Dos aplicaciones de las tablas de decisión

Autor: Luis Roberto Ojeda Ch.
Ingeniero, M.S. en Computación
U. C. L.A.
Universidad Nacional de Colombia
Departamento de Sistemas
Bogotá

Las tablas de decisión como instrumento de programación han sido objeto de estudio para los investigadores en forma un tanto irregular, apareciendo nuevos desarrollos concentrados en determinados períodos. Hacia la mitad de los años setenta es frecuente encontrar uno o dos artículos sobre el tema en cada entrega de las "communications of the ACM".
En los dos últimos años sólo aparece un artículo sobre tablas de decisión en la misma publicación. Para los amantes del tema esto no puede representar obsolescencia definitiva de la técnica. Muchas horas hay invertidas en ellas muchas formas ingeniosas se han producido.
El presente artículo presenta una introducción de las tablas en la programación de autómatas. Hay la esperanza mínima de que esta forma complete el panorama corriente de representación de autómatas. Se desea en el futuro hacer la evaluación de las opciones de codificación de los autómatas.
La segunda parte del artículo trata la conversión de programas a tablas de decisión. Esto se debe ser mejor que el tratamiento tradicional de convertir el programa en un diagrama de flujo, por razones de manejo de espacio en el medio físico de salida, y de tiempo de ejecución.
Es importante mencionar que esta aplicación sigue los lineamientos de una publicación sobre el problema (John Cavoaras, University of Glasgow) y que las proposiciones condicionales se reducen a la forma lógica , para mayor claridad. El desarrollo formulado aquí pretende dar un punto de partida consistente para el análisis de las posibilidades avizoradas en el artículo de cavoaras, reseñado dentro de este trabajo.

Primera Aplicación

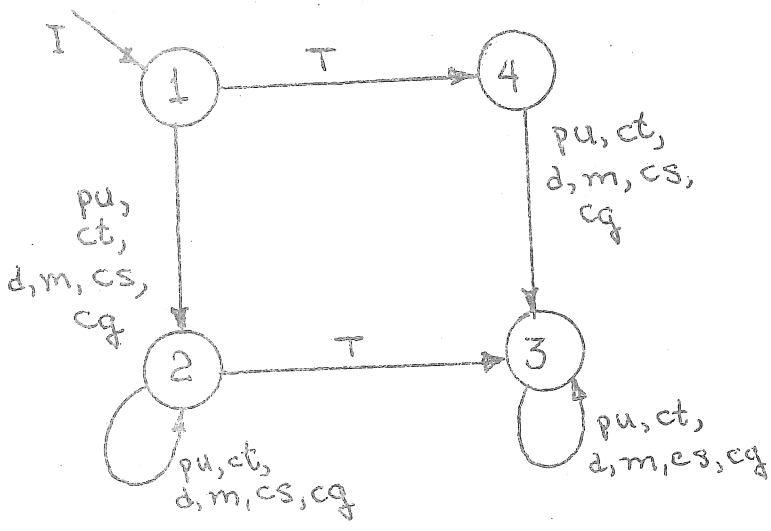
Conversión de un autómata determinístico finito en una tabla de decisión simple.
Aparte de su uso como analizador léxico, el autómata puede realizar funciones de reconocimiento en otros ambientes.
El siguiente caso servirá para ilustrar el procedimiento de conversión.

El artículo 59 de la Constitución de la República de Colombia establece:
"Para ser elegido contralor general de la república se requiere ser colombiano de nacimiento y en ejercicio de la ciudadanía; tener más de 35 años de edad; tener título universitario en derecho o en ciencias económicas o financieras. Además, haber desempeñado en propiedad alguno de los cargos de Ministro del Despacho, Magistrado de la Corte Suprema de Justicia, Consejero de Estado, Contralor General de la República; o habe sido Miembro del Congreso Nacional por lo menos durante cuatro años, o Profesor Universitario en las cátedras de ciencias jurídico económicas, durante un tiempo no menor de cinco años".

