Planteamiento Del Problema

El tipo de sistema formal aquí utilizado es el ideado por el matemático polaco Emil Post, creador del sistema formal llamado máquina de Post, que es formalmente equivalente a una máquina de Turing.
El sistema formal consta de:
a) un axioma que no es otra cosa que la cadena de caracteres:
MI
b) un universo de proposiciones que consiste en las letras:
M
I
U
de ahí lo de sistema formal MIU.
c) 4 regla de inferencia que son:

1. Si se tiene una cadena cuya última letra sea I, se le puede agregar una U al final
2. Supongamos que se tenga Mx donde x es cualquier cadena del sistema formal MIU. Entonces puede inferirse Mxx.
    - ejemplos:
        - dado MIU, se puede obtener MIUIU
        - dado MUM se puede obtener MUMUM
        - dado MU se puede obtener MUU
3. Si en una de las cadenas aparece la secuencia III, se puede sustituir por U
    - ejemplos:
        - dado UMIIIMU se infiere UMUMU
        - dado MIIII se infiere MIU y MUI
        - dado MIII se infiere MU
4. Si aparece UU en el interior de una de las cadenas, está permitida su eliminación.
    - ejemplos:
        - dado UUU, se obtiene U
        - dado MUUUIII se obtiene MUIII

No se pueden utilizar las reglas en sentido inverso. No importa si alguna regla no se ejecuta a la perfección. tener en cuenta que lo que perseguimos es, a partir del axioma MI intentar obtener MU en alguna de las inferencias. A cada cadena inferida mediante la aplicación de una regla se le llama teorema.
Se puede observar fácilmente que dada la naturaleza de las reglas de inferencia, todos los teoremas van a empezar por M
Para ello construiremos un programa que haga lo siguiente:
Dado el axioma MI:

1. ejecutar sistemáticamente las 4 reglas de inferencia en el. Si se infiere teorema, almacenamos un registro con la información en una base de datos.

2.tomamos esa base de datos, la recorremos entera y cada registro le aplicamos las 4 reglas de inferencia y el resultado lo almacenamos.
3.ejecutamos el paso 2, iterativamente un número determinado de veces.
Usando toda la potencia del Access, que tiene su límite en 2GB por fichero, solo he podido ejecutar 19 iteraciones:

NIVEL	TEOREMAS
1	2
2	3
3	5
4	9
5	16
6	32
7	61
8	117
9	224
10	437
11	838

NIVEL	TEOREMAS
12	1613
13	3089
14	5939
15	11353
16	21790
17	41732
18	80208
19	153934

Como puede verse el número de teoremas es acumulativo y se duplica en cada iteración. Existe además un límite en la longitud de las cadenas de caracteres que se pueden representar en Access.
por otra parte es verdad que a un teorema dado se le pueden aplicar varias reglas simultáneamente. Para que el sistema formal se desarrolle paso a paso, el programa que he creado las ejecuta paso a paso. Cada teorema generado solo tiene una regla

aplicada. Se podría hacer una variante del programa donde dada una cadena o teorema, se le apliquen sistemáticamente todas las reglas el número de veces que sea necesario hasta reducirlo a un teorema al que ya no se pueden aplicar reglas, lo cual en este sistema de inferencia sería imposible puesto que todos los teoremas empiezan por M y por la regla 2, nunca acabaríamos. La única opción sería parar el bucle en una iteración n la que solo se pudiera aplicar la regla 2.
En fin, las ramificaciones del problema son muchísimas. Habría que ver asimismo si un teorema dado ya solo crece repitiéndose, entonces quizá no tendría sentido seguir iterándolo y explorar solo los teoremas que van teniendo resultados diferentes.
También se puede cambiar de axioma y ver si es capaz de llegar a MU.
Lo cierto es que con el axioma MI y el objetivo MU, no ha podido inferirse MU en 19 iteraciones. He probado otros axiomas y nada, imposible hallar MU. A ver si algún lector lo encuentra o bien demuestra que es imposible encontrarlo, lo cual es muy difícil, porque habría que probarlo para todos los axiomas finitos.

