Proyecto Final
Estructuras de Datos
Departamento de Electrónica y Ciencias de la Computación
Pontificia Universidad Javeriana Cali

Profesor Carlos Alberto Ramírez Restrepo
carlosalbertoramirez@javerianacali.edu.co

El presente proyecto tiene como objetivo enfrentar a los estudiantes del curso:

- Al diseño e implementación de Tipos Abstractos de Datos.
- Al análisis de la complejidad computacional de la implementación de operaciones primitivas de un Tipo Abstractos de Datos.
- Al análisis y comparación en términos de complejidad computacional de diferentes estrategias de representación para un mismo Tipo Abstracto de Datos.
- Al diseño e implementación de operaciones generales sobre un Tipo Abstracto de Datos.

1. Estructuras Semánticas Lenguaje *Oz*

1.1. Generalidades

Oz es un lenguaje de programación multiparadigma creado por Seif Haridi que integra muy bien y de forma muy coherente los diferentes modelos computacionales. El interprete para el lenguaje de programación *Oz* más algunas funciones y herramienas de uso genérico constituyen lo que se conoce como el sistema de programación ***MOZart***, creado por Peter Van Roy.

Oz, como cualquier lenguaje de programación, involucra un analizador léxico, un analizador sintáctico y un analizador léxico. Cualquier programa que que sea aceptado por estos analizadores puede ser evaluado y eventualmente generará un resultado. De esta manera, la creación de un interpretador para *Oz* y la implementación de *MOZart* es una labor ardua en la que se utilizan una serie de complejas e interesantes estructuras de datos. La construcción de un interpretador para *Oz* es una tarea que se sale del alcance de este proyecto. Sin embargo, en este proyecto usted tendrá que diseñar e implementar algunas de las estructuras que son utilizadas en un intepretador.

Por simplicidad, este proyecto estará centrado en los tipos de datos básicos de *Oz*: los números y los *registros*. Así mismo, se hará énfasis en la operación mediante la cual los variables y los valores se asocian en *Oz* mediante el operador =. Esta operación recibe el nombre de unificación y mediante ella las variables se ligan entre sí de acuerdo a la estructura interna del valor que almacenan. *Oz* utiliza una estructura especial para almacenar las variables y sus valores. Esta estructura es denonminada el *almacén de asignación única* dado que las variables solo pueden ser asignadas una vez, i.e. su valor no puede ser cambiado **nunca**. En este almacén cada variable está asociada a un valor numérico o a un registro, si la variable está ligada. Cuando una variable no está ligada, en el almacén dicha variable estará asociada a un valor especial denotado como

_. De esta manera, ligar una variable `X` con otra variable `Y` producirá que ambas tengan el mismo valor durante toda la ejecución del programa.

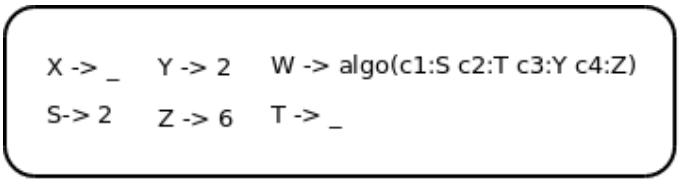
1.2. Unificación

De forma general, puede asumir que en *Oz* se puede unificar un valor `V1` con un valor `V2` mediante una asignación `V1 = V2`. Los valores en *Oz* pueden ser construidos a partir de la siguiente gramática:

$$< \texttt{V} > \ ::= < \texttt{int} > \ | \ < \texttt{float} > \ | \ < \texttt{var} > \ | \ \_ \ |$$
$$\texttt{etiqueta}(\texttt{campo}_1 :< \texttt{V}_1 > \ \texttt{campo}_2 :< \texttt{V}_2 > \ \ldots \ \texttt{campo}_n :< \texttt{V}_n >)$$

donde $< \texttt{int} >$ y $< \texttt{float} >$ se refieren a cualquier valor de tipo entero y tipo flotante, respectivamente. $< \texttt{var} >$ se refiere a una variable y _ se refiere a un valor no ligado. Así mismo, $\texttt{etiqueta}, \texttt{campo}_1, \ldots, \texttt{campo}_n$ son secuencias de carácteres iniciadas con una letra en minúscula y $< \texttt{V}_1 >, \ldots, < \texttt{V}_n >$ son a su vez valores. En consecuencia, los valores pueden ser construidos recursivamente. Los valores construidos con la última regla son conocidos como *registros*. Algunos ejemplos de posibles valores son `8`, `4.3`, `f(c1:10 c2:X c3:_)` y `f(c1:10 c2:X c3:g(algo:3 otro:_ mas:10))`.

