VORTEX

EMBEDDED MAKERS

SASE 2017 SIMPOSIO ARGENTINO DE SISTEMAS EMBEBIDOS

Máquinas de Estados UML (Statecharts)

Ing. Leandro Francucci (lf@vortexmakes.com)

10 de Agosto 2017

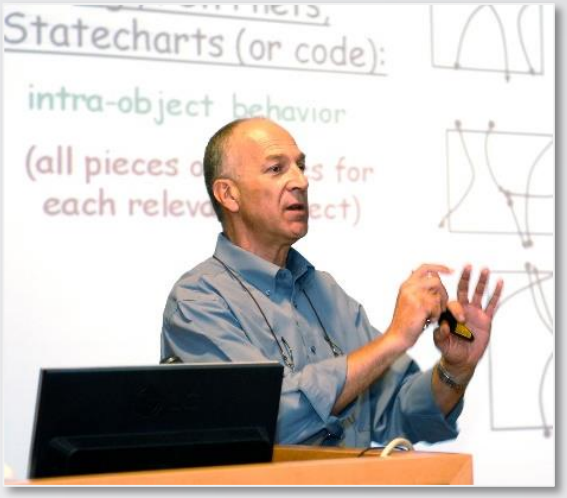
Máquina de estados

- Las máquinas de estados (SM) pueden utilizarse para modelar el comportamiento *reactivo* de un sistema.
- Representado como el recorrido de un grafo de vértices interconectados por uno o más arcos, que representan las *transiciones*, las cuales se accionan como consecuencia del *despacho* de sucesivas ocurrencias de *eventos*, que *coinciden* con los disparadores activos de dicho grafo.
- Durante este recorrido, la SM puede ejecutar una secuencia de *comportamientos* asociados con los diferentes *elementos* que la conforman.
- Una SM se compone de una o más *regiones* independientes, las que a su vez poseen sus propios elementos (*vértices* y transiciones).
- Por su naturaleza orientada a eventos, la ejecución de una SM posee dos estados: *en tránsito* o *en estado*.

Statecharts

- Constituye un **formalismo visual** para describir los estados y transiciones de manera modular, permitiendo la agrupación y concurrencia, fomentando la capacidad de moverse fácilmente entre diferentes niveles de abstracción.
- Respecto de las tradicionales FSM, básicamente incorpora:
  - Anidamiento jerárquico de estados
  - Concurrencia
  - Transiciones condicionadas
  - Acciones de entrada y salida de estados
  - Actividades
  - Pseudoestados

- La notación y semántica fue definida por *Dr. David Harel*, luego adoptada por UML.
- Surgió como una manera *clara* y *precisa* para describir el comportamiento dinámico de un sistema de vuelo de un avión de combate Israelí, específicamente IAI Lavi.

