8"?>`, ajoutez la ligne `declare option saxon:output "omit-xml-declaration=yes";` dans le prologue de votre requête exécutable XQuery.

gen.jar.

Question 2 : Ajoutez dans le module `expression.xq` une fonction

```
eval($name as xs:string) as xs:integer
```

qui évalue l'expression arithmétique décrite dans le fichier de type `expression` dont le nom est passé en paramètre à la fonction. L'expression ne doit contenir aucune variable et votre fonction doit déclencher une erreur dans le cas contraire³. Par exemple, cette fonction appliquée au fichier correspondant à l'expression *E1* de l'exemple doit sortir le texte -5. Testez.

On associe des valeurs entières aux variables dans des fichiers XML dont la structure correspond à l'exemple `variables.xml` ci-dessous :

```
<?xml version="1.0" encoding="UTF-8"?>
<variables>
  <X>8</X>
  <Y>3</Y>
</variables>
```

Question 3 : Ajoutez dans le module `expression.xq` une fonction

```
eval-var($name as xs:string, $variables as xs:string) as xs:integer
```

qui évalue l'expression arithmétique décrite dans le fichier de type `expression`, dont le nom est donné par le paramètre `$name`, en affectant aux variables apparaissant dans l'expression les valeurs qui leur sont associées dans le fichier dont le nom est donné par le paramètre `$variables`. Votre programme doit générer une erreur si une variable apparaît dans l'expression mais est absente ou apparaît plusieurs fois dans le fichier des variables `$variables`. Par exemple, cette fonction appliquée au fichier XML correspondant à l'expression *E2* de l'exemple avec le contenu du fichier `variables.xml` tel que ci-dessus doit sortir le texte 2

Question 4 : Ajoutez dans le module `expression.xq` une fonction

```
simplifie($name as xs:string , $variables as xs:string) as element()
```

qui évalue *le plus possible* l'expression arithmétique décrite dans le fichier de type `expression` de nom `$name` en affectant aux variables apparaissant dans l'expression les valeurs qui leur sont associées dans le fichier dont le nom est donné par le paramètre `$variables`, en calculant sous forme de constante les sous-arbres expressions composés uniquement de constantes et de variables *évaluables* et en ne laissant que les variables qui ne sont pas définies dans le fichier `$variables`. Votre fonction doit générer une erreur si une variable de l'expression en entrée apparaît plusieurs fois dans le fichier `$variables`. Par exemple, cette fonction appliquée au fichier correspondant à l'expression *E2* de l'exemple avec le contenu du fichier `variables.xml` suivant :

3. Vous pouvez utiliser la fonction `fn:error` qui permet d'interrompre une requête XQuery. Voir <http://www.w3.org/TR/xpath-functions/#func-error>

```
<?xml version="1.0" encoding="UTF-8"?>
<variables>
  <Y>9</Y>
</variables>
```

doit sortir le document XML :

```
<?xml version="1.0" encoding="utf-8"?>
<expr xsi:schemaLocation="http://www.expression.org →
  ↪ expression.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.expression.org">
  <op val="+">
    <var>X</var>
    <cons>-12</cons>
  </op>
</expr>
```

Remarque : On ne demande pas que votre programme transforme un fichier de contenu

```
<?xml version="1.0" encoding="utf-8"?>
<expr xsi:schemaLocation="http://www.expression.org →
  ↪ expression.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.expression.org">
  <op val="+">
    <var>X</var>
    <cons>0</cons>
  </op>
</expr>
```

en :

```
<?xml version="1.0" encoding="utf-8"?>
<expr xsi:schemaLocation="http://www.expression.org →
  ↪ expression.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.expression.org">
  <var>X</var>
</expr>
```