TP 3 - ALP - 2020

Introducción

Hola! Este README es un documento complementario al PDF de la consigna del TP. Para este trabajo no van a necesitar nada nuevo, solamente se repiten las intrucciones básicas de `stack`. Si hicieron los TPs 1 y 2, lo único que les va a ser útil es la sección **Estructura del código**.

Stack

Para este TP vamos a usar **Stack**, una herramienta sencilla para desarrollar poryectos en Haskell. Stack tiene muchas utilidades, pero ahora nos vamos a concentrar sus funciones básicas.

Antes que nada, puede que tengas que instalarlo. En 1 hay guías de instalación para distintas plataformas.

Stack se encarga de instalar la versión correcta de GHC, instalar los paquetes necesarios y compilar el proyecto. Para las primeras dos, basta con abrir una terminal en el directorio `TP3` y ejecutar:

```
stack setup
```

Esto puede demorar un rato porque se encarga de descargar e instalar la verisón correcta de GHC. Este comando solo se debería tener que ejecutar una única vez. Al terminar esto, está todo listo para compilar el proyecto, que se hace con:

```
stack build
```

Este es el comando que van a tener que usar para compilar el proyecto cada vez que lo modifiquen.

Estructura del código

La estructura del proyecto es la siguiente:

```
.
├── app
│   └── Main.hs
├── src
│   ├── Common.hs
│   ├── Simplytyped.hs
│   ├── PrettyPrinter.hs
│   └── Parse.y
├── Ejemplos
│   └── Prelude.lam
├── test
├── README.md
├── Setup.hs
├── TP3.cabal
├── package.yaml
├── stack.yaml
└── stack.yaml.lock
```

IMPORTANTE: Solo deberían tener que modificar archivos de los directorios `src` y `Ejemplos`.

- En el directorio `app` se define el módulo `Main`, que implementa el ejecutable final.
- En el directorio `src` se encuentran los módulos sobre los que van a trabajar:
  - `Common` define los tipos de términos y valores en la consigna junto a algunos tipos auxiliares. **Los tipos definidos en este archivo ya cuentan con todas las extensiones planteadas en el TP, por lo que no deberían tener que modificarlo.**
  - `PrettyPrinter` tiene el Pretty Printer del lenguaje, **parcialmente implementado**.
  - `Parse.y` define el parser, que está **parcialmente implementado**. Para ello, este archivo especifica la gramática en BNF y provee el lexer. El módulo `Parse.hs` es generado por la herramienta `Happy` (explicada en la Sección 3 de la consigna) al hacer `stack build`, y se guarda en un directorio oculto.
  - `Simplytyped` tiene las funciones que hacen funcionar al intérprete y el inferidor de tipos, ambos **parcialmente implementados**.
- En el directorio `Ejemplos` está el preludio, con algunos términos del lambda cálculo simplemente tipado (STLC). En este directorio van a resolver el ejercicio 11.
- El resto de los archivos son de configuración del proyecto.

IMPORTANTE: Por favor, no cambiar los nombres de los módulos, tipos, constructores, funciones, etc. Ante cualquier duda consulte a su docente de cabecera.

¿Cómo ejecutarlo?

Una vez compilado el proyecto, se puede correr el ejecutable definido en `app/Main.hs` haciendo:

```
stack exec TP3-exe
```

Esto lanzará el evaluador interactivo de lambda cálculo simplemente tipado a implementar en este trabajo. Con el comando `:?` pueden leer sobre el resto de los comandos disponibles.