Haskell en la Industria

El Caso de Erudite Science, Inc.

Pablo Ariel Duboue, PhD

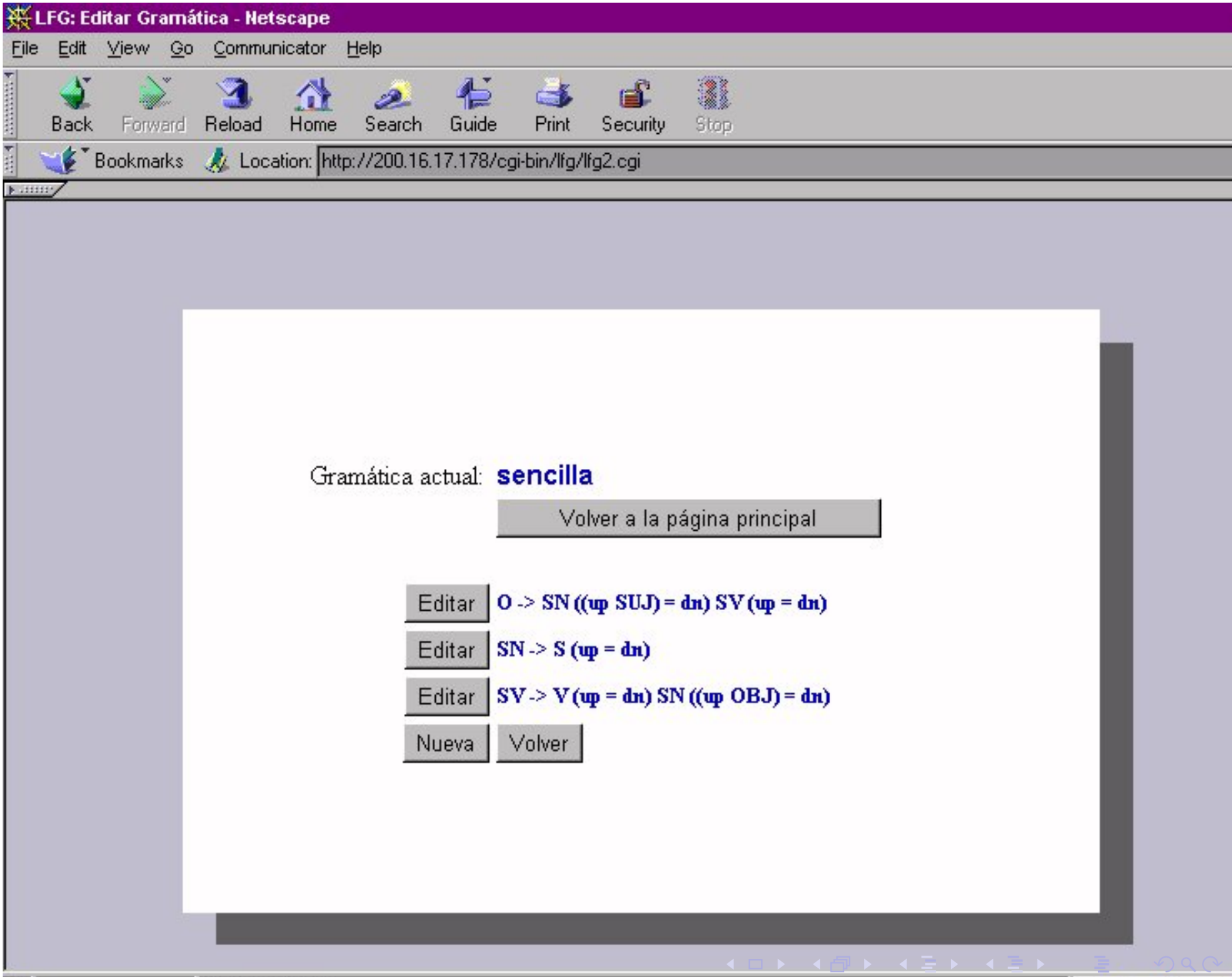
Universidad Nacional de Córdoba,
Facultad de Matemática, Astronomía y Física

