Método para la construcción de programas basado en técnicas de descripción de estados

Lic. Eduardo Miranda*

Introducción

El comportamiento de ciertos programas de computadora, puede ser visualizado como ejecutando una serie de pasos en los que la mayor parte de las tareas útiles están asociadas a ciertos estados o eventos bien determinados. Tal es el caso, entre otros, del software que cumple funciones de interfase entre hombre y máquina.

El presente trabajo explica la utilización de un método simple para la concepción y realización de programas a partir de un diagrama de estados que sirve de especificación al mismo.

Técnicas de descripción de estado

El estado de un sistema en un momento dado, se puede definir como el conjunto de propiedades relevantes que el sistema exhibe en dicho instante. Implícita en el estado del sistema, se encuentra toda la información acerca de los estados y entradas previas que éste necesita recordar para evolucionar correctamente frente a nuevos estímulos.

En muchos casos, aunque no en todos, la evolución y respuestas de un sistema pueden ser convenientemente expresados mediante un diagrama de estados.

Un diagrama de estados está compuesto por elementos de dos tipos: nodos y arcos.

- Los nodos son dibujados como círculos y se los usa para representar estados (Fig. 1). Los distintos estados del sistema son identificados mediante un nombre inscripto en el interior del círculo.
- Los arcos, representan transiciones entre estados, se las dibuja[3] como una flecha. Asociada a cada flecha (Fig. 1) se encuentran una o más condiciones o eventos, los cuales en caso de verificarse dan lugar a la transición y una lista de acciones con las que el sistema responde cuando dicha transición ocurre.

Como ejemplo de aplicación del diagrama de estado a la descripción de un sistema se tomará el caso de una máquina expendedora de golosinas, que vende chicles y chocolates cuyo costo es de 15 y 20 centavos, respectivamente. La máquina acepta monedas de 5 y 10 centavos. Cada vez que una moneda es introducida en ella, la máquina emite un sonido si dicha moneda es de 5 o dos si es de 10. Dos botones selectores permiten elegir la golosina deseada. Existe asimismo un botón de devolución que permite recuperar el dinero ingresado si no se ha efectuado la entrega de ninguna golosina.

Se llega al diagrama de estados con los siguientes pasos:

1º Definición del estado inicial del sistema (Fig. 2).

2º Estando en el estado "cero", el sistema es excitado mediante la inserción de una moneda. Si ella es de 5 centavos, el sistema pasa al estado "cinco" emitiendo un "click", por el contrario si la moneda es de 10 centavos se pasa al estado "diez" produciendo un doble "click" (Fig. 3).

3º Estando en el estado "cinco", los estímulos que el sistema reconoce son (Fig. 4):

a) Monedas de 5 y 10 centavos, las cuales en caso de introducirse, ocasionan una transición al estado "diez" o "quince", respectivamente y la emisión de uno o dos "clicks" según corresponda.

b) Botón de devolución ("dev.") en caso de oprimirse este botón la máquina devuelve 2 centavos y vuelve al estado inicial ("cero").

4º Diagrama de estado correspondiente al funcionamiento de la máquina expendedora (Fig. 5). En el caso de presionar el botón de chicle, estando en el estado "veinte" la máquina responde entregando el chicle y además los 5 centavos de vuelto ("click + 5").

El siguiente ejemplo, ilustra cómo se puede usar un diagrama de estados para modelar el comportamiento

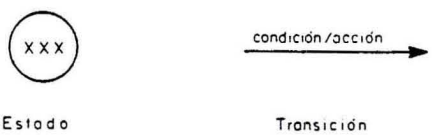

Fig. 1 — Elementos de un diagrama de estado.

Fig. 2 — Estado inicial del sistema.

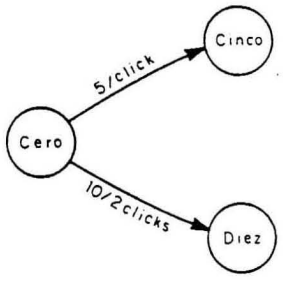

Fig. 3

* Lic. en Análisis de Sistemas.

