



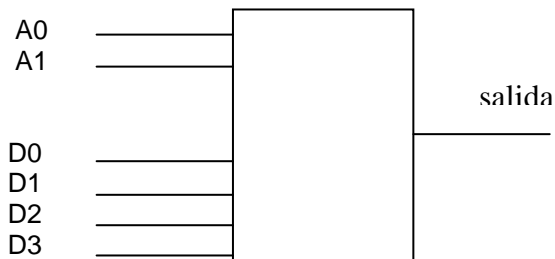
Práctica 3: Programación genética

La práctica consiste en una aplicación de programación genética que descubra la expresión que resuelve 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**. La aplicación permitirá elegir si se utiliza o no la función **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 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:

(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).

Y un ejemplo de expresión sin utilizar la función IF

(AND (OR A0 (OR D2 (NOT A1))) (OR (AND (AND (OR (AND (OR D0 A0) D1) A1) D3) D3) (AND (OR (AND D1 (NOT A1)) (NOT (AND A0 A0))) (OR (AND D0 (NOT (OR A0 (OR A1 (AND (OR (AND (OR (AND A0 A0) A1) D1) A1) D1)))))) (OR A1 (AND (OR (AND A0 A0) A1) D1)))))).

Opcional: 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).

Diseño del algoritmo evolutivo con elitismo:

- Se genera una población inicial de expresiones aleatorias (árboles) usando el conjunto de funciones y terminales definidos. Estos árboles deben ser sintácticamente correctos. Aunque estos árboles tendrán distintos tamaños, hay que establecer un límite a su profundidad.
- Se define la adaptación de cada expresión analizando el número de aciertos o fallos en según el conjunto de ejemplos.
- Aplicar los operadores de selección, cruce y mutación para producir cierto número de generaciones con una elección adecuada de parámetros para el algoritmo.
- Puede estudiarse el uso de técnicas de mejora de los componentes básicos del algoritmo y de extensiones del esquema del algoritmo.
- Se aplicará algún método de control del bloating.

Documentación a entregar

- Hay que enviar al campus virtual antes del **28 de mayo** a las **12:00 a.m.** un archivo comprimido con el código java de la aplicación (proyecto en Eclipse cuyo nombre se corresponde con el nombre del grupo y las siglas P3, por ejemplo G03P3). Es importante seguir esta notación y que el proyecto se llame igual que el archivo comprimido. En el campus virtual el ejercicio está identificado como Practica 3.
- En el archivo comprimido se incluirá una breve memoria con una portada con el nombre de los integrantes del grupo y el número de grupo y que contenga una breve descripción de la representación, de los operadores utilizados, de la función de aptitud y de las mejoras utilizadas. Además se incluirán las gráficas de las cinco mejores ejecuciones obtenidas.
- El día de corrección será en la sesión del laboratorio del 29 de mayo y deberán estar presentes todos los integrantes del grupo.