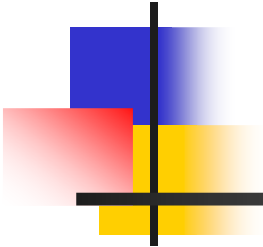




Técnicas Digitales II

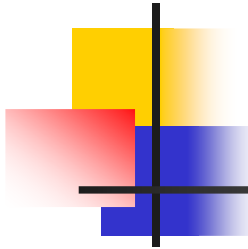
Máquina de Estado



Ing. Juan Manuel Cruz (jmcruz@hasar.com)
Profesor Asociado Ordinario

Ing. Alex Lozano (alex.tdii@gmail.com)
Jefe de Trabajos Prácticos Ordinario

Buenos Aires, 8 de Mayo de 2010



Temario

- Introducción
- Estado del arte
- Problemática general
- Criterios de diseño
- Casos típicos de estudio
- Un ejemplo de aplicación
- Ejercicios propuestos



Máquina de Estado (palotes ++)

- Máquina de Estado (State Machine)
 - Modelo matemático de un sistema con entradas y salidas discretas, que posee sintaxis y semántica formales
 - Útiles para representar aspectos dinámicos no representables por diagramas de otro tipo



¿Como qué se las conoce?

- Existen múltiples terminologías:
 - Statechart
 - FSM & EFSM (Finite State Machine & Extended FSM)
 - PFSM & CFSM (Partial & Complete FSM)
 - TFSM (Timed FSM)
 - DFMS & NFSM (Deterministic & Nondeterministic FSM)
 - LTS (Labelled Transition System)
 - Automata (teoría de lenguajes)
 - Timed Automata
 - IOA (Input Output Automata)
 - Etc., etc., etc.



¿Qué son?

- Modelo de Comportamiento (Behavior), denotan evolución mediante:
 - Estados (comportamiento estático)
 - Transiciones (evolución entre estados)
 - Excitaciones (que condicionan las transiciones)
 - Acciones (asociadas a las transiciones)
- Determinísticas y no determinísticas
- Composición en paralelo de múltiples máquinas



¿Para qué sirven?

- Especificar aspectos formales relacionados con:
 - Sistemas de Control en Tiempo Real
 - Dominios Reactivos o Autónomos
 - Computación Reactiva
 - Protocolos
 - Circuitos Eléctricos/Electrónicos (Lógica Secuencial)
 - Arquitectura de Software
 - Análisis Lexicográfico, Gramatical
 - Etc., etc., etc.



¿Por qué usarlas?

- Rigurosa descripción del comportamiento de la solución al problema planteado
- Soluciones muy sofisticadas y ridículamente sencillas. Incluso compuestas por la operación en paralelo de múltiples máquinas
- Código más simple, eficiente y preciso. Más fácil de depurar, modificar, expandir y mejor organizado
- Facilitan el análisis, el diseño y la implementación



¿Cómo se formalizan?

- Mediante diversos métodos de representación:
 - Funciones matemática
 - Diagramas / Tablas de Estados y Transiciones
 - Redes de Petri
- Existen métodos para minimizar estados
- Mediante diversos lenguajes de programación:
 - Assembly, C p/micros o PCs
 - C++, Java, UML
 - VHDL, etc., etc., etc.



Casos típicos de estudio

- Surgen de combinar los siguientes factores:
 - Excitadas por Eventos
 - Excitadas por Sincronismos (time-out, timer-tick)
 - Excitadas por Semáforos/Variables/Colas (mensajes)
 - Interacción con Entradas/Salidas
 - Composición en paralelo de múltiples máquinas
- Algunas máquinas con nombre propio:
 - Moore & Mealy
 - Turing & Markov