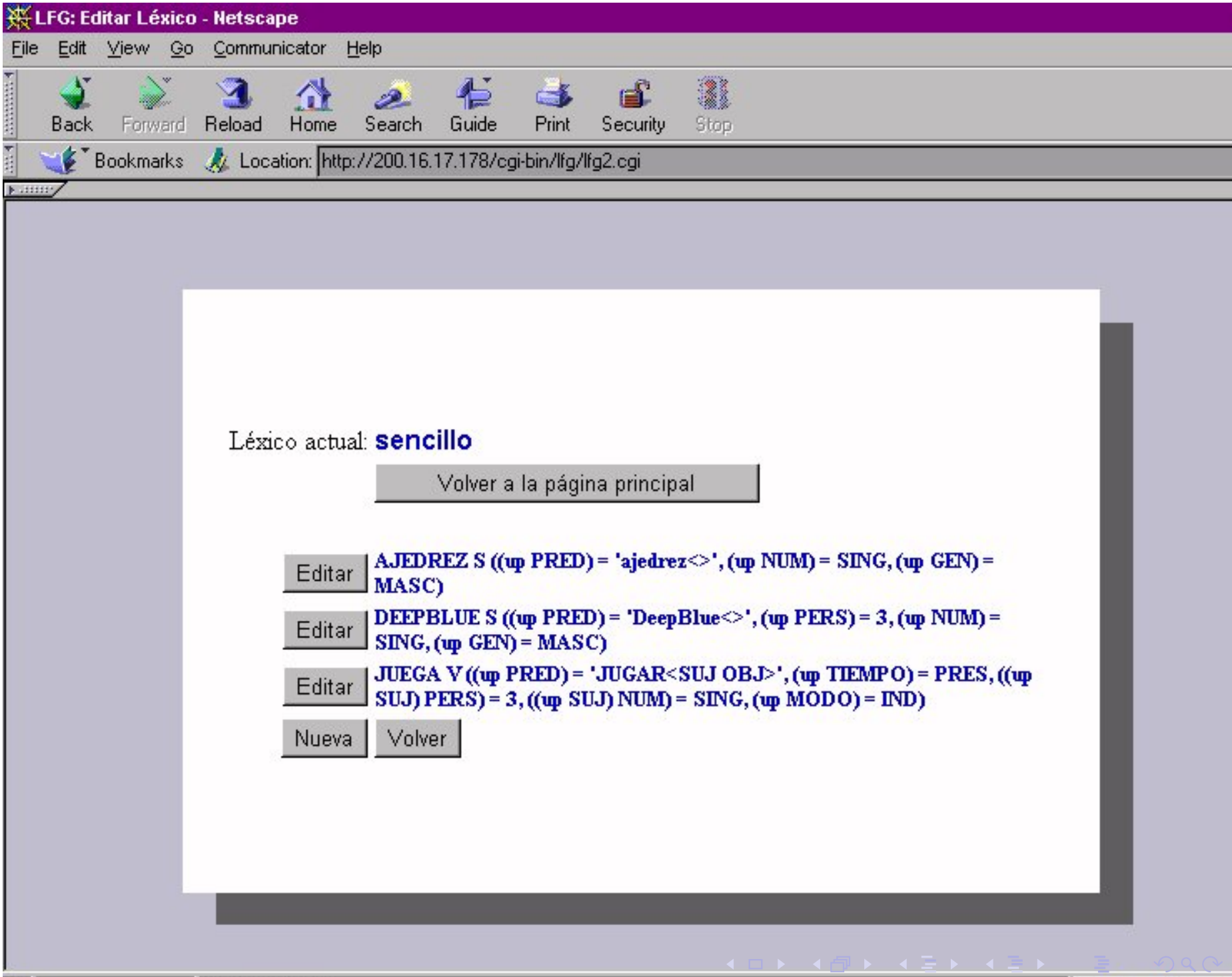
Pablo Duboue, Licenciado

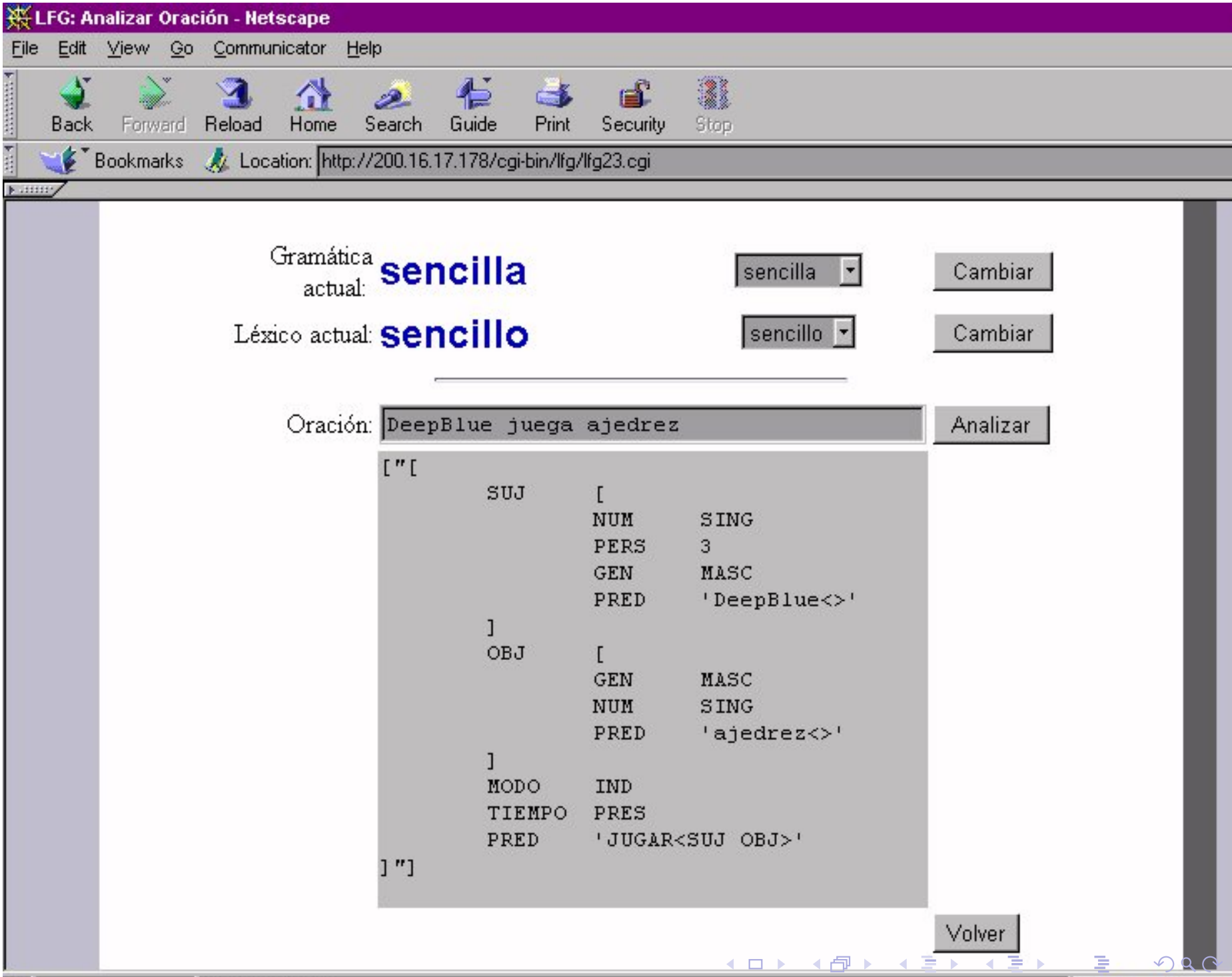
- Licenciado en Computación ingreso 1993
- Trabajo Final en PLN programada en Haskell
- Electiva: compilación de lenguajes funcionales

Trabajo Final

- Desarrollo de un Parser Funcional para el Lenguaje Castellano
- Un pequeño entorno para hacer análisis sintáctico usando una teoría particular (LFG)


LFG: Editar Gramática - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://200.16.17.178/cgi-bin/lfg/lfg2.cgi
Gramática actual: sencilla
Volver a la página principal
Editar O -> SN ((up SUJ) = dn) SV (up = dn)
Editar SN -> S (up = dn)
Editar SV -> V (up = dn) SN ((up OBJ) = dn)
Nueva Volver


LFG: Editar Léxico - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://200.16.17.178/cgi-bin/lfg/lfg2.cgi
Léxico actual: sencillo
Volver a la página principal
Editar AJEDREZ S ((up PRED) = 'ajedrez<>', (up NUM) = SING, (up GEN) = MASC)
Editar DEEPBLUE S ((up PRED) = 'DeepBlue<>', (up PERS) = 3, (up NUM) = SING, (up GEN) = MASC)
Editar JUEGA V ((up PRED) = 'JUGAR<SUJ OBJ>', (up TIEMPO) = PRES, ((up SUJ) PERS) = 3, ((up SUJ) NUM) = SING, (up MODO) = IND)
Nueva Volver


