



# Libreria uu-parsinglib

Carlos Gomez

Universidad Mayor de San Simón
Facultad de Ciencias y Tecnología
Carrera de Licenciatura en Informática
Comunidad Haskell San Simon

17 de Septiembre, 2011

# Contenido



# Introducción

# Libreria uu-parsinglib


Es una herramienta **EDSL**[a] para Haskell que permite procesar una entrada a través de una descripción similar a la Gramática Contreta del lenguaje que se quiere procesar.

[a] EDSL: Embeded Domain Specific Language

# Beneficios


- Usar el mismo mecanismo de abstracción, tipado y nombrado de haskell.
- Crear —parsers— al vuelo o en tiempo de ejecución del programa.
- No depender de otros programas separados para generar el parser. Hacer todo en haskell.
- Usar el mismo formalismo para describir scanners y parsers.

# Beneficios (Continuacion)


- Usar el mismo formalismo para describir funciones semánticas y parsers.
- Trabajar con versiones limitadas de gramáticas infinitas.
- Error-correcting parser combinators.
- Disponer de combinadores para trabajar con gramaticas con permutaciones.
- Disponer de un interface monadica para conocer informacion intermedia de parsing.



# Combinadores

# Combinadores Basicos

## Sus definiciones casi no dependen de otros combinadores


- pSym
- pReturn
- pFail
- Opcional << | >
- pAny

# Combinadores Derivados

## Simples


Ejemplo de algunos Combinadores Derivados Simples:

- $<\$>$
- $<\$$
- $<*$
- $*>$

# Combinadore Derivados

## Secuenciales


Ejemplo de algunos Combinadores Derivados Secuenciales:

- pList, pList_ng, pListSep, pListSep_ng
- pList1, pList1_ng, pList1Sep, pList1Sep_ng
- pSome
- pMany
- pFoldr, pFoldr_ng, pFoldrSep
- pFoldr1, pFoldr1_ng, pFoldr1Sep

# Combinadores Derivados

## Especiales


Ejemplo de algunos Combinadores Derivados Especiales:

- pMaybe
- pEither
- pPacked
- pChainr, pChainl
- pCount

# Combinadores Derivados

## Repetidores


Ejemplo de algunos Combinadores Derivados Repetidores:

- pExact
- pBetween
- pAtLeast
- pAtMost

# Combinadores Derivados

## Permutadores


Ejemplo de algunos Combinadores Derivados Permutadores:

- $<||>$
- $<<||>$
- pmMany



# Utilitarios para construir un parser

# Utilitarios

## Parser para Caracteres Especiales


Ejemplo de algunos Combinadores especificos para caracteres:

- pLetter, pDigit, pLower, ... , pAscii
- pSpaces

# Lexeme Parsers

## Parsers con una ideologia de lexeme


Ejemplo de algunos Combinadores definidos con la ideologia lexeme:

- pDot
- pComma
- pLBrace
- pSymbol
- pInteger
- pParens
- pTuple

# Funciones Principales de uu-parsinglib

## Funciones principales para ejecutar el parser


Las funciones principales para ejecutar el parser:

- execParser
- runParser



# Ejemplo de Aplicacion

# Ejemplo de Aplicacion


Escribir un parser para reconocer un lenguaje de marcado de tipo XML.

# Ejemplo de Aplicacion

## Entrada


Ejemplo de una entrada para la aplicacion:

```
<html>
    <big bold1="ok1" bold2="ok2" >
        image
    </big>
</html>
```

# Ejemplo de Aplicacion

## Salida

Ejemplo de una salida para la aplicacion:

```
---NTag html
   |
   +---NText "\n    "
   |
   +---NTag big
   |    | bold1 = "ok1"
   |    | bold2 = "ok2"
   |    |
   |    +---NText "\n        image\n    "
   |
   +---NText "\n"
```

# Tipo de dato: NTree

## Estructura Rosadelfa (NTree)


```
DATA NTree
   | NTree Node ntrees: NTrees

TYPE NTrees = [NTree]

DATA Node
   | NTag   tag           : String
            attributes    : {[(String, String)]}
   | NText  text          : String
```

# Definicion de Parser

## Parser para atributos


```
pAttributes :: Parser [(String, String)]
pAttributes
    = pListSep pSpaces pAttr
    where pAttr :: Parser (String, String)
          pAttr = (,) <$> pIdentifier
                      <* pSymbol "="
                      <*> pQuotedString
```

# Definicion de Parser

## Parser para reconocer un tag


```haskell
-- | Parser para reconocer un tag con inicio y fin
pTagged :: Parser NTree
pTagged
    = do (tag,attrs) <- pTagInit
         elems       <- pList_ng pXML
         pTagEnd tag
         return $ buildTag elems tag attrs
```

# Definicion de Parser

## pTagInit y pTagEnd


```haskell
-- | Parser para el tag de inicio
pTagInit :: Parser (String, [(String, String)])
pTagInit
    = (,) <$ pSymbol "<"
              <*> pIdentifier <*> pAttributes
          <* pSym '>'

-- | Parser para el tag final
pTagEnd :: String -> Parser String
pTagEnd tag
    = pSymbol "<" *> pSymbol "/" *>
        (lexeme (pToken tag))
    <* pSym '>'
```

# Definicion de Parser

## Parser para reconocer un tag especial


```haskell
-- | Parser para reconocer un tag especial (sin final)
pSpecialTag :: Parser NTree
pSpecialTag
    = buildTag []
        <$ pSymbol "<"
             <*> pIdentifier <*> pAttributes <*
           pSymbol "/" <* pSym '>'
```

# Definicion de Parser

## Parser para el texto


```haskell
-- Parser para reconocer cualquier texto dentro de un tag
pText :: Parser NTree
pText = buildText
        <$> pList1 (pSatisfy fcmp (Insertion "a" 'a' 5))
    where fcmp c = c /= '<'
```

# Definicion de Parser

## Funcion principal, pXML


```
-- | Parser que reconoce un tag normal/especial o texto
pXML :: Parser NTree
pXML = pText <|> pTagged <|> pSpecialTag
```

# Definicion de Parser

## Funcion principal, parser


```haskell
-- | funcion principal para el parser xml
parser :: FilePath -> String -> NTree
parser fn inp = runParser fn p inp
    where p :: Parser NTree
          p = pSpaces *> pXML <* pSpaces
```



# Fin de la Presentación

Carlos Gomez
carliros.g@gmail.com