Para obtener un autómata simplificado, asumiremos el siguiente alfabeto:

T	:	Implica el cumplimiento de: Más de 35 años de edad. Colombiano de nacimiento Ejercicio de la ciudadanía Título en derecho o en ciencas económicas o financieras
d	:	Ministro del despacho (en propiedad)
m	:	Magistrado (en propiedad)
cs	:	Consejero de estado (en propiedad)
cg	:	Contralor general (en propiedad)
ct	:	Miembro del Congreso Nacional ($\geq$ 4 años)
pu	:	Profesor universitario jurídico-económicas ($\geq$ 5 años)

El autómata será:



El procedimiento propuesto para la conversión es el siguiente:

I) Tomar el estado de partida (estado I) del autómata.

II) Formar una condición múltiple con signos del alfabeto.
(Condición múltiple: Es un conjunto de condiciones sencillas.
Cada condición sencilla es un subconjunto del alfabeto que marque algún(os) arco(s) del autámata).
Si de un estado hay salida a más de un estado formar la condición K = Estado.
Formar siempre (de todas maneras) la condición K = Estado para el estado de aceptación (Estado Final).
Si hay un número n de estados cada uno con salida solamente a un estado, dar lugar para la condición K = Estado para n - 1 de tales estados.
El número de condiciones es igual al número de marcas diferentes en los arcos, más el número de estados con más de un estado de transición. Si hay estados (n) con transición a un sólo estado, se suman n-1 condiciones.

III) Acciones
Formar una acción K = Estado con cada estado.
(una acción por cada estado del autómata)"

IV) Acciones
Formar las acciones: Volver (para regresar a aplicar la tabla); Para 1 (para aceptar) y Para 2 (para recharzar).

V) Reglas de decisión
Formar - por cada estado con más de una salida - una regla de decisión para cada arco que marque una transición desde ese estado.
Si de todo estado hay más de una salida el número de reglas de decisión es igual al número de arcos.
Para (n-1) de los n estados con sólo una salida aplicar el mismo concepto anterior -V-

LLenado de la tabla

I) U = I (estado de partida); nn = 0
k = primer arco saliente de I

II) (Condiciones).
Marcar S en la casilla K = valor de U
Marcar S en la casilla identificada por k.
Marcar N en el resto de casillas de condiciones
(Acciones).
Marcar (X) en la casilla correspondiente (K =) al estado al que se hace transición con éste arco.
Marcar (X) la casilla correspondiente a 'volver'
Aplicar con todos arcos salientes del estado actual el paso 2 para formar diferentes reglas de decisión.
variark
Tomar un nuevo estado conectado directamente con los ya procesados.
Si de él sale mas de un arco, hacer U = Estado k = primer arco saliente. Aplicar el paso 2.

Si del estado sale solamente un arco y nn $\neq 0$
formar en la s condiciones la condición K = Estado; Hacer U = estado, k = arco saliente.
Aplicar el paso 2.
Si el estado sale solamente un arco y nn = 0, encontrar el estado al que se transita. Si el tránsito es al mismo estado, buscar otro nuevo estado conectado directamente a los ya procesados. Si no hay, seguir como si el tránsito fuese a otro estado .
Si el tránsito es a otro estado, suponer que éste arco sale y regresa en este nuevo estado. Verificar que no se repite un arco. Si es así, eliminar la repetición k = primer arco saliente.
Formar las reglas correspondientes a este estado según el paso 2 excepto que no se marcará S en la casilla (K = valor de U) y quedará en blanco.
nn = 9

Para cerrar

Al finalizar con los estados, formar la regla de aceptación negando todas las condiciones sencillas (condición múltiple) y afirmando la condición (K =) (estado final). En las acciones marcar (X) Parar 1.
Formar la regla de rechazo asumiendo cualquier conjunto de condiciones no contempladas en las reglas anteriores, y marcar (X) la acción Parar 2.