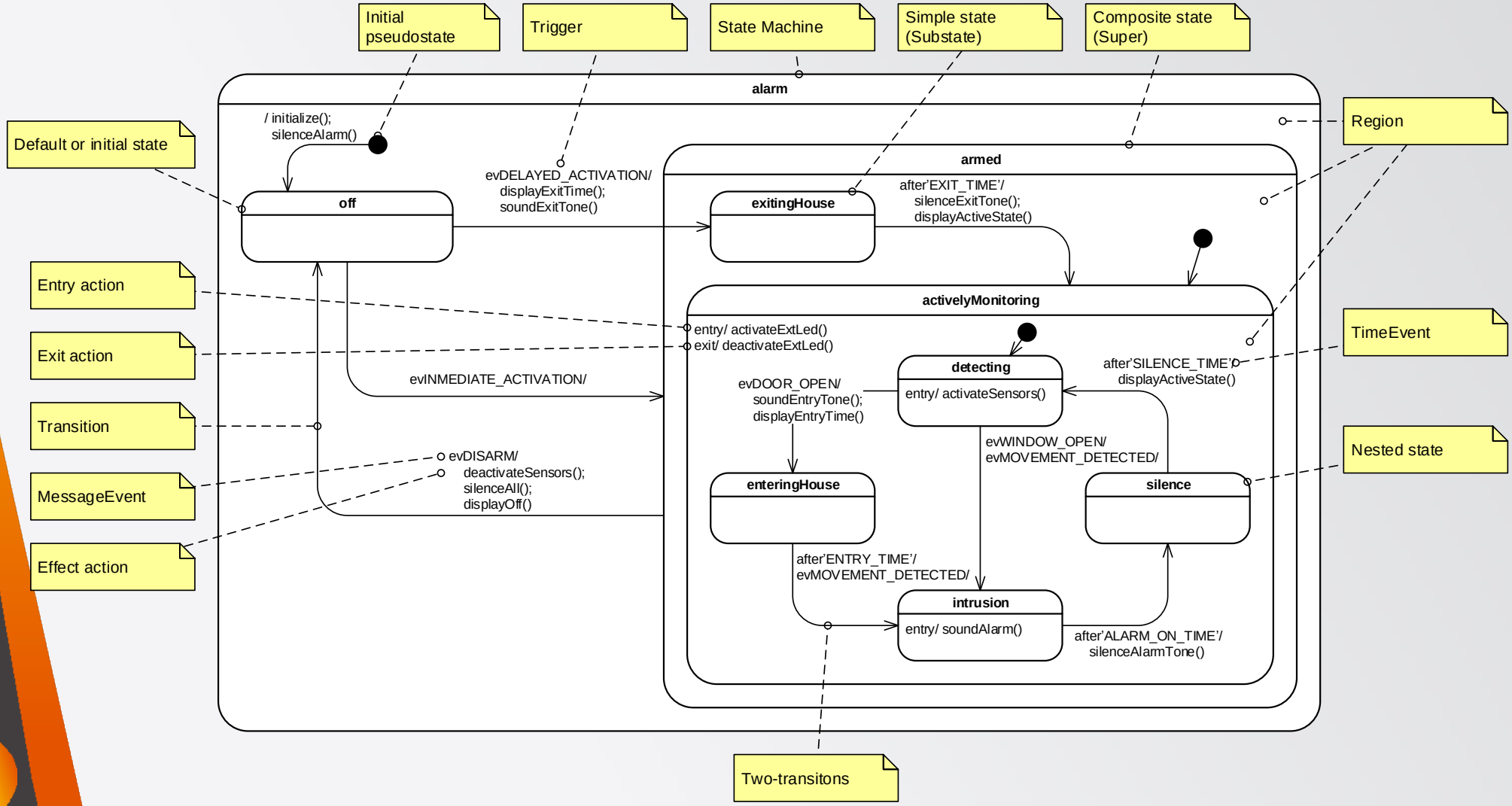


Prof. David Harel

Máquina de estados
Elementos del modelo

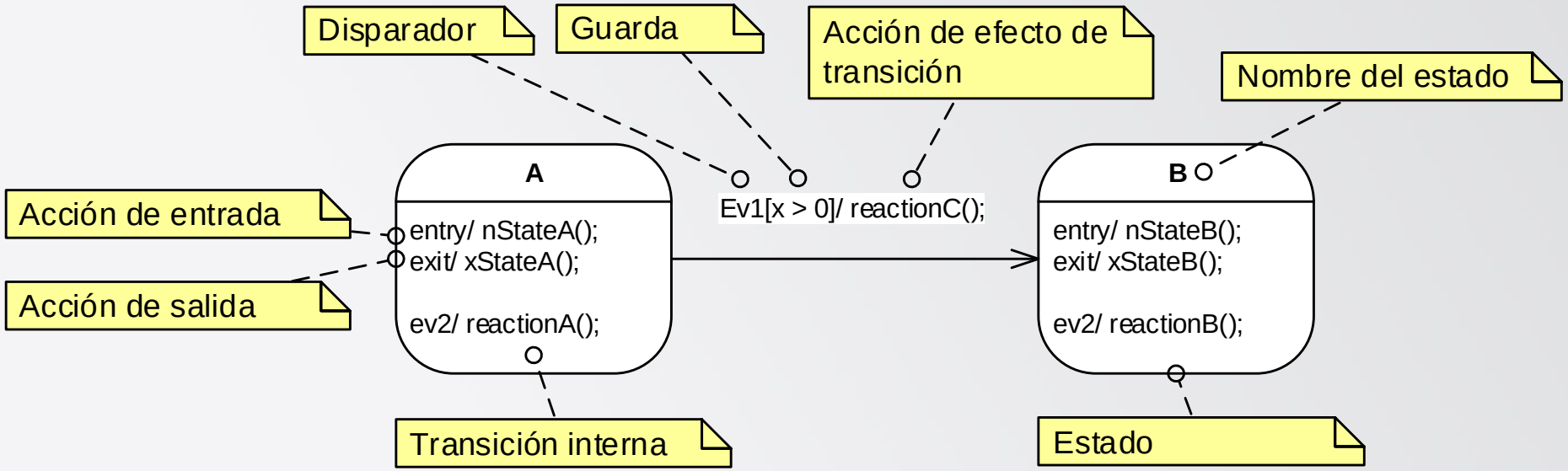
- Máquina de estados
- Región
- Vértice
  - Estado: Simple, Compuesto, Submáquina
  - Referencia de punto de conexión
  - Estado Final
  - Pseudoestado: Initial, DeepHistory, ShallowHistory, Join, Fork, Junction, Choice, EntryPoint, ExitPoint, Terminate
- Transición
- Evento
  - Message Event: SignalEvent, CallEvent
  - TimeEvent
  - ChangeEvent

Máquina de estados
Ejemplo básico



Máquina de estados
Elementos básicos

Estados simples, transiciones, disparadores, guardas y acciones



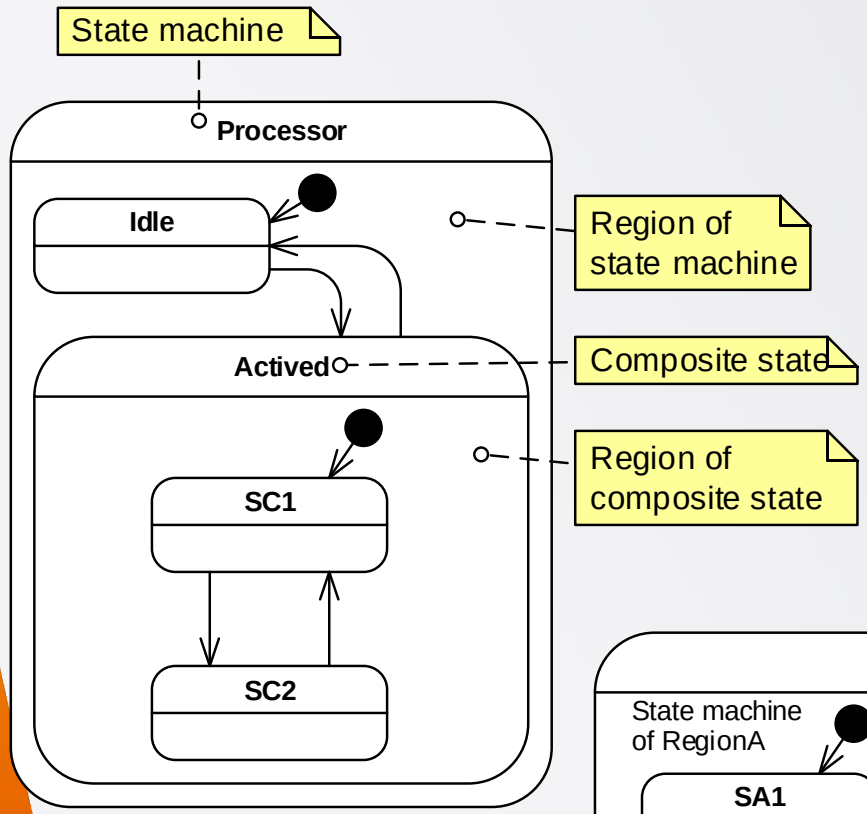
Notación de la transición

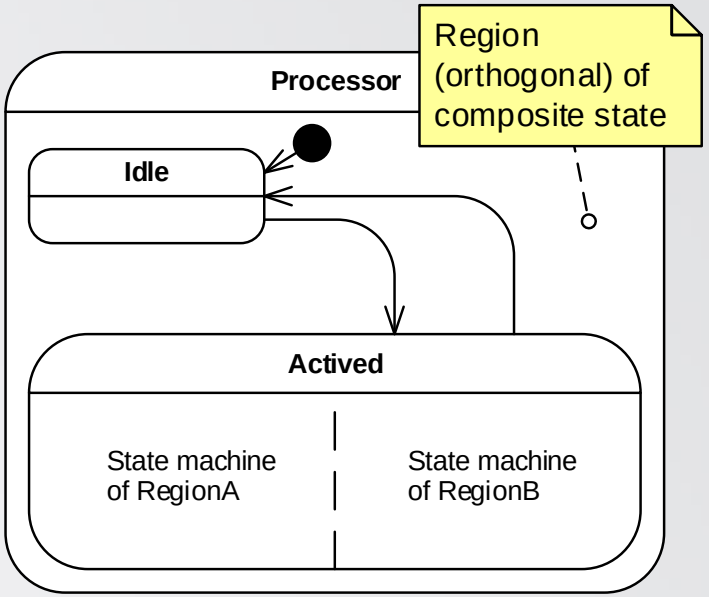
event-name'('parameter-list')''['guard-expression']''/'action-list

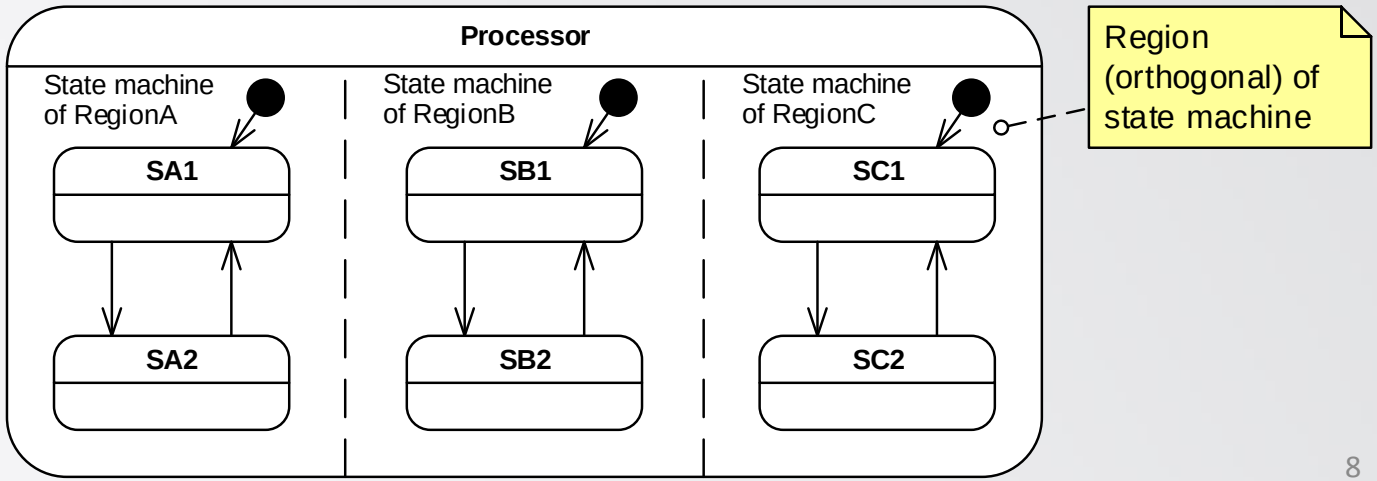
- Nombre del evento que dispara la transición. *Señal del evento.*
- Lista separada por comas, que contiene los nombre de los parámetros pasados con la señal del evento
- Expresión lógica, la cual debe ser verdadera para tomar la transición
- Lista de operaciones ejecutadas como resultado de la transición tomada

