Ciencias de la Computación Aplicadas a la Solución de Problemas CCPG1007

Federico Domínguez, PhD.

Unidad 5 – Sesión 6: Máquinas de estado finito

Finite State Machines (FSM) o máquinas de estado finito es un patrón de diseño ideal para procesos pequeños en programación de sistemas y sistemas embebidos.

También conocido como un autómata finito, es un modelo conceptual de una máquina que puede estar en un número finito de estados, un estado a la vez.

En la programación de sistemas tienen aplicaciones en sistemas embebidos y en diseño de lenguajes y compiladores.

Una FSM es definida por sus estados, su estado inicial y las condiciones para la transición entre estados. Esto usualmente se representa con un grafo dirigido.

Es ideal para programar un comportamiento simple que ejecuta una serie de acciones de acuerdo a entradas al sistema.

Una FSM es necesaria cuando detectamos que el código se llena de condicionales (if – else).

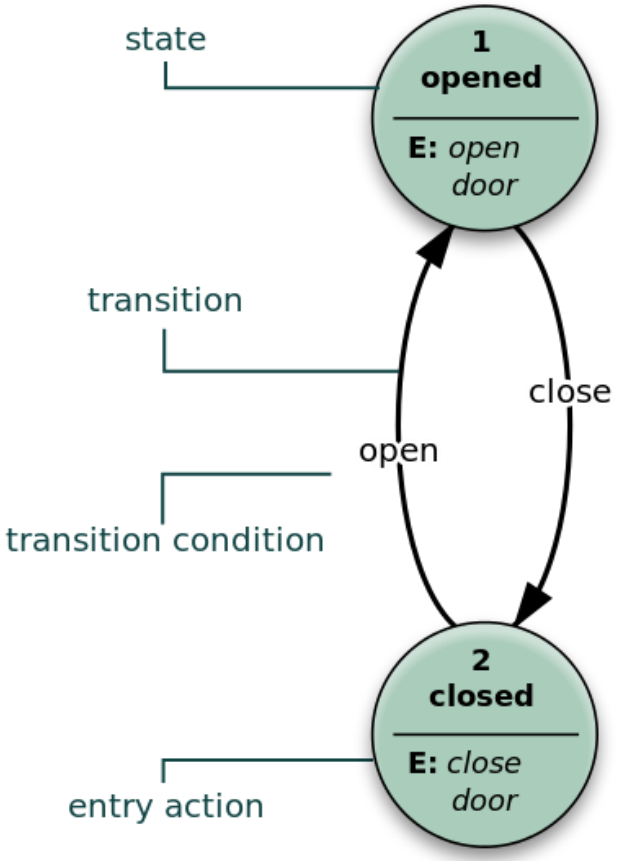
Es ideal para simplificar código con un número creciente de condicionales, son reemplazados por un *switch*.

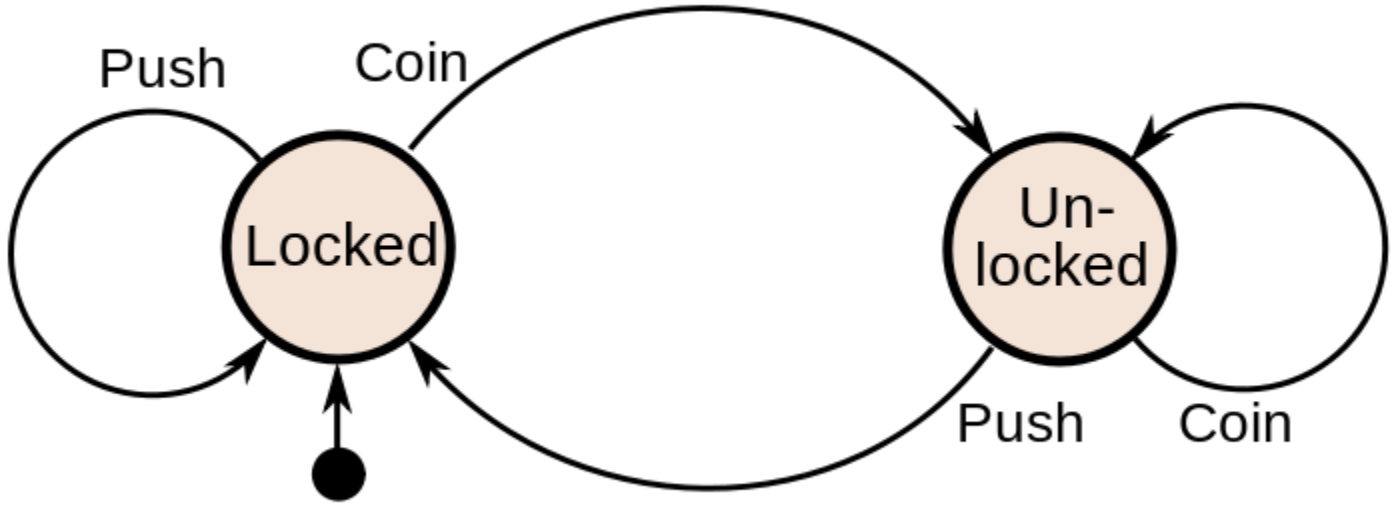
El patrón de diseño FSM es muy común, algunos lenguajes de programación incluyen librerías que simplifican su programación.

