Strategic Querying of XML Documents

Objectivo

- Definir a linguagem HaQuery, desenvolvida do DI/UM sobre a biblioteca Strafunski

Ponto da Situação

HaXml

HaQuery

Combinadores XPath sobre o HaQuery

Strafunski como biblioteca para *generic programming*??

- Combinadores XPath sobre Strafunski

HaXml

- Parsing de ficheiros XML para tipos de dados haskell
- Validação de XML
- Pretty-printing para XML

- Parsing de HTML com correção de erros

HaXml (cont)

Representação de ficheiros XML em Haskell:

```
data Document = Document Prolog (SymTab EntityDef) Element
...
data Element = Element Name [Attribute] [Content]
data Content = CElem Element
             | CElem String
```

HaXml (cont)

Aplicação de filtros aos documentos:

type CFilter = Content -> [Content]

Recebe como argumento o documento completo e recebe como resultado a lista dos elementos/texto que satisfazem o filtro.

Combinadores de filtros:

position :: Int -> CFilter -> CFilter	seleciona um elemento numa determinada posição.
children :: CFilter	devolve o descendente directo de um elemento
o :: CFilter -> CFilter -> CFilter	composição de dois filtros
txt :: CFilter	devolve o conteudo textual de um elemento

HaXml (cont)

Exemplo de Utilização:

```
processXMLwith filter =
   do
        xmlInput <- readFile “teste.xml”
        let xml = xmlParse “teste.xml” xmlInput
        let cont= getElem xml
        let res=filter cont
        putStr $ render.vcat.map content $ res
```

HaQuery

- Querys XPath em Haskell
- Faz parsing de uma query em texto e gera a query em Haskell
- Tipo de dados para a query
- Combinadores de Parsing
- Exemplo:

  “Q: /curso/cadeira/inscritos ?”

Combinadores HaQuery

- Combinadores para expressar querys XPath em HaQuery
- Elimina o processo de parsing
- Exemplo:

  (-/) $ tag "curso" / tag "cadeira" / tag "inscritos"

Combinadores HaQuery (cont)

-/	selecciona a partir do contexto raiz
-//	selecciona em profundidade desde a raiz
./	selecciona a partir do contexto actual
tag *nome*	selecciona os elementos com o nome /nome/
prefixtag *nome*	selecciona os elementos com o nome /nome*/
sufixtag *nome*	selecciona os elementos com o nome /*nome/
*****	selecciona um elemento, qualquer que seja o seu nome
-	selecciona o conteúdo textual de um ficheiro
predicatetag *str lst*	selecciona os elementos de nome *str* e que satisfazem o predicado *lst*
positiontag *str pos*	devolve os elementos de nome *str* e que se encontram na posição *pos*

Tem ainda definidos os seguintes operadores:

==, !=, <=, <, >, >= (definidos para tipos textuais)

.==., .!=., .<=., .<., .>., .>=. (definidos para tipos numéricos)

Strafunski

- O que é?
- Para que serve?
- Como se usa?

Any Questions?