Aplicación En VBA Para Access

Y esta es la segunda parte y no menos importante de este post. Hay que hacer una aplicación para inferir MU por vía sistemática. Esto se puede hacer usando el lenguaje que uno quiera y la tecnología de base de datos que uno prefiera o que mas domine.
Yo conozco varios lenguajes con mayor o menor grado de destreza, pero tengo mi sistema para hacer una aplicación de forma rápida y efectiva. Imagino que cada uno de vosotros tendrá su sistema. Yo os voy a explicar el mío que lo uso desde hace muchos años y me ha dado muy buen resultado.
Utilizo una base de datos Access con programación en Visual Basic, por las siguientes razones:

1. En casi todas las aplicaciones se usa una base de datos de una u otra forma. Y usar Access es la forma mas sencilla y rápida de usar una base de datos
2. Access simplifica la creación de tablas, consultas y macros. Es asombroso como hay muchas aplicaciones que se hacen en mas de 50% solo usando consultas y macros, ahorrando mucho tiempo de programación.
3. Lenguaje e IDE integrado en la base de datos. En Access todo está dentro de un único fichero: base de datos, consultas, macros, programas, informes, formularios, menús etc.
4. IDE muy conseguido para el desarrollo de aplicaciones.
5. Uso del lenguaje de forma tradicional o enfocado a objetos. El lenguaje VBA es sorprendente porque se puede usar de forma tradicional, sin orientación a objetos o sea de forma procedural. O bien, usando los modelos de objeto de Microsoft, utilizar las características de orientación a objetos. Todo ello de forma natural y fácil sin problemas y sin comeduras de coco. por contra otros lenguajes modernitos, son muy exigentes en el tipado, en la orientación a

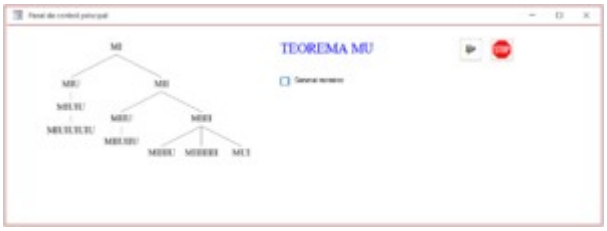
objetos imponiendo unos corsés muy desagradables. Si algo no tiene que haber en informática son corsés.

Como he dicho antes el modelo de datos, las tablas se puede diseñar directamente en la aplicación Access, dicho modelo para resolver este problema, sería:

TABLA TEOREMAS
TEOREMA Texto largo -
NIVEL Doble 8
REGLA_APLICADA Entero largo 4
ANTECEDENTE Texto largo -
CONTROL Byte 1
TABLA X_TEMPORAL
TEOREMA Texto largo -
NIVEL Doble 8
REGLA_APLICADA Entero largo 4
ANTECEDENTE Texto largo

En la tabla TEOREMAS almacenaremos los teoremas inferidos con la regla de inferencias. En la tabla X_TEMPORAL almacenaremos el último cálculo de nivel.

A continuación voy a presentar la función que hace estos cálculos, y un enlace a la aplicación para su descarga y prueba. Lo único que hay que tener instalado es una versión completa y actualizada de Microsoft Office.