LFG: Analizar Oración - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://200.16.17.178/cgi-bin/lfg/lfg23.cgi
Gramática actual: sencilla
sencilla
Cambiar
Léxico actual: sencillo
sencillo
Cambiar
Oración: DeepBlue juega ajedrez
Analizar
["[
SUJ [
NUM SING
PERS 3
GEN MASC
PRED 'DeepBlue<>'
]
OBJ [
GEN MASC
NUM SING
PRED 'ajedrez<>'
]
MODO IND
TIEMPO PRES
PRED 'JUGAR<SUJ OBJ>'
]"]
Volver

Compilación de Lenguajes Funcionales Perezosos

- Optativa
- Implementing functional languages: a tutorial
  - Simon Peyton Jones and David Lester. Published by Prentice Hall, 1992

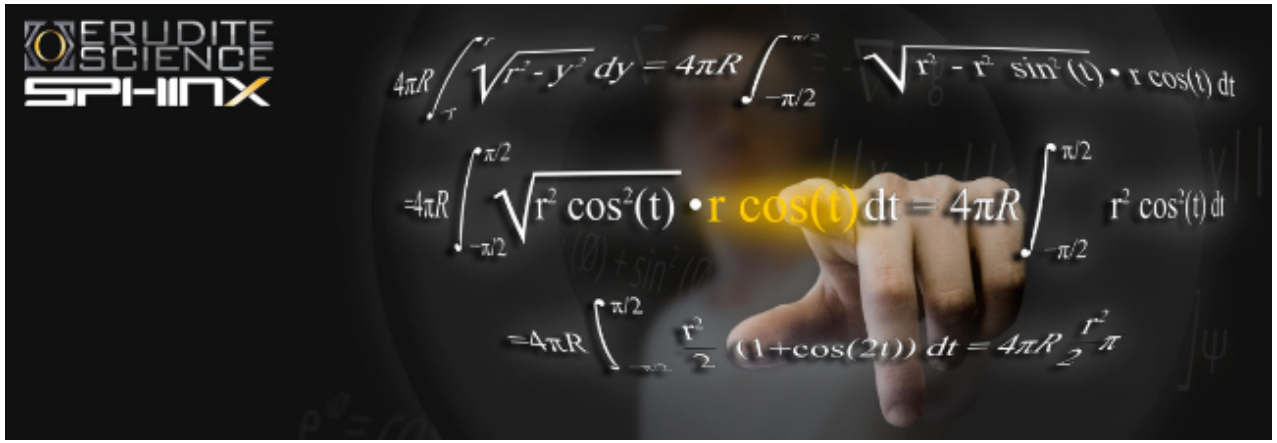
Haskell

- (Wikipedia) Haskell es un lenguaje de programación estandarizado multi-propósito puramente funcional con semánticas no estrictas y fuerte tipificación estática.
- Proceso de estandarización en los '80
- Lenguaje creado en los '90 (un poco después de Perl, un poco antes de Python)
- Funciones como elementos principales
- Tipos de datos, funciones recursivas y definición por patrones

Otras Aplicaciones Industriales

- De la wiki de Haskell: https://wiki.haskell.org/Haskell_in_industry
  - Empresas grandes: Microsoft, Intel, Google
  - Empresas contratistas militares: Galois Inc, BAE
  - Empresas de operación bursátil digital
- En dice.com:
  - Nada, 24 posts que mencionan Haskell como "un plus"
- En google (http://functionaljobs.com/):
  - Más interés en otros lenguajes funcionales (Scala, OCamL, Clojure, JavaScript)

Erudite Science, Inc.



- Fundada en 2013
- Mejorar la educación matemática usando tecnología

Hacer de la tutoría personalizada un hecho para todos los alumnos, cuando y donde la necesiten, salvando las distancias entre alumnos, educadores y las aulas.