La aplicación del procedimiento origina la siguiente tabla de decisión para el autómata definido antes:

pu, ct, d, m, cs, cg	N	S	S	N	S	N	O
T	S	N	N	S	N	N	T
K = 1	S	S	N	N	N	N	R
K = 2	N	N	S	S	N	N	O
K = 3	N	N	N	N	-	S	S
K = 1							
K = 2		X	X				
K = 3				X	X		
K = 4	X						
volver	X	X	X	X	X		
Parar 1						X	
Parar 2							X

Las formas conocidas de representación para un autómata determinístico finito incluyen:

a) Una estructura de listas
Gries, Compiler Construction for digital computers, J. Wiley, 1971 Ch. 3
Cada estado con K arcos salientes se representa con 2K+2 palabras. La primera contiene el nombre del estado. La segunda el valor de K.
Cada par subsiguiente contiene un marcador de arco y un apuntador al comienzo de la representación del estado al que se transfiere.
No hay indicación de eficiencia.

b) Una cláusula CASE (cuando sea disponible en el lenguaje de representación del autómata)
Waite y Goos (Compiler Construction, Springer Verlag 1984 Ch. 6) establecen solamente que el costo es considerable con esta alternativa. Lo comparan con la matriz de transición (generalmente costoso en términos de memoria) y sólo reconocen estas dos posibilidades para representar el autómata.

c) La matriz de transición

While (símbolo todavía incompleto) do
estado: = tabla [estado, nuevo caracter]

Aho y ullman (principles of Compiler Design, Addison Wesley 1979 Ch. 3) señalan que éste arreglo bidimensional indizado con estados y caracteres es el más rápido pero puede consumir mucho espacio [varios centenares de estados - filas - por 128 caracteres - columnas -]
Los autores citados en b) y c) presentan formas de comparación de los métodos conocidos.
La facilidad Lex de UNIX proporciona el tratamiento de matriz de transición.
Para ubicar el tratamiento propuesto en este artículo se resalta : a) su parentesco con la matriz de transición, basada en estados y caracteres como entradas para recorrer los estados.
b) La variedad de formas de programar una tabla de decisión para sacar ventajas de su estructura o de los datos que comunmente maneja.
Para tener sólo una posibilidad interesante ver: "Extending the information theory Approach to Converting Limited- entry Decision Tables to Computer programs".
Keith Shwayder, Comm ACM Sept. 1974 Vol. 17 nr 9 p. 532.
(El objetivo es minimizar el tiempo de ejecución del programa resultante).

c) Finalmente, puesto que la conversión de una tabla de decisión en un programa es algorítmica, puede pensarse en variar las posibilidades para un mismo autómata.

Segunda aplicación

Conversión de programas a tablas de decisión
Lograr un procedimiento eficiente para convertir un programa (escrito en un lenguaje particular) en una tabla de decisión es un objetivo seductor: en primer término, la tabla de decisión, que

es un vehículo natural de comunicación, se convierte en un sustituto ventajoso del diagrama de flujo (todavía útil en programación) por cuanto el tamaño del programa para lograr una tabla de decisión a partir de un código es necesariamente inferior al que corresponde a la obtención de un diagrama de flujo.
En segundo lugar, existiendo métodos diversos para convertir automáticamente una tabla de decisión en un programa de computador, y algoritmos para optimizar una tabla (information theory applied to the conversion of decision tables to computer programs S. Ganapathy, V. Rajaraman, Comm. ACM Sept 1973) (The development of Decision Tables via parsing of Complex Decision situations, Comm ACM, Jun 1973) se puede pensar en la conversión con el propósito de optimizar y eventualmente, por lo menos como ejercicio teórico, con el objetivo de disponer de codificación en otro lenguaje.
El procedimiento que viene enseguida está fundamentado en el trabajo de John Cavouras (On the conversion of programs to Decision Tables: Method and Objectives Comm. ACM Agosto, 1974) en este trabajo los objetivos propuestos son depuración del programa y optimización. Aunque Cavouras estima alcanzado el objetivo pese a que en algún caso un programa puede conducir a dos tablas diferentes, lo que le lleva a concluir que las tablas de decisión no son muy útiles para determinar la semántica de programas de computación, vale la pena un replanteamiento de base del procedimiento de conversión, con el propósito de analizar mejor el planteamiento teórico y las referencias bibliográficas del trabajo citado.
El procedimiento fué desarrollado pensando en que las proposiciones condicionales usadas en el programa son solamente del tipo IF lógico, para simplificar.