- Ante un evento recibido, la máquina recorre los estados, ascendiendo en jerarquía, desde el estado más interno hasta encontrar la transición que dispara, caso contrario se descarta.
- Algunos ejemplos de transiciones permitidas:
  1) E1
  2) E1[x < my_sensor >value]
  3) E1(x: float) / y = filter(x * 100);
  4) E1(x: float, y: long) [abs(x) > abs(y ^ 2) > 0] / result = x / y;
  5) E1 / y = sqrt(x^2 + y^2); z = 2*y – 3;

Región


State machine
Processor
Idle
Actived
SC1
SC2
Region of state machine
Composite state
Region of composite state


Processor
Region (orthogonal) of composite state
Idle
Actived
State machine of RegionA
State machine of RegionB


Processor
State machine of RegionA
State machine of RegionB
State machine of RegionC
SA1
SA2
SB1
SB2
SC1
SC2
Region (orthogonal) of state machine

Región

- Es una parte de *alto nivel*, de una SM o un estado *compuesto*, que contiene vértices y transiciones. Al igual que las SM, los estados compuestos pueden contener más de una región.
- Representa un fragmento de comportamiento, que puede ejecutarse simultáneamente con otras regiones del mismo nivel.
- Dos o más regiones son ortogonales, si pertenecen al mismo estado o al nivel más alto de una SM.
- Pueden tener su propio *pseudoestado initial* y *EstadoFinal*.
- Puede activarse de forma implícita (por defecto) o explícita.
  - Activación por defecto: la ejecución comienza con la transición emergente de su pseudoestado initial (si está definido). En SM y estados compuestos.
  - Activación explícita: se ingresa a la región por una transición que termina en uno de sus vértices internos.

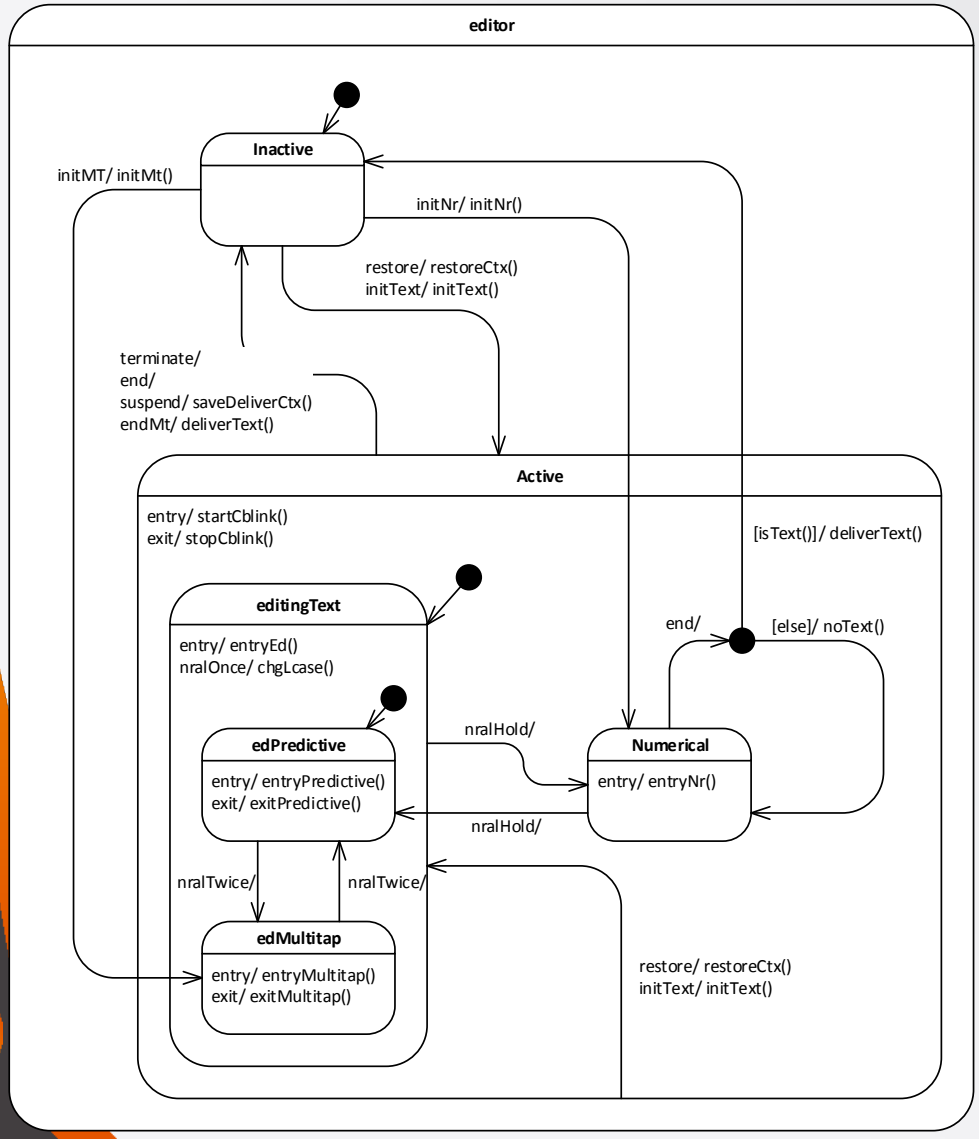
Vértice

- Es una abstracción que captura las características comunes de un conjunto de diferentes tipos de nodos: estado, pseudoestado, referencia de punto de conexión y estado final.
- Con algunas excepciones, puede ser el origen y/o el destino de cualquier número de transiciones.
- Pueden ser *transitivos* o *estables*.
- Su semántica depende del tipo de nodo que representa.

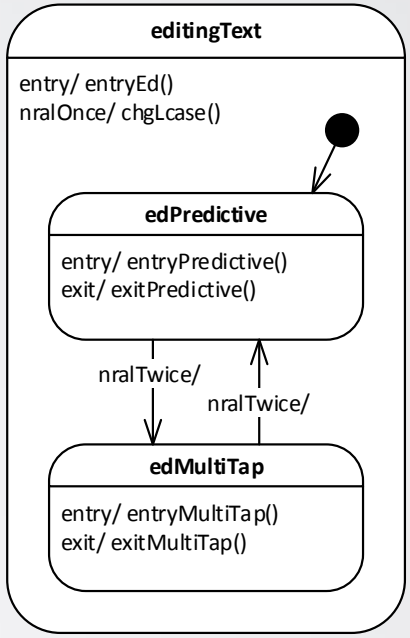
Estado

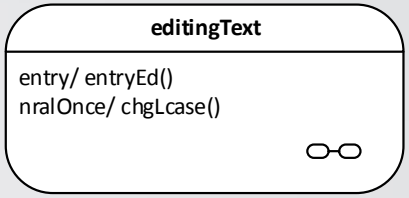
- Modela una situación, en la ejecución de la SM, durante la cual, una condición se mantiene invariante.
- **Simple**: no posee internamente ni vértices ni transiciones.
- **Compuesto**:
  - Simple: contiene una única región.
  - Ortogonal: con más de una región.
  - Todo estado contenido en una región de un estado compuesto es un *subestado* del mismo. Este puede ser:
    - Directo
    - Indirecto
- **Submáquina**: se refiere a una SM completa, la cual se considera estar conceptualmente “anidada” dentro del estado.

