Facultad de Ciencias Exactas, Ingeniería y Agrimensura

Trabajo Práctico: DSL para la planificación de eventos

Análisis de Lenguajes de Programación

María Agustina Torrano

T-2989/1

2024

Índice

1. Introducción

Este trabajo consiste en un calendario implementado en Haskell, el cual sirve para gestionar eventos de manera sencilla. El calendario cuenta con las funciones básicas: agregar, modificar y borrar un evento. Además, es posible consultar los eventos programados para el día, semana o mes, entre muchas funcionalidades más que se irán explicando a lo largo del informe.

Personalmente, disfruto el uso de calendarios virtuales, sin embargo, estos tienden a enfocarse más en las interfaces gráficas. El propósito de este gestor de eventos es ofrecer una herramienta más minimalista que simplifique la escritura de eventos.

2. Manual de uso e instalación del software

Para poder utilizar el software, se debe tener instalado el compilador de Haskell, GHC. Este se puede descargar desde The Glasgow Haskell Compiler. Además, es necesario el sistema de paquetes Cabal, el cual facilita la instalación y el manejo de bibliotecas en Haskell. En el siguiente link hay más información sobre como instalarlo The Haskell Cabal.

El siguiente paso es asegurarse de que todas las bibliotecas utilizadas por el software estén correctamente instaladas con sus versiones correspondientes. Para esto, se debe ejecutar el siguiente comando:

```
$ cabal install
```

El calendario está diseñado para ser utilizado de forma interactiva, por lo tanto, para iniciar el programa basta con ejecutar:

```
$ cabal run
```

A partir de ahora ya es posible interactuar con el calendario.

3. Guía de usuario

3.1. Comandos

Los 9 comandos implementados para el calendario son los siguientes:

- `:load` o `:l`. Carga un programa desde un archivo el cuál se pasa como argumento.

- `:print` **o** `:p`**.** Imprime el calendario actual.
- `:reload` **o** `:r`**.** Vuelve a cargar el último archivo.
- `:quit` **o** `:q`**.** Sale del intérprete.
- `:help` **o** `:?`**.** Muestra la lista de comandos.
- `:ops` **o** `:o`**.** Muestra las operaciones posibles del calendario.
- `:close` **o** `:c`**.** Cierra y guarda el calendario actual.
- `:export` **o** `:e`**.** Exporta el calendario actual a un archivo .ics.
- `:import` **o** `:i`**.** Importa un archivo .ics el cual se pasa como argumento.

Es importante destacar que los archivos *.ics* son compatibles con Google Calendar, lo que permite que al exportar un archivo desde el calendario, pueda importarse en Google Calendar para visualizar los eventos de forma más gráfica. Además, este formato es compatible con Microsoft Outlook, Apple Calendar, Yahoo Calendar y otras aplicaciones.

3.2. Operaciones

Las operaciones que puede realizar el calendario pueden dividirse en operaciones básicas, de modificación o de consulta. Ahora veremos como funciona cada una.

3.2.1. Operaciones básicas

Dentro de las operaciones básicas tenemos aquellas que crean un calendario, crean y agregan un evento al calendario y eliminan un evento. Veamos cada una en detalle.

- **Creación de calendario.** Se espera la línea `C 'nombre'`. Luego, crea un nuevo calendario con el nombre que se pasa como argumento.
- **Creación y agregado de un evento.** El comando básico para realizar esta operación es el siguiente `E 'titulo' 'inicio' - 'finalizacion'`. Luego, el evento con el título creado se agrega al calendario. Sin embargo, existen muchas formas de crear un evento, algunas equivalentes y otras con mayores funcionalidades.

  1. `E 'titulo' 'dd/mm/aa' 'hh:min' - 'dd/mm/aa' 'hh:min'`
  2. `E 'titulo' 'dd/mm/aa' 'hh:min' - 'dd/mm/aa' 'hh'`
  3. `E 'titulo' 'dd/mm/aa' 'hh' - 'dd/mm/aa' 'hh:min'`

4. `E 'titulo' 'dd/mm/aa' 'hh' - 'dd/mm/aa' 'hh'`
5. `E 'titulo' 'dd/mm/aa' 'hh:min' - 'hh:min'`
6. `E 'titulo' 'dd/mm/aa' 'hh:min' - 'hh'`
7. `E 'titulo' 'dd/mm/aa' 'hh' - 'hh:min'`
8. `E 'titulo' 'dd/mm/aa' 'hh' - 'hh'`
9. `E 'titulo' 'dd/mm/aa' - 'dd/mm/aa'`
10. `E 'titulo' 'dd/mm/aa'`

Los primeros 8 ejemplos se pueden usar para escribir un mismo evento de distintas formas, es decir, `E Gimnasio 4/11/2024 15 - 16` es equivalente a `E Gimnasio 4/11/2024 15:00 - 4/11/2024 16:00`. Los últimos dos casos son considerados eventos que duran todo el día. Estos se verán de una forma particular al exportar el calendario como un archivo *.ics*, pero en este calendario se manejarán de la misma forma que el resto.