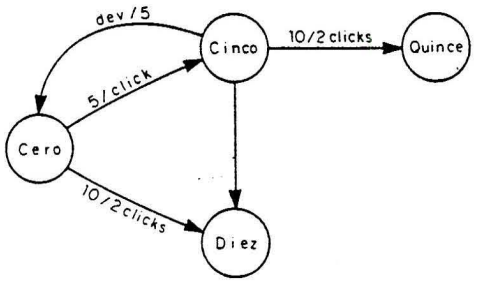

Fig. 4

de un programa. La Fig. 6 muestra parte de un diagrama de flujo de un programa, el cual después de haber ejecutado un cierto número de pasos, llega a la situación que denominamos e_A, en ella permanece hasta que una de las condiciones se verifique. Dependiendo de cuál de éstas haya sido, ejecuta la secuencia de acciones "$A_1 - A_2 - A_3$" ó "$A_2 - A_4$" y pasa a uno de los dos estados posibles e_B ó e_C.

La Fig. 7 describe el programa en términos de un diagrama de estado transición. En este ejemplo, se asume que C_1 y C_2 son excluyentes, si así no fuera se debería tener presente que el diagrama de flujo contiene información implícita en la secuencia de control, que no está presente en el diagrama de estado. En otras palabras si C_1 y C_2 fueran ambas verdaderas, observando el diagrama de flujo se sabría que las acciones que se ejecutarán serían "$A_1 - A_2 - A_3$" cosa que no se puede deducir del diagrama de estado; si bien esta situación no es insalvable, debe prestársele atención a los efectos de que el comportamiento del programa no sea el resultado de una decisión, muchas veces fortuita, del orden en que se evalúan las condiciones.

Máquinas de estados finitos

El diagrama de estado no es más que la representación gráfica del objeto "máquina de estados finitos".

Formalmente una máquina de estados finitos es el arreglo sextuple:

$$(S,\ s_0,\ I,\ 0,\ F,\ G)$$

donde:

S: Es el conjunto de estados posibles.

s_0: Es el estado inicial.

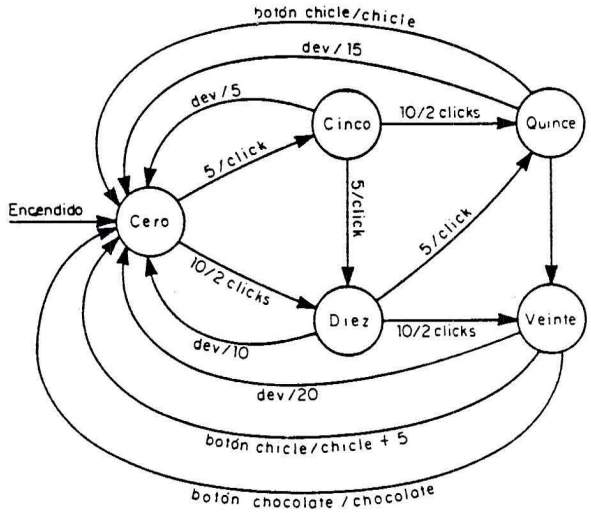

Fig. 5 — Diagrma de estados de una máquina expendedora de golosinas.