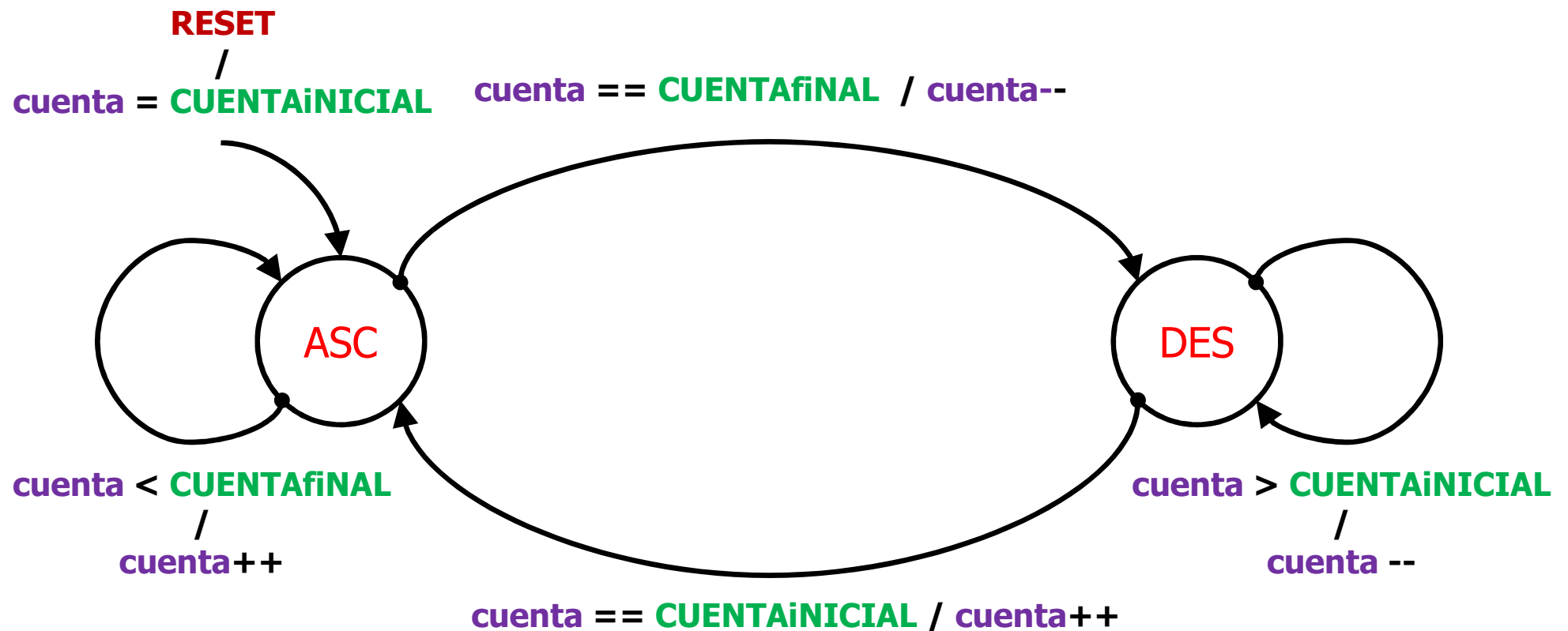


Un ejemplo de aplicación: Contador

- Contador Ascendente / Descendente
 - Cuenta ascendente desde CUENTAiNICIAL → hasta CUENTAfINAL
 - Cuenta descendente desde CUENTAfINAL → hasta CUENTAiNICIAL
 - Itera
- Recursos
 - Variables: estado , cuenta
 - Defines: ASCENDENTE, DESCENDENTE,
ESTADOmAXIMO, ESTADOiNICIAL,
CUENTAiNICIAL, CUENTAfINAL

Contador

Diagrama de Estados y transiciones



Contador

Tabla de Estados y Transiciones

Estado Actual	Excitación	Estado Futuro	Acción
X	RESET estado >= ESTADOmAXIMO	ESTADOiNICIAL	cuenta = CUENTAiNICIAL
ASCENDENTE	estado < ESTADOmAXIMO && cuenta < CUENTAfINAL	ASCENDENTE	cuenta ++
	estado < ESTADOmAXIMO && cuenta == CUENTAfINAL	DESCENDENTE	cuenta --
DESCENDENTE	estado < ESTADOmAXIMO && cuenta > CUENTAiNICIAL	DESCENDENTE	cuenta --
	estado < ESTADOmAXIMO && cuenta == CUENTAiNICIAL	ASCENDENTE	cuenta ++



Contador

Implementaciones posibles

- Implementación mediante:
 - Switch (estado)
 - Arrays de punteros a función (estado)
 - Múltiples if (estado)
 - Desarrolladas en esta presentación
 - Tablas de excitaciones & acciones (estado, excitación)
 - Se escucharán propuestas en esta presentación
 - Se recomienda compararlas en la práctica de laboratorio



Contador

Implementación con switch (1 de 3)

```
/*  Inclusión de Archivos  */
#include <Macros51.h>      // Incluyo Archivo con Macros de Uso General
#include <Defines51.h>     // Incluyo Archivo con defines para 8051

/*  Declaración de Prototipos  */
void InicializarContador (void);
void Contador (void);

/*  Defines del correspondientes al Contador  */

#define ASCENDENTE        0           // Estados
#define DESCENDENTE       1
#define ESTADOMAX         DESCENDENTE + 1

#define ESTADOiINICIAL    ASCENDENTE

#define CUENTAiINICIAL    0x00        // Límites de Cuenta
#define CUENTAfINAL       0xFF
```



Contador

Implementación con switch (2 de 3)

```
/*  Reserva de Variables    */  // Reservas en RAM Interna de acceso directo
```

```
unsigned char data estado;  // Estado de Contador de 8 bits Asc. / Desc.
```

```
unsigned char data cuenta;  // Contador de 8 bits Asc. / Desc.
```

```
/*  Programa    */
```

```
void InicializarContador (void)
```

```
{  
    estado = ESTADOiINICIAL;  
    cuenta = CUENTAiINICIAL;
```

```
    // Inicializa las Variables y Salidas de Control
```

```
    return;  
}
```