En todos los casos se puede o no agregar dos campos, uno para categorizar el evento y otro para agregarle recurrencia. Algunos ejemplos agregando estas funcionalidades son:

1. `E Gimnasio 4/11/2024 15 - 16 Salud`
2. `E Gimnasio 4/11/2024 15 - 16 every 2 days`
3. `E Gimnasio 4/11/2024 15 - 16 Salud every 2 days`

En los ejemplos, la categoría del evento es **Salud** y la recurrencia es **cada 2 días**. Observemos que se pueden poner los dos campos o sólo uno, pero siempre que se agreguen los dos, la categoría debe estar primera.

- **Eliminación de evento.** El comando que se debe ejecutar para eliminar un evento del calendario es `delete 'evento'`. Por ejemplo, `delete E Gimnasio 4/11/2024 15 - 16`.

  En este caso, el evento se debe escribir con el título, inicio y finalización (si es que tiene) de la misma forma que vimos en el item anterior. Esto es porque puede haber un mismo evento con el mismo nombre pero con fechas distintas o en una misma fecha puede haber distintos eventos.

3.2.2. Operaciones de modificación

Como vimos antes, al eliminar y agregar un evento, es necesario escribirlo de forma completa. Esto puede resultar bastante engorroso cuando lo único que se quiere modificar es la fecha de inicio por un error de tipeo o agregar una categoría a un evento ya existente. Es por esto que existe una operación de modificación la cual, a pesar que sea necesario escribir el evento completo para poder diferenciarlo del resto, se puede utilizar para cambiar distintas partes del mismo sin necesidad de escribirlo dos veces.

El comando que se debe ejecutar es `modify 'evento'`. Este generará un texto que esperará un número del 1 al 4 según las posibles modificaciones que se pueden realizar, de la siguiente forma:

```
Escriba la opción según la parte del evento 'título' que
quiera modificar:
1. Título del evento.
2. Horario/Día de incio.
3. Horario/Día de finalización.
4. Categoría.
```

Según el número que escriba el usuario, se realizarán las distintas modificaciones. Es importante aclarar que en todos los casos se debe seguir el formato con el que se viene trabajando.

3.2.3. Operaciones de consulta

Finalmente, tenemos operaciones que permiten consultar eventos con un mismo título, con una misma categoría y los que están programados para el día, la semana o el mes.

- **Búsqueda de un evento.** El comando a ejecutar es `search 'titulo'`. Se listarán, en formato de tabla, todos los eventos que tengan el mismo nombre.
- **Búsqueda de eventos con una misma categoría.** El comando a ejecutar es `category 'categoria'`. Se listarán, en formato de tabla, todos los eventos de una misma categoría.
- **Eventos del día.** El comando a ejecutar es `day`. Se mostrarán, en formato de línea de tiempo, todos los eventos del día.
- **Eventos de la semana.** El comando a ejecutar es `week`. Se mostrarán, en formato de calendario semanal, todos los eventos de la semana.

- **Eventos del mes.** El comando a ejecutar es `month`. Se mostrarán, en formato de almanaque, todos los eventos del mes.
- **Todos los eventos.** El comando a ejecutar es `all`. Se mostrarán, en formato de tabla, todos los eventos del calendario.

4. Módulos del programa

La estructura del proyecto es la siguiente:

```
.
|__ app
|    |__ Main.hs
|__ src
|    |__ CalendarOps.hs
|    |__ Common.hs
|    |__ ICS.hs
|    |__ Parse.hs
|    |__ PrettyPrinter.hs
|    |__ RecurrenceOps.hs
|__ Ejemplos
|    |__ Clases.ics
|    |__ ejemplo1.cal
|    |__ ejemplo2.cal
|__ calendario.cabal
|__ README.md
|__ Setup.hs
|__ Informe.pdf
```

4.1. CalendarOps.hs

Tiene las implementaciones de todas las operaciones del calendario mencionadas en la sección anterior así como funciones auxiliares utilizadas. Este módulo es el utilizado por `Main.hs` para manejar los eventos y cambios en el calendario.

4.2. Common.hs

Define los tipos de datos y comandos del intérprete y del calendario. Estos últimos están representados como:

```
data InterCom =
    Compile String
  | Reload
  | Print
  | Quit
  | Help
  | Noop
  | Ops
  | Close
  | Export
  | Import String
```

```
data CalCom =
    NewCalendar Name
  | NewEvent Event
  | ModifyEvent Event
  | DeleteEvent Event
  | SearchEvent String
  | ThisDay
  | ThisWeek
  | ThisMonth
  | AllEvents
  | Category Category
```

Para comenzar a agregar eventos es necesario cargar un calendario, ya sea cargándolo con el comando `:l` o mediante `C 'nombre'`. Si no se hace esto, el lenguaje no sabe a donde debería agregar el evento, por lo que generaría un error. Es por esto que un calendario se define como:

```
data Calendar = Calendar Name [Event] | Null
```