El proceso de unificación se lleva a cabo entre las variables que existen en el almacén de asignación de única. El siguiente esquema representa un ejemplo de un posible almacén para un programa de *Oz*:



En este almacén se tienen 6 variables: `X`, `Y`, `W`, `S`, `S` y `T`. Las variables `X` y `T` están no ligadas, mientras que las variables `Y`, `S`, `Z` y `W`. En particular, la variable `W` está ligada a un registro con etiqueta `algo` y campos `c1`, `c2`, `c3` y `c4`, los cuales están unificados con las variables `S`, `T`, `Y` y `Z`, respectivamente.

A continuación se describe el proceso general de unificación. Asuma que se quiere realizar la unificación `V1 = V2`. El algoritmo procederá por casos como sigue:

1. Si `V1` y `V2` son exactamente el mismo valor, el almacén se deja intacto y la unificación termina exitosamente.

2. Si `V1` y `V2` son valores diferentes, la unificación falla.

3. Si `V1` es una variable y `V2=_`, no se hace nada en el almacén y la unificación termina exitosamente.

4. Si `V1` es una variable sin ligar y `V2` es cualquier valor numérico o un registro, en el almacén se liga la variable `V1` al valor asociado a `V2` y la unificación termina exitosamente.

5. Si `V1` es una variable ligada y `V2` es cualquier valor numérico o un registro, si el valor al que está ligado `V1` es igual a `V2`, no se hace nada y la unificación termina exitosamente. En caso contrario, la unificación falla.

6. Si `V1` y `V2` ambas son variables que están ligadas al mismo valor, el almacén se deja intacto y la unificación termina exitosamente.

7. Si `V1` y `V2` ambas son variables que están ligadas a un valor diferente, la unificación falla.

8. Si `V1` y `V2` son registros, se procede de la siguiente manera:

   a) Si los registros tienen diferente etiqueta, la unificación falla.

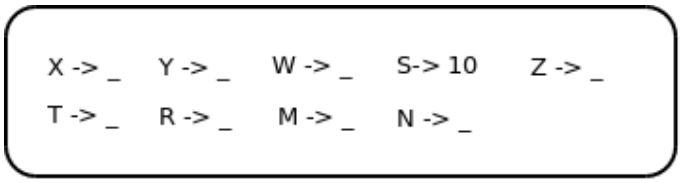
   b) Si los registros tienen igual etiqueta, pero un número diferente de campos, la unificación falla.

   c) Si los registros tienen igual etiqueta, igual número de campos pero las etiquetas son diferentes, la unificación falla.

   d) En cualquier otro caso, se debe proceder a unificar los valores asociados a cada campo (que pueden estar en desorden) y si todas las unificaciones son posibles, la unificación principal termina exitosamente.

1.3. Ejemplos Unificación

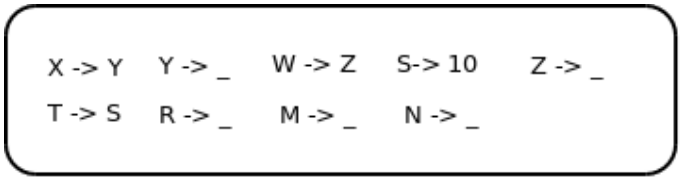
Considere el siguiente almacén:



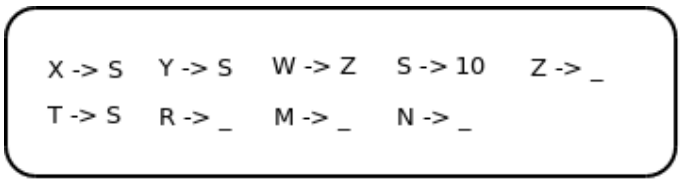
suponga ahora que se realiza la siguiente secuencia de unificaciones:

`X` = `Y`
`W` = `Z`
`S` = `T`

Después de esto, el almacén debería lucir más o menos así:



Suponga que ahora se realiza la unificación `S` = `Y` y la unificación 10 = `X`. Como resultado de lo anterior se tendrá el siguiente almacén:

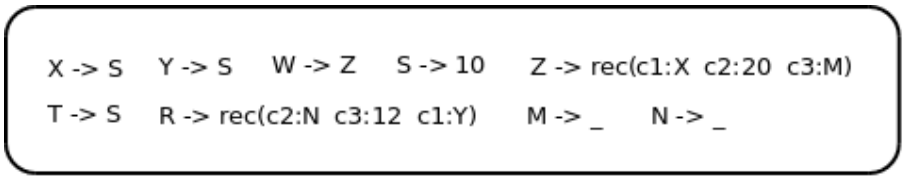


Tenga en cuenta que si en lugar de haber hecho la unificación 10 = `X`, se hubiera hecho la unificación 15 = `X`, el resultado habría sido un error ya que `X` está ligado a `S` que está ligado a 10 y no es posible unificar 10 y 15 puesto que son valores diferentes.