En lenguajes orientados a objetos, como C++, existe una versión más robusta del patrón FSM llamada patrón Estado.

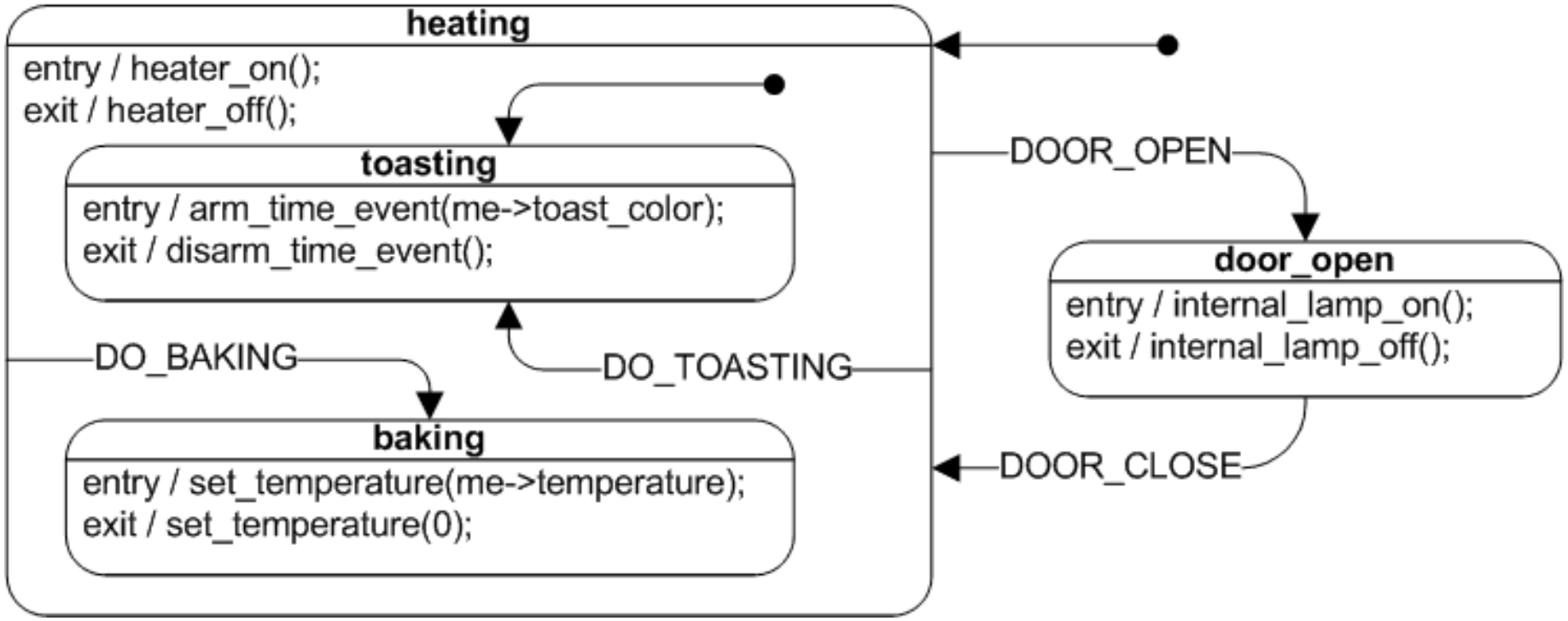
El patrón FSM no es muy escalable y es preferible no usarlo en sistemas complejos donde se tendría un número considerable de estados.

Las FSM son usualmente documentadas usando grafos dirigidos (también conocidos como diagramas de estado).





UML provee un estándar para representar diagramas de estado.



Definir estados para la práctica

En C, las FSM se pueden implementar combinando enumeraciones con un *switch*.

Las enumeración es un tipo de datos usado para declarar constantes. Todos los estados posibles son especificados en la enumeración y se declara una variable global con el tipo de datos de la enumeración.

```
enum Estado {EXIT = 0, ESPERANDO_NOMBRE, ARCHIVO_INVALIDO, ENVIANDO_ARCHIVO} estado;
```

La variable global es inicializada con el estado inicial y luego usada en un *switch* dentro de un lazo para implementar la FSM y sus transiciones.

```
estado = ESPERANDO_NOMBRE;
while(estado) {
        switch(estado)
        {
                case ESPERANDO_NOMBRE:
                        /* IMPLEMENTAR AQUÍ CODIGO DE ESTE ESTADO */
                        break;
                case ARCHIVO_INVALIDO:
                        /* IMPLEMENTAR AQUÍ CODIGO DE ESTE ESTADO */
                        estado = EXIT;
                        break;
```