Procedimiento

1) K = 1 identificador de regla de decisión
Símbolo = S
ra = 1 rótulo actual de condición
rac = 1 rótulo actual de acción

2) Visitar y marcar las proposiciones del programa hasta encontrar una proposición condicional .
Marcarla.

3) Proposición condicional
Llenar el rótulo actual de condición (ra)
Abrir una columna (regla de decisión K) y frente al rótulo actual de condición llenar con S.
Poner en la pila la dirección de la rama N y el valor de ra (para recordar la condición que se niega por éste camino).
Seguir la trayectoria S.

4) Si encontramos una proposición condicional, marcarla ra = ra+1
Llenar el rótulo actual de condición
Llenar la entrada correspondiente (columna K) con S
Poner en la pila la dirección de la rama N y el valor de ra.

Si encontramos otra proposición condicional, repetimos 4.
Si encontramos una acción, llenamos el rótulo actual de acción y la entrada correspondiente en la columna K con X, marcamos la acción, hacemos rac = rac+1, y repetimos con las acciones que prosigan hasta encontrar un Alto.

5) Cuando aparece un Alto sacamos de la pila: ra(p) (tomado de la pila) identificará la línea de condición.
Si ra(p) = 1 hacer símbolo = N
Hacemos K = K + 1 Abrimos una nueva columna (columna K)
Sea T: tabla de condiciones (parte superior de la tabla de decisión).

En T(ra(p), K) escribimos N
Si ra(p) > 1 Hacemos:

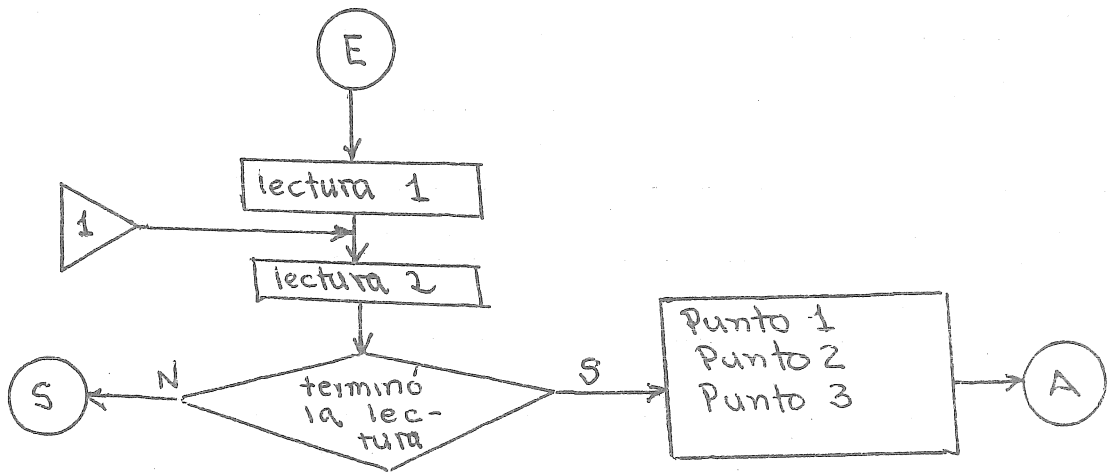
n = 1
Si T(ra(p) - n , K - 1) ≠ b
Hacemos T(ra(p) - n , K) = T(ra(p) - n, K - 1)
de otro modo hacer n = n+1 y repetirnos este paso.