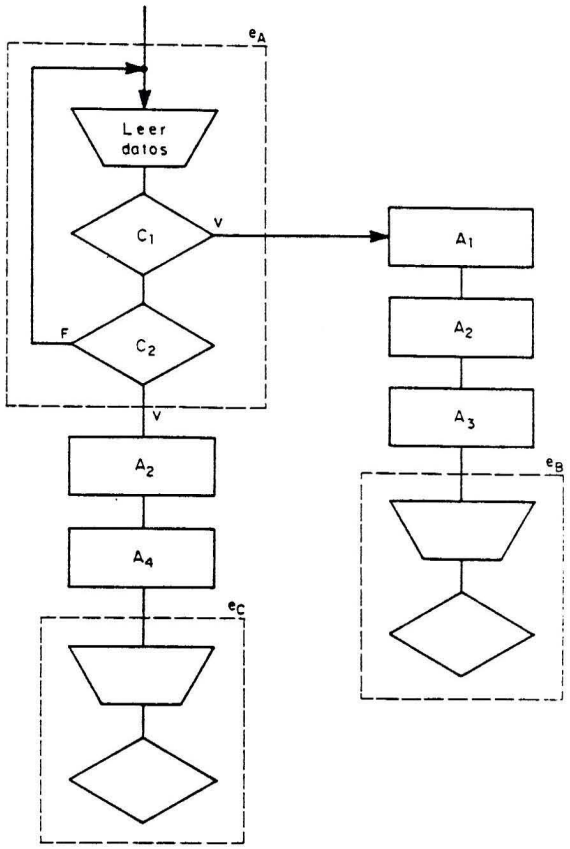

Fig. 6 — Diagrama de flujo.

I: Es el conjunto de entradas o estímulos a los que el sistema responde.

0: Es el conjunto de salidas o acciones con las cuales el sistema responde.

F: $(S \times I)$ es la función que determina, dados un cierto estado y un cierto estímulo, cuál va a ser el próximo estado.

G: $(S \times I)$ es la función que, dados un cierto estado y un cierto estímulo, determina la salida que el sistema produce en respuesta a dicho estímulo.

Tomando el ejemplo de la máquina expendedora de golosinas resulta:

S = {cero, cinco, diez, quince, veinte}

s_0 = cero

I = {5, 10, botón, chicle, botón, chocolate, dev.}

0 = {click, 2 clicks, chicle, chicle + 5, chocolate, 5, 10, 15, 20}

Las funciones F y G pueden expresarse en forma tabular como se muestra en las Figs. 8 y 9.

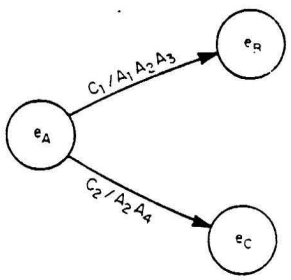

Fig. 7 — Representación de un programa mediante diagramas de estado transición.

F

	5	10	bot. chicle	bot. choc.	dev.
cero	cinco	diez			
cinco	diez	quince			cero
diez	quince	veinte			cero
quince	veinte		cero		cero
veinte			cero	cero	cero

Fig. 8 — Función F.

G

	5	10	bot. chicle	bot. choc.	dev.
cero	click	2 clicks			
cinco	click	2 clicks			5
diez	click	2 clicks			10
quince	click		chicle		15
veinte			chicle+5	chocolate	20

Fig. 9 — Función G.

Implementación de programas

Es deseable en todo programa separar las funciones de control (qué tareas realizar), de las de procesamiento (las tareas propiamente dichas), puesto que esto redunda en programas mejor estructurados, lo que implica menores tiempos de programación y mantenimiento.

Cuando un programa puede ser convenientemente caracterizado por una máquina de estados finitos, la tabulación de las funciones F y G provee un método de realización directa que permite implementar la separación antedicha.

Retomando el ejemplo de la máquina expendedora, supóngase que ésta va a estar controlada por un microprocesador debiendo escribirse el programa correspondiente. Para basar la operación del programa en el diagrama de estado que especifica el funcionamiento de la máquina, dicho diagrama deberá ser codificado en alguna forma aceptable.

Para ello se comienza numerando los conjuntos S, I, y 0:

S = {cero (1), cinco (2), diez (3), quince (4), veinte (5)}

I = {5 (1), 10 (2), botón chicle (3), botón chocolate (4), dev. (5)}

0 = {click (1), 2 click (2), chicle (3), chicle + 5 (4), chocolate (5), 5 (6), 10 (7), 15 (8), 20 (9)}

F

	1	2	3	4	5
1	2	3	1	1	1
2	3	4	2	2	1
3	4	5	3	3	1
4	5	4	1	4	1
5	5	5	1	1	1

Fig. 10 — Matriz de la función F.