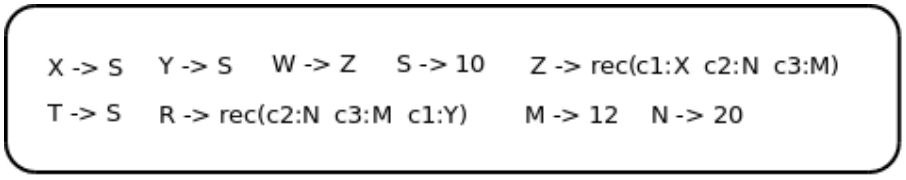
Posteriormente, se realiza las siguientes unificaciones:

`rec(c1 : X c2 : 20 c3 : M) = R`
`rec(c2 : N c3 : 12 c1 : Y) = Z`

Luego, el almacén resultante será el siguiente:



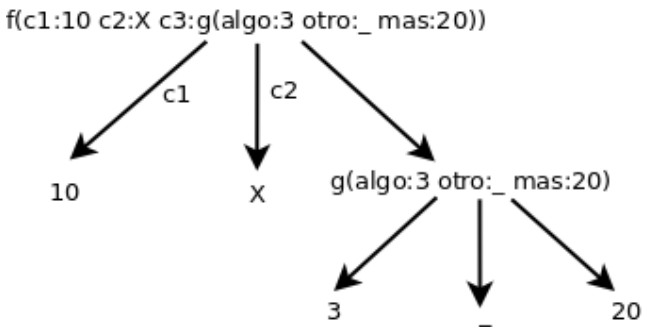
Ahora, si se realiza la unificación `R = Z` se tendrá como consecuencia el siguiente almacén:



2. Estructuras Semánticas

El presente proyecto pretende que usted defina e implemente TADs para el almacen de asignación única utilizado en *Oz* y para representar la gramática asociada a los valores descrita en la sección anterior. En este sentido, cada posible valor puede ser descrito mediante una expresión que puede ser construida composicionalmente de acuerdo a las reglas dadas en la gramática. De esta manera, desde una perspectiva conceptual, cada valor en *Oz* puede ser representado mediante un árbol.

Considere por ejemplo el valor `f(c1:10 c2:X c3:g(algo:3 otro:_ mas:20))`. El árbol asociado a este registro es mostrado en la siguiente figura:



3. Operaciones

A continuación se describe brevemente las operaciones básicas internas para el TAD `Almacen`:

Crear Almacen: Esta operación permite crear una instancia del TAD `Almacen`. Para esta operación se tienen dos posibilidades: (i) crear el almacén vacío y (ii) crear el almacen con un conjunto de variables específicas que inicialmente están sin ligar.

Imprimir Almacen: Esta operación imprime el contenido del almacen en un formato que sea fácil de entender para el usuario. Tenga en cuenta que esta operación involucra que los diferentes valores almacenados sean también mostrados en un formato claro y entendible.

Consultar Variable: Esta operación permite consultar el valor con el que está ligado una variable en el almacén.

Consultar Ligadura Variable: Esta operación permite consultar si una variable está ligada (asociada a un valor) o no en el almacén.

Modificar Variable: Esta operación permite crear una ligadura para una variable.

Agregar Variable: Esta operación permite agregar una variable nueva y sin ligar al almacén.

Obtener Lista Variables: Esta operación permite obtener la lista de variables en el almacén. Tenga en cuenta que el tipo de retorno de esta función debe ser una lista.

Unificar Variables: Esta operación permite unificar dos variables en el almacén.

Por otro lado, el TAD `ValorOz` debe contener por lo menos las siguientes operaciones:

Crear Valor: Esta operación permite construir una instacia del TAD `ValorOz` de cualquier tipo. De esta manera, deben haber diferentes versiones de esta operación según sea el valor específico que se quiere crear. Sin embargo, cada versión debería recibir una cadena que represente en texto el valor a crear.

Obtener Cadena Valor: Esta operación permite obtener la representación de una instancia del TAD `ValorOz` como cadena. No será permitido almacenar en el TAD la representación como cadena para luego ser retornada directamente.