Estado
Ejemplo estado compuesto simple



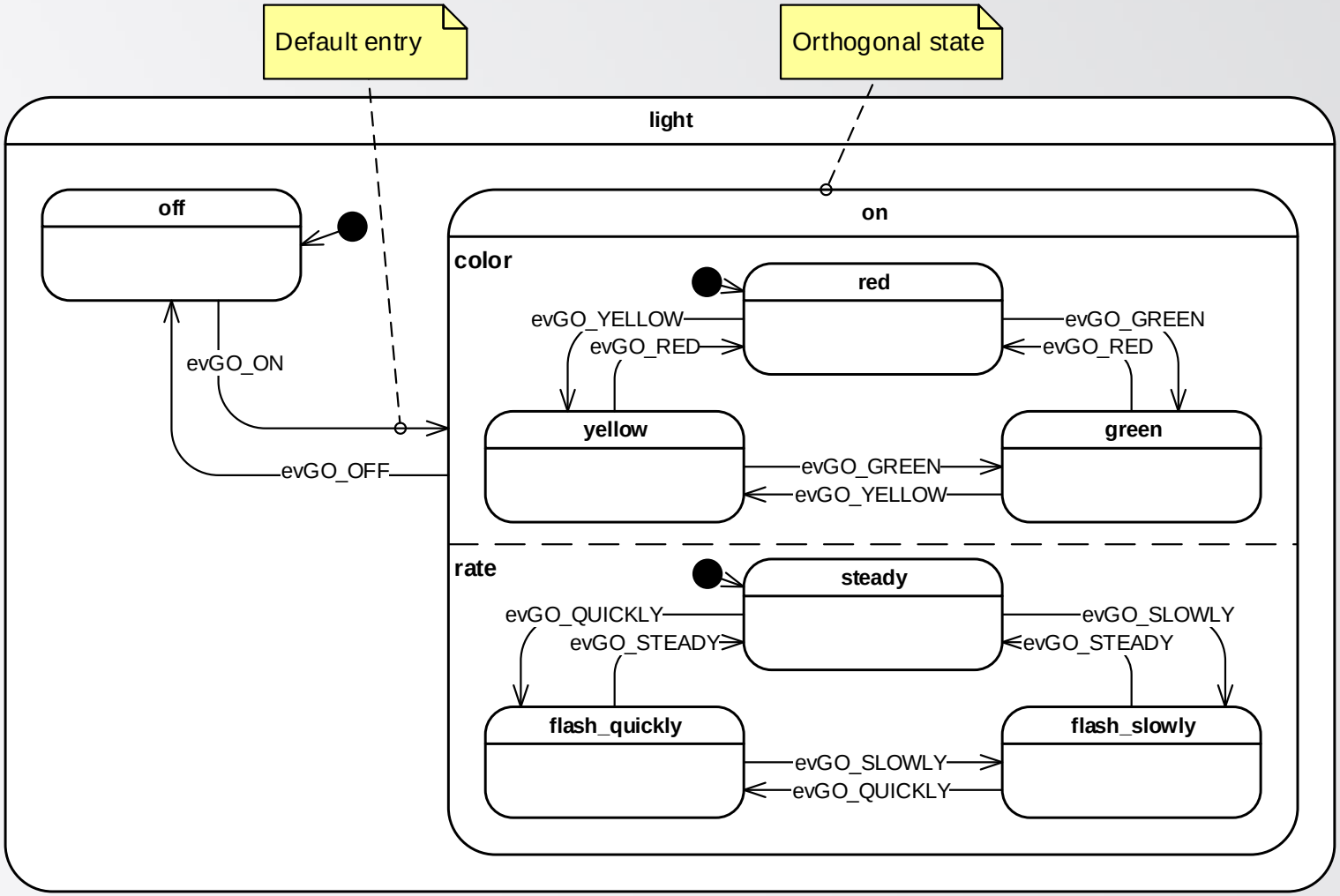
- `editor`: SM con una única región
- `Active`: estado compuesto simple
- `Numerical`: subestado directo de Active
- `edPredictive`: subestado indirecto de Active