G

	1	2	3	4	5
1	1	2	0	0	0
2	1	2	0	0	6
3	1	2	0	0	7
4	1	0	3	0	8
5	0	0	4	5	9

Fig. 11 — Matriz de la función G.

Una vez hecho esto se reemplaza en la especificación de las funciones F y G, los nombres por los números correspondientes. Con las acciones y transiciones no existentes (casilleros en blanco de las matrices F y G) se procederá de la siguiente manera:

1) Matriz F: $V\, f_{ij}$ = vacío, $f_{ij} = i$
2) Matriz G: $V\, g_{ij}$ = vacío, $g_{ij} = 0$ u otro valor inválido

De la aplicación del procedimiento anterior resultan las matrices de las Figs. 10 y 11.

El algoritmo de control del programa será el siguiente:

```
ESTADO ← 1
"do for ever"
   Leer entrada                          !muestrear sensores
   Codificar entrada
   ACCION ← G (ESTADO, ENTRADA)          !determinar respuesta
   ESTADO ← F (ESTADO, ENTRADA)          !pasa a nuevo estado
   "Case of" ACCION
   1: emitir click
   2: emitir click
      emitir click
   3: entregar (chicle)
   4: entregar (chicle)
      devolver (5)
   5: entregar (chocolate)
   6: devolver (5)
   7: devolver (10)
   8: devolver (15)
   9: devolver (20)
   "endcase"
"endo"
```

Salvo en lo que hace a las acciones particulares con las que el sistema responde, el algoritmo propuesto es válido para cualquier programa que pueda ser representado mediante un diagrama de estado.

(Continúa en la pág. 793)

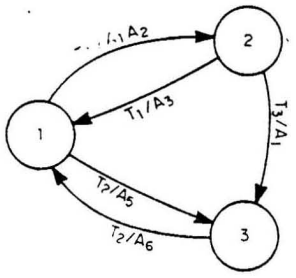

Fig. 12 — Asignación de tareas a los estados del sistema. Diagrama de estados (T_i denota la tecla i).

cargado con la dirección de comienzo de la Subrutina 1.

Esta subrutina 1 está preparada para recibir un dato desde la microcomputadora (debería ser el código ASCII de la letra M-2A en hexadecimal). En caso negativo se vuelve a inicializar la GPIA y comienza otro ciclo.

En caso positivo continúa el programa principal. A continuación se desvincula la GPIA, o sea se la inhabilita para mandar interrupciones.

Luego se inicializa la PIA y se manda a través de ella una señal de borrado hacia el contador universal. Esta señal hace que el instrumento efectúe una medición. Al finalizar la medición manda una señal de escritura ("print"), la que ingresada en la PIA hará que ésta baje la línea IRQ. De esta forma se llega a la Subrutina 2, la cual se encarga de tomar los ocho dígitos de a pares, a través de una señal de reloj ("clock") emitida por la PIA.

Una vez finalizada la subrutina sigue el programa principal con una rutina encargada de acomodar y convertir a código ASCII los dígitos recibidos.

Luego se carga el contenido de los almacenamientos temporarios de "unidad de medida y punto decimal", los cuales contiene información acerca de la posición de las llaves "Tiempo" y "Función" ("TIME" y "FUNCTION") del contador. Con esta información se ingresa a una rutina que se encargará de determinar la posición del punto decimal y la unidad de medida correspondiente.

Al salir de esta rutina se tiene toda la información de la medición a enviar en un cartel ("string") de los 14 bytes ubicados en RAM.