Obtener Etiqueta: Esta operación permite obtener la etiqueta de una instancia del TAD `ValorOz` que esté asociada a un registro.

Obtener Lista Campos: Esta operación permite obtener una lista con los campos de una instancia del TAD `ValorOz` que esté asociada a un registro.

4. Entrega

En el presente proyecto se espera que usted implemente el TAD `Almacen` y el TAD `ValorOz` mediante clases en C++ utilizando y explotando las características de la POO. Además, usted debe crear una clase llamada `Operaciones` en la que hayan funciones que permitan unificar valores (instancias del TAD `ValorOz`) con respecto a una instancia del TAD `Almacen`. Más especificamente, cada grupo de trabajo debe:

1. Definir e implementar el TAD `Almacen`. En su diseño e implementación debe incluir las operaciones indicadas anteriormente y utilizar apropiadamente los conceptos de POO (herencia, polimorfismo, sobrecarga de métodos, sobrecarga de operadores, etc).
2. Definir e implementar el TAD `ValorOz`. En su diseño e implementación debe incluir las operaciones indicadas anteriormente y utilizar apropiadamente los conceptos de POO (herencia, polimorfismo, sobrecarga de métodos, sobrecarga de operadores, etc).
3. Analizar la complejidad de cada operación. No es necesario que se analice detalladamente cada linea de cada operación pero se debe explicar claramente la complejidad que se reporte para cada una de ellas.
4. Explicar claramente las decisiones de implementación realizadas y probar cada una de las operaciones implementadas.

5. Informe

Se debe realizar un informe que corresponderá a un archivo **pdf** escrito en LaTeX. El informe debe contener las explicaciones de las diferentes decisiones que se tomen en la implementación y las conclusiones del proyecto. Más especificamente, el informe debe contener:

- Detalles principales de la implementacion del TAD `Almacen` y del TAD `ValorOz`. Se debe explicar en este documento las funciones auxiliares que hayan sido definidas.
- Análisis de la complejidad de estas implementaciones.
- Conclusiones y aspectos a mejorar. Siendo esta una de las partes más interesantes del trabajo, usted debe **analizar los resultados obtenidos** y **justificar** cada una de sus afirmaciones.

Aclaraciones

1. El proyecto se debe realizar en **grupos de 2 o 3** personas. No se admitirán entregas individuales. Cualquier indicio de copia causará la anulación del proyecto y la ejecución del proceso disciplinario que corresponda de acuerdo a la normativa de la universidad.
2. Para la entrega debe generar un archivo `.zip` o `.tar.gz` con el contenido del proyecto (código fuente, documentación del proyecto y ayudas) el cual debe seguir la convención *Apellido1Apellido2Apellido3ProyectoEstr18II.zip*. Deben incluir un archivo llamado `informe.pdf`, correspondiente al informe del proyecto.

El proyecto debe ser realizado en LaTeXy tener una estructura apropiada y buena redacción. Estos elementos serán evaluados.

3. Los criterios de evaluación del proyecto son los siguientes:

   - Implementación TAD `Almacen`: 15 %
   - Implementación TAD `ValorOz`: 25 %
   - Implementación unificación: 30 %
   - Correcta Utilización POO: 10 %
   - Informe y sustentación: 20 %

4. La entrega del proyecto debe contenter todos los elementos descritos anteriormente. Dicha debe hacerla a través de la plataforma *moodle* a más tardar el día **28 de Mayo a las 9:00am**. La sustentación del proyecto se llevará a cabo los días 28, 29 y 30 de Noviembre..

   La nota final del proyecto será individual y dependerá de la sustentación de cada estudiante. Cada estudiante, después de la sustentación tendrá asignado un número real (el factor de multiplicación) entre 0 y 1, correspondiente al grado de calidad de su sustentación. Su nota definitiva será la nota del proyecto, multiplicada por ese valor. Si su asignación es 1, su nota será la del proyecto. Pero si su asignación es 0.9, su nota será 0.9 por la nota del proyecto. La no asistencia a la sustentación tendrá como resultado una asignación de un factor de 0.

   Tenga muy en cuenta está aclaración. El propósito de la sustentación es que usted demuestre su participación en el proyecto. Por esta razón trabaje a conciencia y prepare muy bien su sustentación. Durante la sustentación se harán preguntas sobre los detalles del proyecto y es posible que los integrantes de cada grupo tengan que explicar sus implementaciones en el tablero.

5. Recuerde, en caso de dudas y aclaraciones puede preguntar al inicio de las clases, asistir al horario de monitoria o enviar un correo con su consulta al profesor.