Contador

Implementación con switch (3 de 3)

```
void Contador (void)
{
    // Si el Estado esta fuera de rango
    reinicializa la M de E y retorna

    if (estado >= ESTADOmAX) {
        InicializarContador();

        return;
    }

    switch (estado) { // En función del
        Estado y la Excitación Actualiza las
        Variables, las                               Salidas de Control
        y retorna

        case ASCENDENTE:

            if (cuenta < CUENTAfinal)
                cuenta++;
```

```
        esle {
            estado = DESCENDENTE;
            cuenta--;
        }
        break;

        case DESCENDENTE:

            if (cuenta > CUENTAinICIAL)
                cuenta--;

            else {
                estado = ASCENDENTE;
                cuenta++;
            }
            break;
    }
}
```




Contador

Implementación c/punteros a función (1 de 2)

```
/* Declaración de Prototipos */

void Ascendiendo (void);
void Descendiendo (void);
// Arreglo de Punteros a las Funciones para cada Estado

code void (code *ArrayFuncionesEstadoContador []) (void) = { Ascendiendo, \
                                                                Descendiendo};

/* Programa */

void Contador (void)
{
    // Si el Estado esta fuera de rango reinicializa la M de E y retorna

    if (estado >= ESTADOMAX) {
        InicializarContador();
        return;
    }

    (*ArrayFuncionesEstadoContador [estado]) ();
}
```



Contador

Implementación c/punteros a función (2 de 2)

```
void Ascendiendo(void)
{
    if (cuenta < CUENTAfinal)
        cuenta++;

    else {
        estado = DESCENDENTE;
        cuenta--;
    }

    return;
}
```

```
void Descendiendo (void)
{
    if (cuenta > CUENTAinICIAL)
        cuenta--;

    else {
        estado = ASCENDENTE;
        cuenta++;
    }

    return;
}
```



Contador

Implementación con múltiples if

```
/* Programa */

void Contador (void)
{ // Si el Estado esta fuera de rango
  reinicializa la M de E y retorna

    if (estado >= ESTADOMAX) {
        InicializarContador();
        return;
    }

    if (estado == ASCENDENTE) {
        if (cuenta < CUENTAfinal)
            cuenta++;

    else {
        estado = DESCENDENTE;
        cuenta--;
```

```
    }

    return;
}

// (estado == DESCENDENTE)
else {

    if (cuenta > CUENTAinICIAL)
        cuenta--;
    else {
        estado = ASCENDENTE;
        cuenta++;
    }
}
}
```



Contador

Comparación de Implementaciones

Module Information	Switch	Punteros a función	Multiples if
Code size	59	77	55
Contant size	-	4	-
XDATA size	-	-	-
PDATA size	-	-	-
DATA size	2	2	2
IDATA size	-	-	-
BIT size	-	-	-



Ejercicios propuestos

- Detector de Secuencia
 - Seguimiento de señales
- Generador de Secuencia
 - Señales luminosas
- Control de Acceso con/sin cupo
- Ascensor de 2 plantas
- Escalera mecánica unidireccional/bidireccional
- Puerta corrediza/Portón levadizo
- Máquinas expendedoras o de autoservicio
- Maquinaria o procesos industriales



Referencias

- Ingeniería de Software I, DC - FCEyN – UBA
- Software de Tiempo Real - FRBB – UTN
- Sintaxis y Semántica del Lenguaje - FRT - UTN
- Finite State Machines: Making simple work of complex functions
- Ingeniería en Controladores - Máquinas de Estado
- An Introduction to Finite State Machines
- JOURNAL OF OBJECT TECHNOLOGY: A Typing Scheme for Behavioural Models
- Teoría de Autómatas - Guillermo Morales-Luna
- Máquinas de Estados Finitos -Jorge Alejandro Gutiérrez Orozco
- Finite State Machines - James Grimbleby
- Grupo de Inteligencia Artificial y Robótica - FRBA - UTN



Referencias

- **R3CT18:** Manejo de Cola en C de 8051
- **R3CT19:** Máquina de Estado (Contador de 8 bits) en C de 8051
- **R3CT20:** Máquina de Estados (Tanque) en C de 8051
 - Autor: Ing. Juan Manuel Cruz
 - Publicados por CEIT – Electrónica – FRBA – UTN (2002)
- **R3CT23:** Teclado Matricial en C de 8051
- **R3CT24:** Display Multiplexado en C de 8051
 - Autor: Ing. Juan Manuel Cruz
 - Publicados por CEIT – Electrónica – FRBA – UTN (2003)
- **R3CT25:** Comunicación Serie en C de 8051
 - Autor: Ing. Juan Manuel Cruz
 - Publicado por CEIT – Electrónica – FRBA – UTN (2004)