Después se manda un SRQ a través de la GPIA y se controla un bit de un registro de la GPIA, el de estado activo de encuesta serie o SPAS ("Serial poll active state") para verificar si ocurrió la encuesta serie, lo que implica el envío del byte de estado correspondiente. A continuación se vuelve a inicializar la GPIA para que mande TRQ cuando sea direccionada. En ese caso se salta a la subrutina 3, la cual se encarga de enviar los datos de la medición al controlador.

Se vuelve al inicio del programa.

Conclusiones

La interfase fue construida en el Laboratorio de Instrumental de la Facultad de Ingeniería (UBA) y luego fue utilizada con una microcomputadora PET Commodore y con un sistema controlador con microprocesador 8080A. Se desarrolaron programas de envío y recepción de datos ("Source" y "Aceptor Handshake") para que la microcomputadora, como controlador, se comunique con dispositivos que están interconectados a través del bus IEEE 488/78. Además se desarrolló un programa que implementa la función de interfase "Requerimiento de Servicio" ("Service Request"), ya que dicha PC no cuenta con esta facilidad.

Actualmente la interfase se utiliza en la Cátedra de Instrumentos y Mediciones Electrónicas.

Agradecimientos

El autor agradece la colaboración en el proyecto y construcción de la interfase de la Srta. Susana Boloqui, del Ing. Julio Schuchner, del Sr. Massa y del Sr. Orellana. ■

Referencias

[1] Motorola: "Microprocessors Data Manual", 1981.

[2] Texas Instrument Incorporated: "TMS 9914 GPIB Adapter Preliminary Data Manual", 1979.

[3] Texas Instrument Incorporated: "ALS/AS Logic Circuits Data Book", 1983.

[4] Beckman: "Models 6360, 6370, 6380. Operating and Servicing Manual", Electronic Instrument Division.

[5] Beckman: "Models 673 and 675, Function Modules. Operating and Servicing Manual", Electronic Instruments Division.

[6] SMP S.A.: "MK 80 Manual del Usuario", 1979.

[7] Fisher, E. y Jensen, C. W.: "PET and the IEEE 488 Bus (GPIB)". Osborne/McGraw-Hill, Berkeley, California, 1980.

[8] Osborne, A. y Donahue, C. S.: "PET/CBM Personal Computer Guide". Osborne/McGraw-Hill, Berkeley, California, 1980.

[9] The Institute of Electrical and Electronics Engineers Inc.: "IEEE Standard Digital Interface for Programmable Instrumentation", IEEE Instrumentation and Measurements Group, 1978.

Método para la construcción de programas basados...

(de la pág. 779)

En el ejemplo visto, todas las tareas útiles del sistema se encuentran asociadas a transiciones, pero bien pudiera ser que en otros sistemas, fuera necesario asignar tareas a los estados del sistema.

Consideremos el caso de una consola "inteligente", que cuenta con teclas luminosas debiendo producirse el refresco de las mismas por programa ya que éstas no están "latcheadas". Se llama "inteligente" a una consola que mediante la iluminación selectiva de las teclas va guiando al usuario en la operación del sistema. Tal situación es convenientemente resuelta asociando a cada estado, la excitación de las teclas que constituyen un estímulo válido para ese estado (Fig. 12).

El algoritmo de excitación será el siguiente:

```
"case of" ESTADO                    !procedimiento
                                     de excitación
  1: encender (T1)
     encender (T2)
  2: encender (T1)
     encender (T3)
  3: encender (T2)
"endcase"
```

Conclusiones

La naturaleza gráfica y facilidad de comprensión, así como la existencia de algoritmos de implementación directa, hacen de los diagramas de estado una herramienta atractiva para la construcción de programas, no obstante, debe recordarse que no todo programa puede ser adecuadamente modelizado mediante la utilización de esta técnica. ■

Referencias

[1] Landau, J. V.: "Hardware Oriented State Description Techniques". Microprocessor Applications Handbook McGraw Hill.

[2] Valette, R. y Courvouser, M.: "Systemes de Commande en Temps Real". SCM.

[3] Wulf y col.: "Fundamental Structures of Computer Science". Addison Wesley.