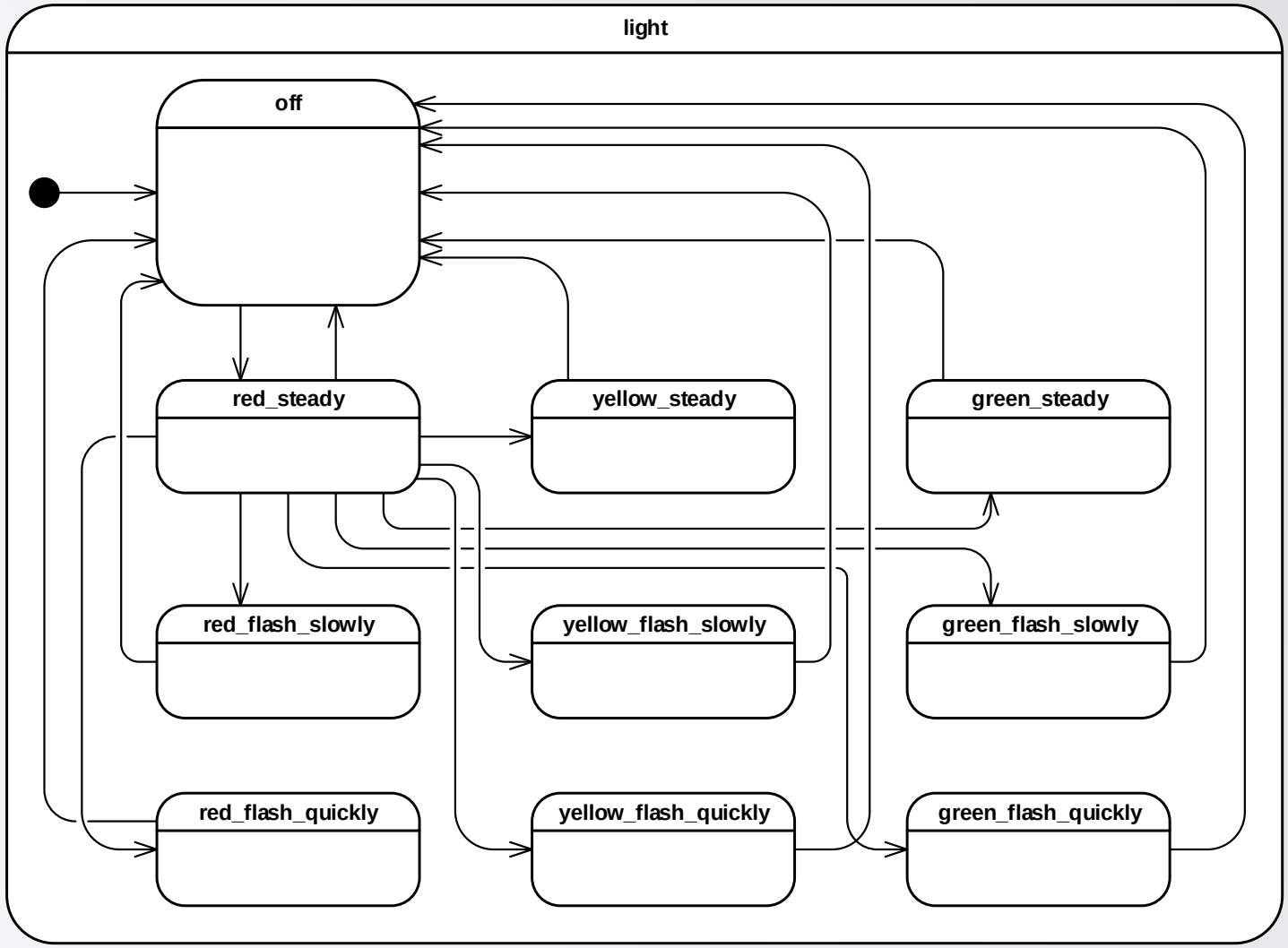


Estado

Ejemplo estado ortogonal



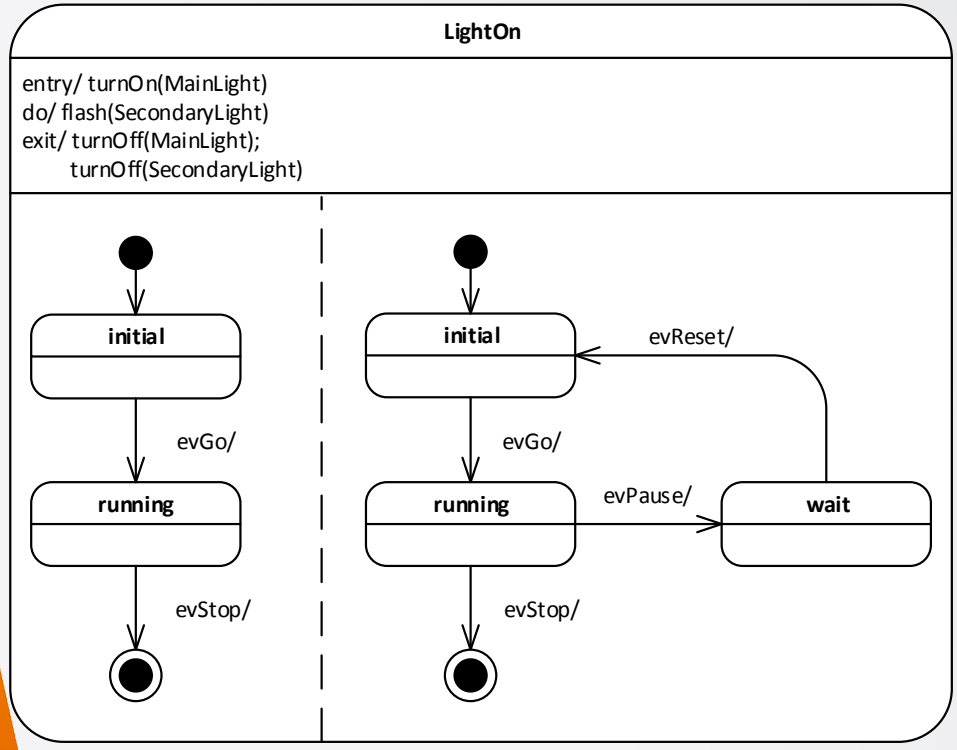
Estado
Equivalencia estado ortogonal

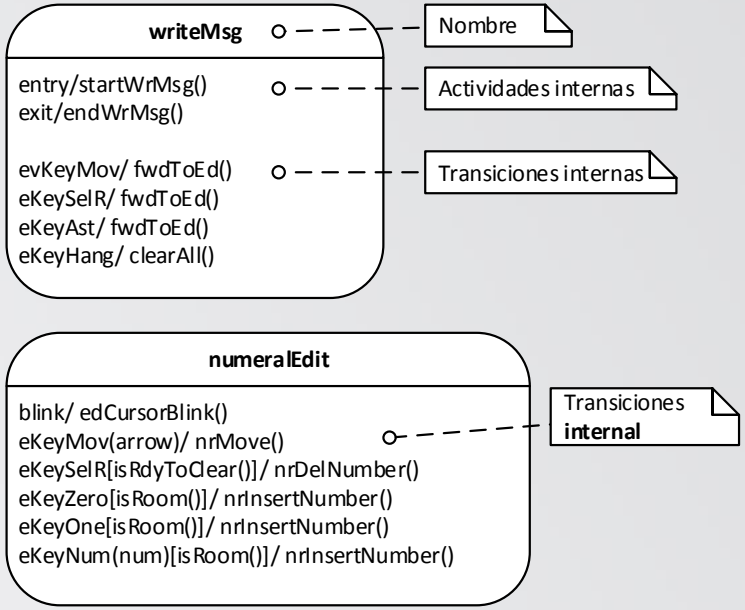


Estado
Configuración de estados

- El “estado” actual de una SM en ejecución, se representa por una o más jerarquías de estados, conocida como *configuración de estados*.
- Una SM en ejecución puede estar exactamente en una configuración de estados a la vez, conocida como *configuración de estados activa*.
- Un estado está *activo* si pertenece a la configuración de estados activa.
- La ejecución de una SM transita de una configuración a otra.
- Es *estable* cuando:
  - no posee transiciones habilitadas,
  - todos sus *comportamientos de entrada (entry),* si existen, finalizaron.

Estado
Comportamientos asociados





- `turnOn`: acción de entrada al estado `LightOn`
- `flash`: actividad
- `turnOff`: acción de salida del estado `LightOn`
- `fwdToEdit`: acción de efecto del estado `WriteMsg` al evento `evKeyMov`

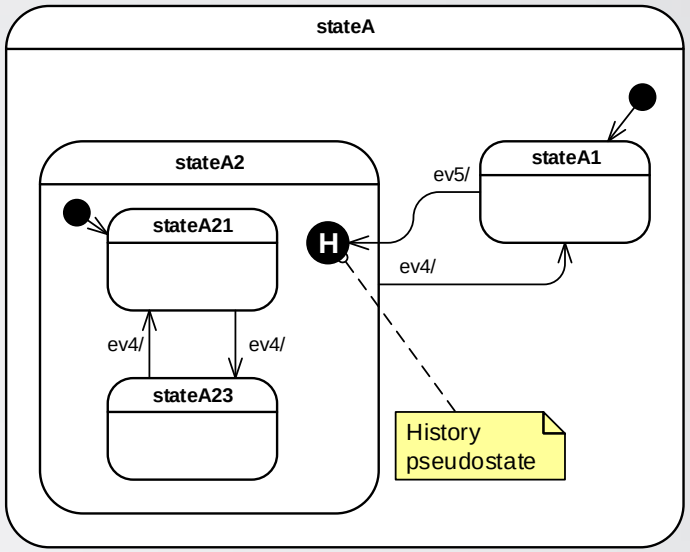
Estado
Comportamientos asociados

- Comportamiento de entrada (**entry**)
  - Si está definido, se ejecuta siempre que se ingrese al estado.
- Comportamiento de salida (**exit**)
  - Si está definido, se ejecuta siempre que se abandona el estado.
- Comportamiento de actividad (doActivity)
  - Si está definido, comienza su ejecución cuando se ingresa al estado, finalizado su comportamiento entry.
  - Se ejecuta concurrentemente con otros comportamientos del mismo estado, hasta que:
    - La actividad termine, en cuyo caso se genera un *evento de terminación*.
    - Se abandone el estado, interrumpiendo así la actividad.
  - También conocido como “do”.