Finalmente, para representar los eventos se tuvo que considerar el caso que no se le ponga categoría o recurrencia, y que el evento podía ser considerado como un evento que abarca todo el día. Es por esto que se utilizó el siguiente tipo de datos:

```
data Event =
  Event
    { summary :: String
    , startTime :: DateTime
    , endTime :: DateTime
    , category :: Maybe Category
    , recurrence :: Maybe Recurrence
    , holeDay :: Bool
    }
```

, donde `type Category = String`, `newtype DateTime = DateTime (Int, Int, Int, Int, Int)` y

```
data Recurrence =
    Daily Int
  | Weekly Int
  | Monthly Int
```

4.3. ICS.hs

Tiene las operaciones `importIcs` utilizada para la importación y `exportToIcs` utilizada para la exportación de archivos **.ics**. Además, tiene funciones auxiliares como `suma3` o `resta3`, las cuales son necesarias para sumar y restar 3 horas a los eventos, ya que Google Calendar utiliza el formato UTC (Tiempo Universal Coordinado).

Si se importa un archivo que fue exportado directamente desde Google Calendar y después se exporta, el archivo generado por el DSL no será el mismo. Esto es porque, por temas de simplicidad, en este calendario utilizamos los atributos básicos de iCalendar. Sin embargo, si este archivo se vuelve a importar a Google Calendar se verá de una forma similar a la anterior.

4.4. Parse.hs

Tiene el esqueleto para el parser capaz de consumir los eventos y comandos mencionados anteriormente.

4.5. PrettyPrinter.hs

Tiene el Pretty Printer del lenguaje. Este sirve para imprimir los calendarios de una manera más legible para el usuario.

Tenemos distintos modos de mostrar un calendario. El printer utilizado para generar los archivos **.ical**, el cual es el mismo que el utilizado por `:p`, está en sintonía con lo que consume el parser. Los eventos del día se presentan en formato de línea de tiempo para poder visualizar intersección de eventos y duración gráfica de cada uno. Los eventos de la semana se presentan en formato de calendario semanal, es decir, cada evento esta separado según el día de la semana en el que ocurren (o comienzan en el caso de eventos de día completo). Los eventos del mes se muestran en formato de almanaque, donde los días que tienen algún evento se diferencian del resto porque tienen un asterisco (*). Finalmente, en

el resto de los casos, los eventos de exhiben en formato de tabla, la cual incluye el nombre del evento, el día y horario de comienzo, el día y horario de finalización y la categoría (si es que tiene).

4.6. RecurrenceOps.hs

Este módulo tiene las operaciones que manejan la recurrencia de eventos. Por ejemplo, un evento que ocurre cada un día utilizará estas funciones para agregar las ocurrencias del mismo en el calendario. Cabe destacar que solo se agregan 10 instancias de un mismo evento, por un lado por simplicidad en el código, y por otro lado porque no se sabe con certeza por cuanto tiempo el usuario realizará la misma acción (una modificación a futuro puede ser agregar la cantidad de ocurrencias que se desean del evento).

4.7. Main.hs

Módulo principal desde donde se lanza la interacción con el usuario.

5. Decisiones de implementación

A lo largo del informe se fueron mencionando algunas decisiones de implementación. En esta sección ampliaremos algunas de ellas con mayor detalle.

- **Sintaxis.** Se tuvo que decidir como se iban a escribir los eventos y los distintos comandos. Para esto se consideraron los campos principales necesarios para cada comando y una forma intuitiva de escribirlo. Por ejemplo, las fechas separadas por '/' y dos fechas distintas separadas por '-'.

  Una de las motivaciones del programa era simplificar la escritura de eventos, es por esto que no se le agregó más complejidad a la sintaxis que una simple `E` para identificar que es un evento.

- **Estructuras de datos.** Las estructuras siguen el concepto intuitivo de la sintaxis. Los eventos tienen los campos fundamentales que necesitan, junto con campos opcionales, los cuales se diferencian por el tipo parametrizado `Maybe`.

  Por otro lado, el día y horario se implementó como una quintupla para poder representar el día, el mes, el año, la hora y los minutos, y de esta forma poder ser más preciso en el momento de agendar un evento. No se representan los segundos porque no se consideró necesaria tanta exactitud en el calendario. En el caso de eventos que duran todo el día, la hora y minuto son 0.

- **Manejo de errores.** El programa le informa al usuario los errores a medida que los encuentra. Estos pueden ser errores de parseo, errores en la línea de comandos o errores producidos por las operaciones propias del calendario, por ejemplo, al agregar un evento que ya existe. De esta forma es más simple para el usuario reconocer cuando se produjo un error y así poder corregirlo.

6. Bibliografía

1. Trabajo práctico 3 de la materia y compilador de la materia Compiladores.
2. https://cabal.readthedocs.io/en/stable/
3. https://hackage.haskell.org/package/prettyprinter-1.7.1/docs/Prettyprinter.html
4. https://hackage.haskell.org/package/time-1.14/docs/Data-Time.html
5. https://hackage.haskell.org/package/parsec-3.1.17.0/docs/Text-Parsec.html
6. https://icalendar.org