Nuestro Producto

- Sphinx: paso a paso tutor para resolución de expresiones formulaicas
- [Demo]

Grupo de Computación y Lógica de McGill

- Francisco Ferreira (UBP'98)

- Steven Thepshourinthone (McGill'15)

Python

- Integración: ejecutable binario independiente
- Sistema python basado en el micro-framework Flask
- Haskell puede generar librerías dinámicas que se pueden cargar desde Python pero el binario es más robusto

Python (2)

Flask

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello World!'

if __name__ == '__main__':
    app.run()
```

Comunicación: JSON

JavaScript Object Notation

```
{ "x" : 5.0, "y" : 1.3,
  "dimensions" : [ "x", "y", "z" ],
  "system" : { "name" : "cartesian" }
}
```

Comunicación: JSON (2)

Utilizando la librería AESON

```
{-# LANGUAGE DeriveGeneric #-}
import qualified Data.Aeson as J

data Coord = Coord { x :: Double, y :: Double }
    deriving (Show, Generic)
instance J.FromJSON Coord

increase :: String -> Coord
increase jstr = case (J.decode jstr) of
  Just coord -> Coord { (x coord) + 1, y }
  _ -> error "Parse error"
```

Literate Programming (+)

- Programar de manera profesional no es solo el código
  - es poder mantener un equipo de gente trabajando y contribuyendo de manera eficiente.
  - es una actividad en equipo y muy social
- Comunicación es fundamental
  - énfasis en trabajo grupal durante la carrera

Mónadas, cuando y porqué (+)

una *mónada* será una tres-upla (`M`, `unitM`, `bindM`) consistente en un constructor de tipos `M` y dos funciones polimórficas.

```
unitM  ::  a -> M a,
bindM  ::  M a -> (a -> M b) -> M b,
```

Estas funciones deben satisfacer tres leyes:

Identidad a izq: `(unitM a) 'bindM' k = k a`

Identidad a dcha: `m 'bindM' unitM = m`

Asociatividad:

```
  m'bindM'(λ a→(k a)'bindM'(λ b→h b))
= (m'bindM'(λ a→k a)'bindM'(λ b→h b)
```

QuickCheck (+)

- QuickCheck es una librería que permite la generación de casos de tests aleatorios y el chequeo de propiedades sobre ellos.
- En nuestro caso generamos fórmulas al azar y ejecutamos una serie de derivaciones sobre las mismas
  - Chequeamos que la fórmulas den el mismo resultado
- Está basado en un mónada especial Gen
  - Concepto de generadores y propiedades

Previsibilidad tiempo/espacio (-)

- La programación funcional perezosa hace difícil predecir el comportamiento en tiempo de ejecución en materia de tiempo y espacio (memoria)
- Algunos bugs muy difíciles de corregir
  - Utilización del módulo Debug.Trace y la función trace
- Chequeos de memoria en tiempo de ejecución
  - Opción del compilador -with-rtsopts="-M1024m" (o simil)
- Profiler ghc -prof -fprof-auto -rtsopts

Manejo de Dependencias

- Manejo de dependencias: instalar los módulos necesarios en las versiones correctas
  - Por ejemplo, AESON y QuickCheck son módulos definidos en el sistema Hackage
- Haskell tiene el sistema Cabal para instalar módulos dependientes
- No lo utilizamos todavía, usamos los paquetes definidos en Debian GNU/Linux

Otras Herramientas

- Usamos mucho ghci, un entorno de línea de comando para Haskell
- Permite la evaluación dinámica de expresiones
- Parecido a lo que era Hugs en su época y a la REPL de python actualmente

Compiladores estrictos y microemprendimientos (+)

- En un microemprendimiento informático el producto cambia rápidamente
  - Una empresa pequeña se puede adaptar a cambios y feedback rápidamente
- Cuando el producto cambia, el código obviamente cambia
- Estos cambios generan código re-adaptado con errores de interfaces
  - Los invariantes del código ya no son los mismos
- La alternativa a usar un lenguaje estrictamente tipado es definir gran cantidad de casos de test que chequeen que las distintas funciones son llamadas con los valores correctos

Fay

- Fay es un backend para GHC que compila un subconjunto de Haskell a JavaScript
- Se instala mediante Cabal
- Utiliza FFI (la funcionalidad de Haskell para ejecutar funciones fuera de Haskell) para ejecutar funciones JavaScript
- Una función pura en Haskell se convierte en una función JavaScript
    - Punto único de integración entre los dos lenguajes

Instalación

https://github.com/faylang/fay/wiki

```
cabal install cpphs
# asegurarse que cpphs esté en el PATH
cabal install fay fay-base
```

Ejemplo

https://github.com/DrDub/Albahaca/blob/master/AlbahacaHS.hs

- http://drdub.github.io/Albahaca/