Estado
Historia

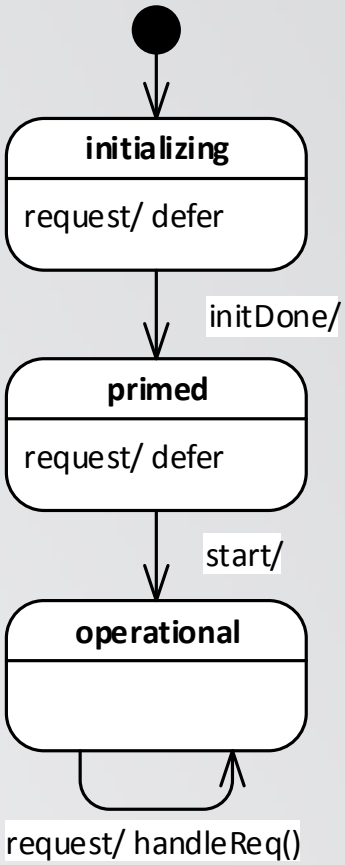
- Este concepto se asocia con las regiones de los estados compuestos.
- Mediante el cual, una región mantiene un registro de la última configuración de estados activa.
- Permite, si se requiere, volver a esa misma configuración de estados:
  - La próxima vez que la región esté activa,
  - O si hay una transición que termina en su historia

- Existen dos tipos de historia:
  - ***Shallow***: mantiene registro sólo del último subestado directo visitado de la región
  - ***Deep***: mantiene registro de la configuración de estados completa de la región



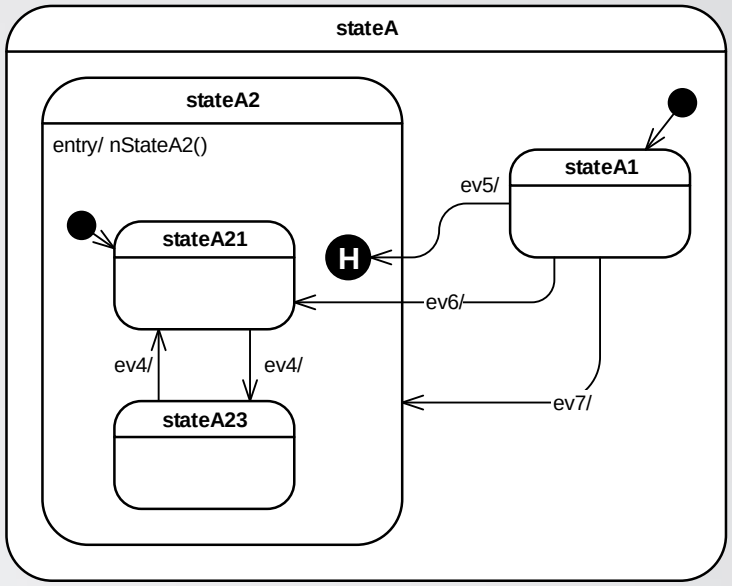
Estado
Eventos diferidos

- Un estado puede especificar un conjunto de tipos de eventos que pueden diferirse en dicho estado.
- Implica que estos no se despacharán mientras se mantenga activo el estado, permaneciendo en el *event pool*, hasta que:
  - se alcance una configuración de estados donde ya no se difieran, o
  - si uno de estos eventos se utiliza explícitamente como disparador.
- En el caso de estados compuestos y submáquina, la condición perdura mientras el estado continúe en la configuración de estados activa.

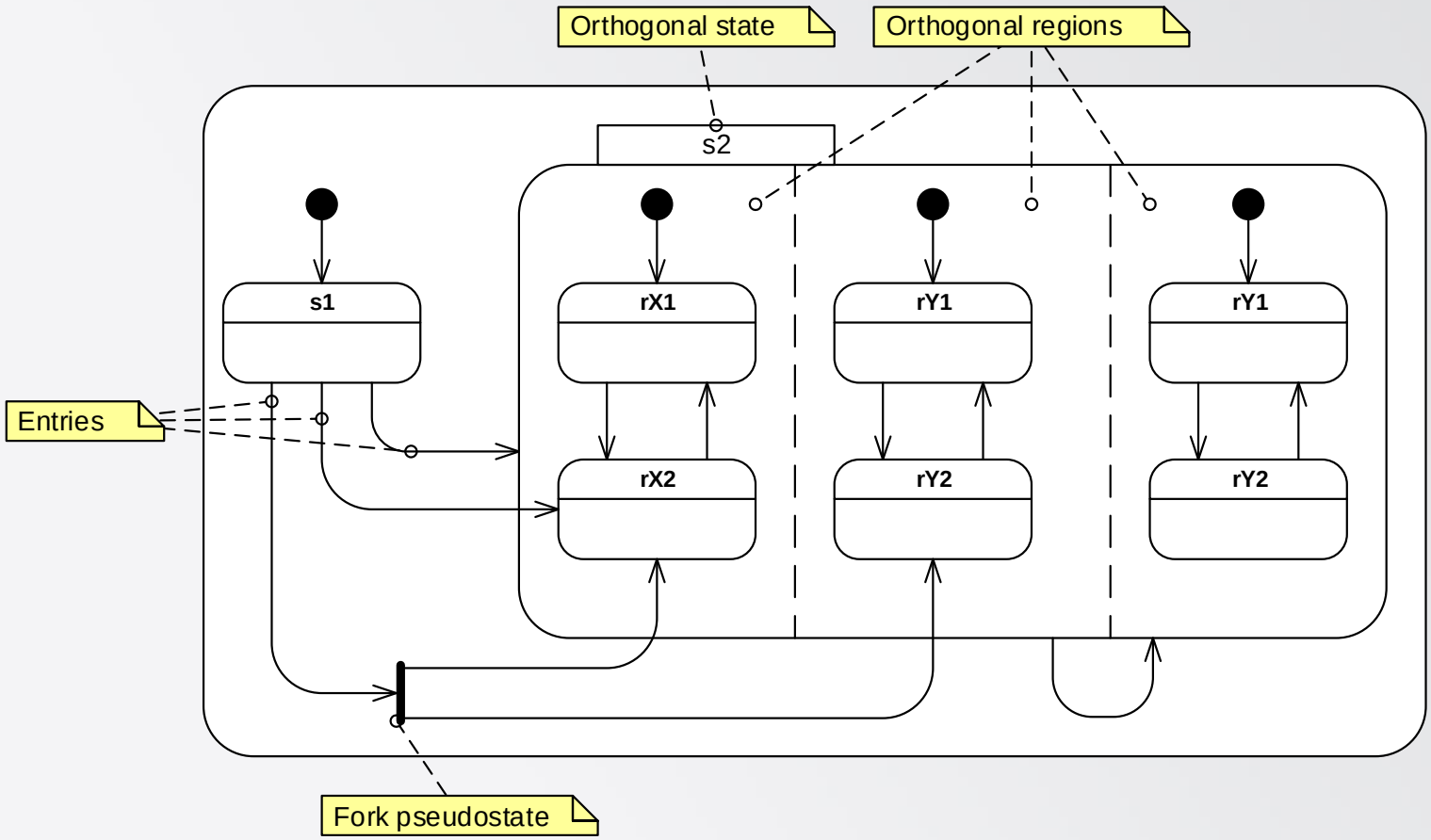


Estado
Entrar a un estado

- Al entrar a un estado, se ejecuta, luego del comportamiento efecto asociado con la transición entrante:
  - su comportamiento de entrada, si está definido,
  - y finalizado el mismo, comienza inmediatamente el comportamiento de actividad, si está definido.
- Entrada a un estado compuesto simple:
  - Entrada por defecto
  - Entrada explícita
  - Entrada por shallow history
  - Entrada por deep history
  - Entrada por *entry point*
- Entrada a un estado ortogonal
  - Entrar desde el borde (por defecto)
  - Entrar explícitamente a una o más regiones (mediante el *pseudoestado fork*).