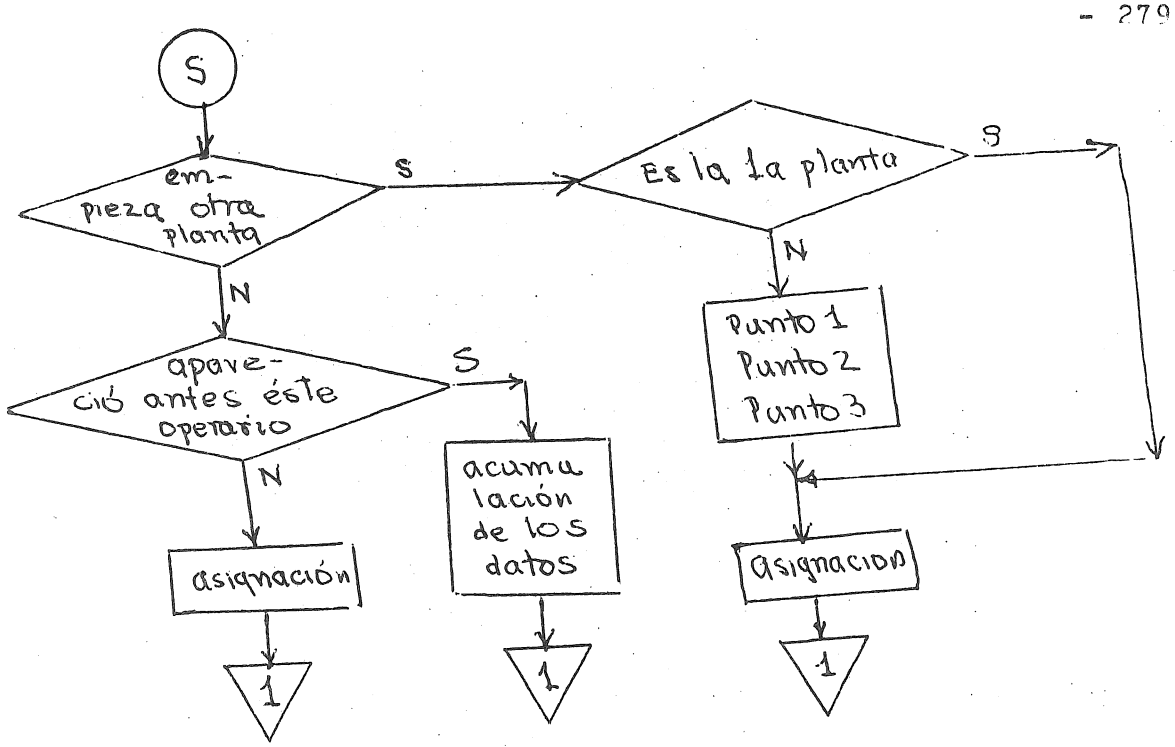
Si ra(p) > 2

Hacemos T(nn, K) = Símbolo
para nn desde 1 hasta ra(p) - 2

Con la dirección de la instrucción (rama N) pasamos a 4.

Un Alto es una proposición condicional que aparece luego de una o mas acciones.
Si esta proposición condicional ya está marcada y es la primera de la tabla, se adiciona la acción volver antes de proceder con 5.
De otro modo se adiciona la acción 'ejecutar nueva tabla', antes de proceder con 5.
La nueva tabla comenzará con la proposición condicional que originó el Alto.
Es más fácil seguir el procedimiento utilizando un diagrama de flujo en lugar del código correspondiente en algún lenguaje.





La tabla de decisión resultante es:

Terminó la lectura?	S	N	N	N	N
Empieza otra planta?		S	S	N	N
Es la 1ra planta?		S	N		
Apareció antes éste operario?				S	N
Punto 1	X		X		
Punto 2	X		X		
Punto 3	X		X		
Alto	X				
Asignación		X	X		X
Acc. de datos				X	
Lectura 2		X	X	X	X
Volver		X	X	X	X