Facultad de Informática

Programación evolutiva

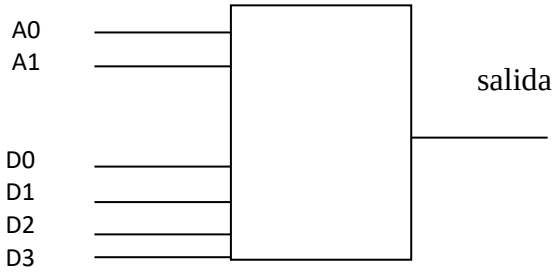
Curso 2021/2022

Práctica 3: Programación genética/Gramáticas evolutivas

Parte 1: La práctica consiste en una aplicación de programación genética o de gramáticas evolutivas que descubra la expresión que resuelva el funcionamiento del multiplexor de 6 entradas. El problema de 6 entradas (2 de dirección y 4 de datos) tiene definidos seis elementos terminales como entradas **A0, A1, D0, D1, D2, D3** y cuatro funciones: **AND, OR, NOT, IF**.

Las tres primeras funciones son los operadores lógicos con dos y un argumento respectivamente. La función **IF** tiene tres argumentos. **(IF X Y Z)** evalúa el primer argumento **X**; si es true, se evalúa el segundo argumento (**Y**) y en caso contrario se evalúa el tercero (**Z**).

Se trata de encontrar un programa que devuelva el valor del terminal **D** que direccionan las entradas **A**. Por ejemplo, si **A0=0** y **A1=1** la función devuelve **D1**. Si **A0=1 y A1=1** la función devuelve **D3.**
La aptitud o fitness de un programa se obtiene analizando el total de aciertos o fallos sobre el total de los 64 casos posibles de prueba (todas las combinaciones de las entradas junto con el valor correcto de salida)



Ejemplos de expresiones obtenidas:

(IF (AND A0 A1) D3 (IF A0 D1 (IF A1 D2 D0)))

(IF A0 D1 (IF A1 D2 D0))

(AND A0 (NOT A0)).

(IF (IF (IF D2 D2 D2) D2) D2).

Parte 2: ampliar la práctica anterior para que también permita resolver el problema del multiplexor de 11 entradas (3 de dirección y 8 de datos), que tiene definidos once elementos terminales como entradas **A0, A1, A2, D0, D1, D2, D3, D4, D5, D6, D7, D8** y cuatro funciones: **AND, OR, NOT, IF**.

En este caso tenemos 2048 casos posibles de prueba (todas las combinaciones de las entradas junto con el valor correcto de salida).

El problema se puede resolver mediante la opción 1 (programación genética) o la opción 2 (gramáticas evolutivas) o ambas

Opción 1: Programación genética

- Se genera una población inicial de programas aleatorios (árboles) usando el conjunto de funciones y terminales posibles. Estos árboles deben ser sintácticamente correctos. Limitamos la profundidad. Métodos Creciente, Completa y Ramped and half.
- La aptitud o fitness de un programa se obtiene analizando el total de aciertos o fallos sobre el total de los 64 casos posibles de prueba (todas las combinaciones de las entradas junto con el valor correcto de salida)
- Operador de Cruce: Intercambiar dos subárboles (elegidos aleatoriamente) entre los dos árboles padres.
- Operadores de Mutación: al menos 3 de los vistos en el material de la asignatura.
- Algún método de controlar el bloating para mejorar la adaptación y evitar programas muy largos

Opción 2: Gramática evolutiva

- Resolver los casos anteriores (parte 1 y parte 2) utilizando gramáticas evolutivas
- Se utilizarán los operadores tradicionales sobre cromosomas de enteros.
- Se podrá seleccionar el número de wraps
- Ejemplos de gramáticas: ANEXO al final del documento

Con ambas opciones la aptitud o fitness de una expresión se obtiene analizando el total de aciertos o fallos sobre el total de los 64 casos posibles de prueba (todas las combinaciones de las entradas junto con el valor correcto de salida)

Se mostrarán las gráficas de evolución y la expresión final obtenida, junto con su fitness.

Entrega

- **Plazo de entrega: 4 de mayo a las 16:00**. Debes entregar mediante la tarea de entrega del Campus Virtual un archivo comprimido con el código java de la aplicación (**proyecto en Eclipse** o **NetBeans)** que incluya una breve memoria que contenga el estudio de las gráficas y los resultados obtenidos con cada función. Aquí se valorarán las conclusiones y observaciones que se consideren interesantes respecto al resultado obtenido.
- No olvidéis nombrar correctamente el proyecto e incluir en el código todas las librerías necesarias. El archivo comprimido y el nombre del proyecto Eclipse tienen que ser **GXXP3**, donde XX es el número de grupo.
- Debes incluir una memoria similar a las entregas anteriores e incluir al final de la memoria una breve descripción del **reparto de tareas** para reflejar lo que ha hecho cada miembro del grupo.

- La corrección se realizará en la sesión del **viernes 6 de mayo** y también se podrán hacer sesiones de corrección en horas de tutoría acordadas.
- Es importante conocer bien la práctica y los aspectos teóricos en los que se basa, pues es lo que determina la calificación final.
- Se realizará control de copias de todas las prácticas, comparando las entregas de todos los grupos de PE. Se considera copia la reproducción total o parcial del código de otros alumnos o cualquier código extraído de Internet o de cualquier otra fuente, salvo aquellas autorizadas explícitamente por el profesor. Si se detecta algún tipo de copia sin justificar se calificará como suspenso.

Importante:

- Si se implementan las dos opciones se puede obtener la calificación completa de la asignatura (100%) sin necesidad de realizar ninguna actividad adicional.

- Si se implementa solo una de las dos opciones, la calificación de las prácticas supondrá un 75% de la nota total y para obtener el 25% de nota por actividad adicional será necesario hacer un trabajo-memoria extra (poe ejemplo, una breve presentación Powerpoint o similar) sobre algún tema de los propuestos en el **Tema 8** de la asignatura o de algún artículo científico sobre la materia.

ANEXO:

Opción 2: Ejemplos de gramáticas:

```
S      ::=    <expr>
<expr> ::=    ( <expr>  AND  <expr>)  |
              ( <expr>  OR  <expr> )  |
              NOT ( <expr> )          |
              IF  ((<expr>) (<expr>) (<expr>)) |
              AO | Al | DO | D1 | D2 | D3
```

```
S      ::= <expr>
<expr> ::= (<expr> <op> <expr> )  |
           <pre-operation>        |
           <var>
<op>   ::=   AND | OR
<pre-operation> ::=  NOT (<expr>)   |
                     IF  ((<expr>) (<expr>) (<expr>))
<var>  ::=  AO |A1 | DO | D1 | D2| D3
```