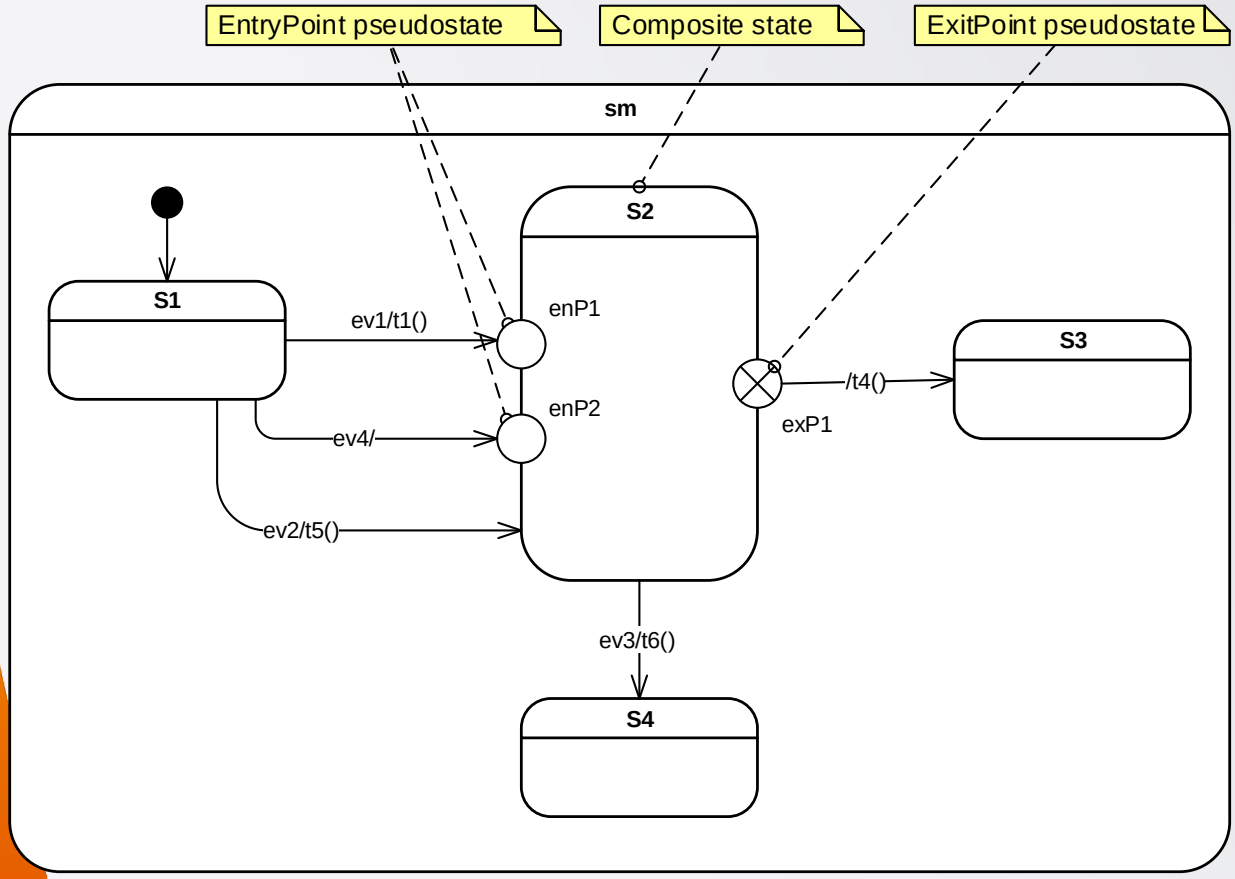
Estado
Ejemplo entradas a un ortogonal

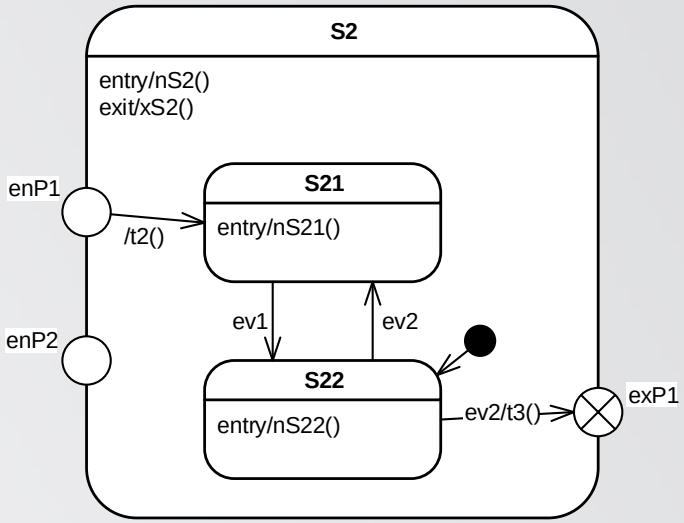


Estado
Salir de un estado

- Al salir de un estado, ya sea compuesto o simple, se ejecuta su comportamiento de salida. Interrumpiendo, previamente, su comportamiento de actividad, si es que existe.
- Al salir de un estado compuesto, la salida comienza desde el estado más interno en la configuración de estados activa, ascendiendo por la jerarquía. Si la salida ocurre desde un *exitPoint*, se ejecuta el exit luego del effect de la transición que termina en dicho pseudoestado.
- Al salir de un estado ortogonal, se abandonan todas sus regiones, luego de ejecutar el exit de dicho estado.