Una vez que arranca la base de datos, aparece el menú de la imagen que solo tiene una opción. Pulsamos en ella y el programa empieza los cálculos. Al finalizar, se muestra la base de datos y una consulta totalizadora de teoremas. la función es la siguiente:

```
Function buscar_TEOREMAS()
DoCmd.RunSQL "DELETE * FROM TEOREMAS;"
DoCmd.RunSQL "DELETE * FROM X_TEMPORAL;"
Dim MYWS As Workspace
Dim MYDB As Database
Dim XTEOREMA As Recordset
Dim XTEMPORAL As Recordset
Dim x_nivel As Double
Dim x_axioma As String
Dim x_cadena As String
Set MYWS = DBEngine(0)
Set MYDB = MYWS.Databases(0)
Set XTEOREMA = MYDB.OpenRecordset("TEOREMAS")
Set XTEMPORAL = MYDB.OpenRecordset("X_TEMPORAL")
```

```
Let x_axioma = "MI"
Let x_nivel = 1
Let x_cadena = x_axioma
'APLICACIÓN REGLA1 AL AXIOMA
If Right(x_cadena, 1) = "I" Then
XTEMPORAL.AddNew
XTEMPORAL.Fields("TEOREMA").Value = x_cadena & "U"
XTEMPORAL.Fields("NIVEL").Value = x_nivel
XTEMPORAL.Fields("REGLA_APLICADA").Value = 1
XTEMPORAL.Fields("ANTECEDENTE").Value = x_cadena
XTEMPORAL.Update
End If
'APLICACIÓN REGLA2 AL AXIOMA
If Left(x_cadena, 1) = "M" Then
XTEMPORAL.AddNew
XTEMPORAL.Fields("TEOREMA").Value = x_cadena & Right(x_cadena,
Len(x_cadena) - 1)
XTEMPORAL.Fields("NIVEL").Value = x_nivel
XTEMPORAL.Fields("REGLA_APLICADA").Value = 2
XTEMPORAL.Fields("ANTECEDENTE").Value = x_cadena
XTEMPORAL.Update
End If
'APLICACIÓN REGLA3 AL AXIOMA
If InStr(x_cadena, "III") > 1 Then
XTEMPORAL.AddNew
XTEMPORAL.Fields("TEOREMA").Value = Replace(x_cadena, "III", "U")
XTEMPORAL.Fields("NIVEL").Value = x_nivel
XTEMPORAL.Fields("REGLA_APLICADA").Value = 3
XTEMPORAL.Fields("ANTECEDENTE").Value = x_cadena
XTEMPORAL.Update
End If
'APLICACIÓN REGLA4 AL AXIOMA
If InStr(x_cadena, "UU") > 1 Then
XTEMPORAL.AddNew
XTEMPORAL.Fields("TEOREMA").Value = Replace(x_cadena, "UU", "")
XTEMPORAL.Fields("NIVEL").Value = x_nivel
XTEMPORAL.Fields("REGLA_APLICADA").Value = 4
XTEMPORAL.Fields("ANTECEDENTE").Value = x_cadena
XTEMPORAL.Update
End If
DoCmd.OpenQuery "ANEXAR_A_TEOREMAS"
DoCmd.OpenQuery "RESETEAR_CONTROL"
```

```
Do While True
Let x_nivel = x_nivel + 1
XTEMPORAL.MoveFirst
Do While Not XTEMPORAL.EOF
x_cadena = XTEMPORAL.Fields("TEOREMA").Value
'APLICACIÓN REGLA1 AL TEOREMA
If Right(x_cadena, 1) = "I" Then
XTEOREMA.AddNew
XTEOREMA.Fields("TEOREMA").Value = x_cadena & "U"
XTEOREMA.Fields("NIVEL").Value = x_nivel
XTEOREMA.Fields("REGLA_APLICADA").Value = 1
XTEOREMA.Fields("ANTECEDENTE").Value = x_cadena
XTEOREMA.Update
End If
'APLICACIÓN REGLA2 AL TEOREMA
If Left(x_cadena, 1) = "M" Then
XTEOREMA.AddNew
XTEOREMA.Fields("TEOREMA").Value = x_cadena & Right(x_cadena,
Len(x_cadena) - 1)
XTEOREMA.Fields("NIVEL").Value = x_nivel
XTEOREMA.Fields("REGLA_APLICADA").Value = 2
XTEOREMA.Fields("ANTECEDENTE").Value = x_cadena
XTEOREMA.Update
End If
'APLICACIÓN REGLA3 AL TEOREMA
If InStr(x_cadena, "III") > 1 Then
XTEOREMA.AddNew
XTEOREMA.Fields("TEOREMA").Value = Replace(x_cadena, "III", "U")
XTEOREMA.Fields("NIVEL").Value = x_nivel
XTEOREMA.Fields("REGLA_APLICADA").Value = 3
XTEOREMA.Fields("ANTECEDENTE").Value = x_cadena
XTEOREMA.Update
End If
'APLICACIÓN REGLA4 AL TEOREMA
If InStr(x_cadena, "UU") > 1 Then
XTEOREMA.AddNew
XTEOREMA.Fields("TEOREMA").Value = Replace(x_cadena, "UU", "")
XTEOREMA.Fields("NIVEL").Value = x_nivel
XTEOREMA.Fields("REGLA_APLICADA").Value = 4
XTEOREMA.Fields("ANTECEDENTE").Value = x_cadena
XTEOREMA.Update
End If
XTEMPORAL.MoveNext
```

```
Loop
DoCmd.RunSQL "DELETE * FROM X_TEMPORAL;"
DoCmd.OpenQuery "ANEXAR_A_XTEMPORAL"
DoCmd.OpenQuery "RESETEAR_CONTROL"
If x_nivel = 19 Or x_cadena = "MU" Then
Exit Do
End If
Loop
XTEMPORAL.Close
XTEOREMA.Close
MYDB.Close
MYWS.Close
DoCmd.RunSQL "DELETE * FROM X_TEMPORAL;"
DoCmd.OpenTable "TEOREMAS"
DoCmd.OpenQuery "TEOREMAS_TOTALES"
End Function
```