Estado
Compuesto encapsulado

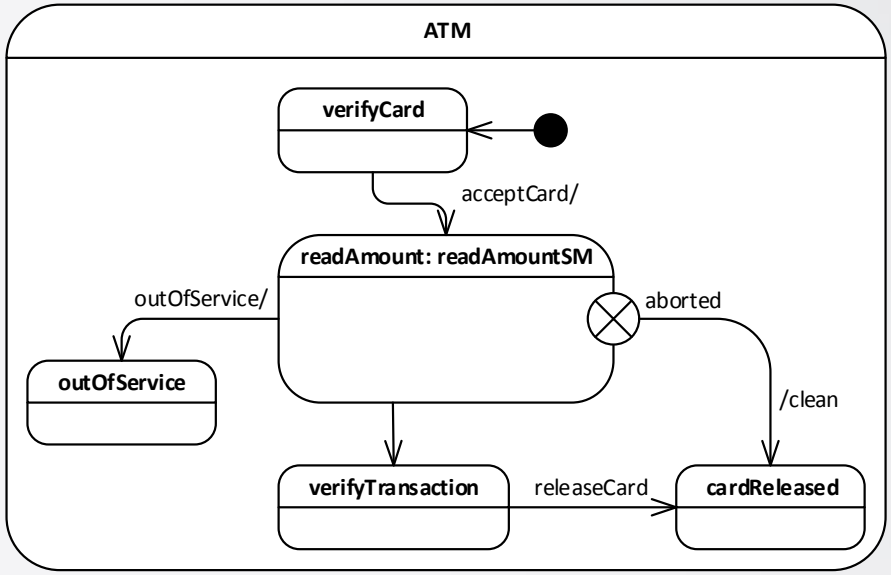


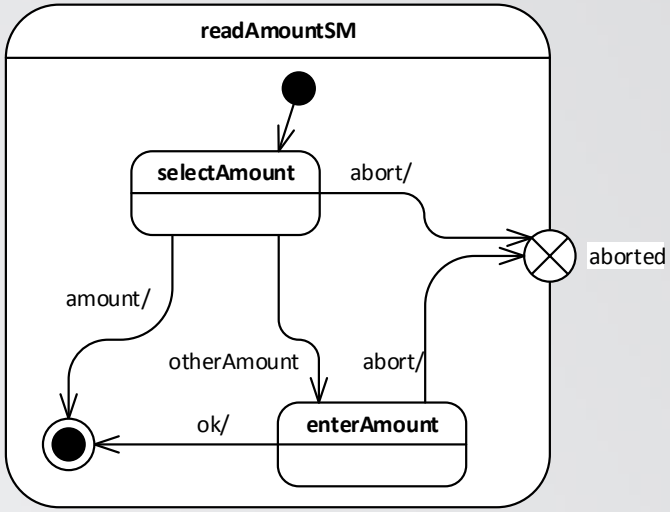


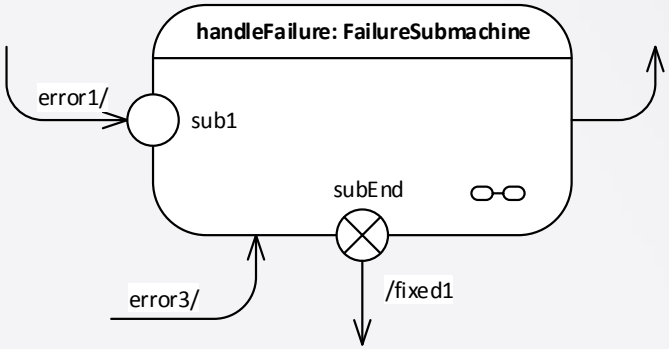
Estado
Compuesto encapsulado

- En ciertas ocasiones es útil encapsular un estado compuesto, impidiendo que las transiciones terminen en alguno de sus vértices internos.
- Para vincular los elementos internos del estado compuesto con sus transiciones entrantes y salientes, se utilizan los pseudoestados entryPoint y exitPoint, respectivamente.
- Entrar mediante entryPoint
  - Es una continuación de la transición entrante, donde el entry del estado compuesto se ejecuta (si está definido) entre el effect de la transición entrante y el correspondiente a la saliente.
- Salir desde exitPoint
  - El exit del estado compuesto se ejecuta (si está definido) entre el effect de la transición entrante al exitPoint y el asociado a la transición saliente.

Estado
Submáquina

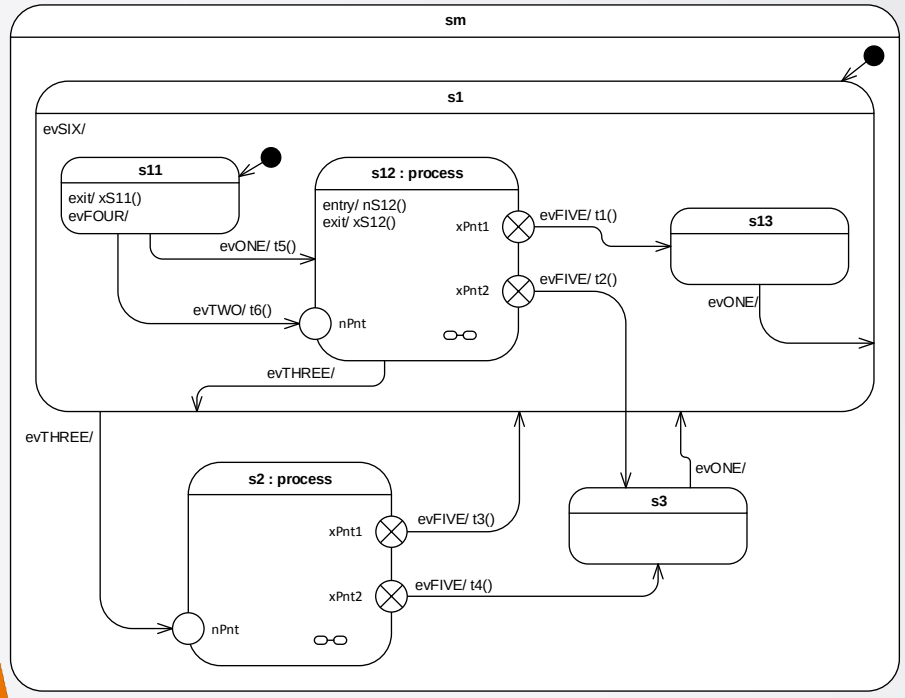


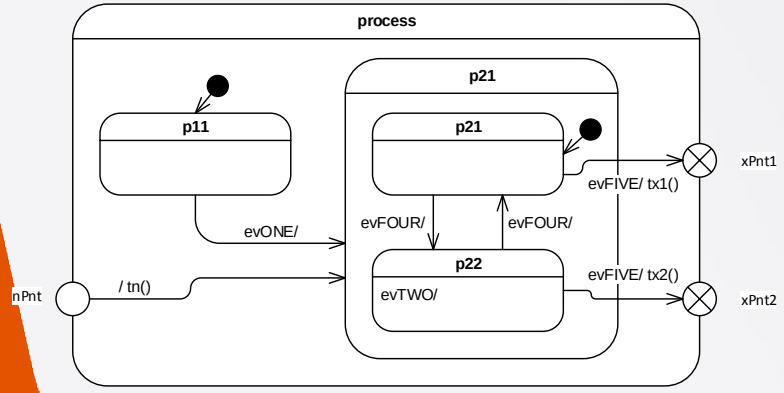




- Las submáquinas son un medio por el cual una única especificación de SM puede reutilizarse más de una vez.
- El estado submáquina representa referencias a las correspondientes SM submáquina.

Estado
Submáquina



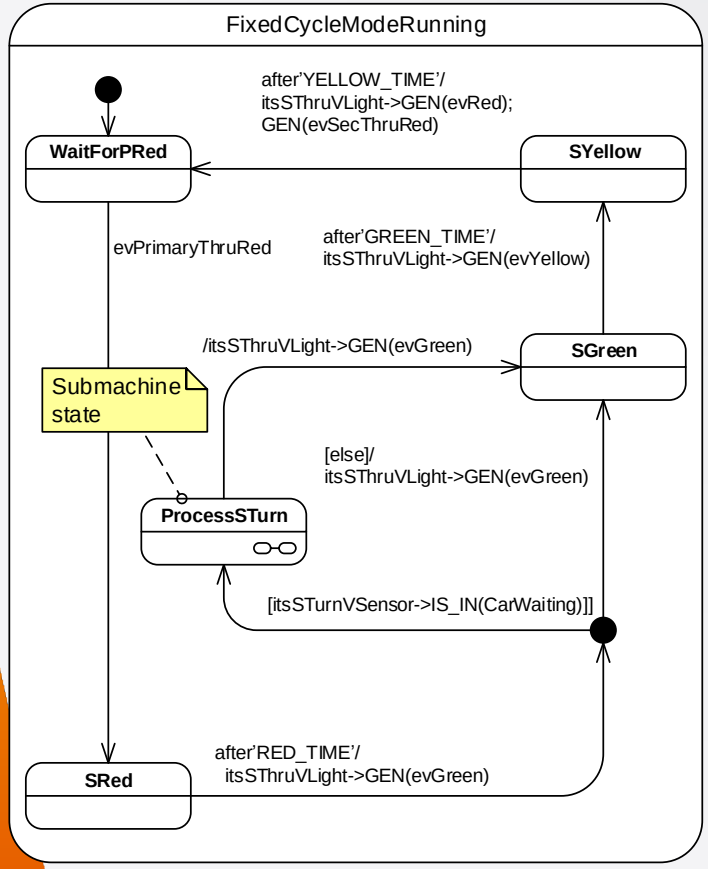


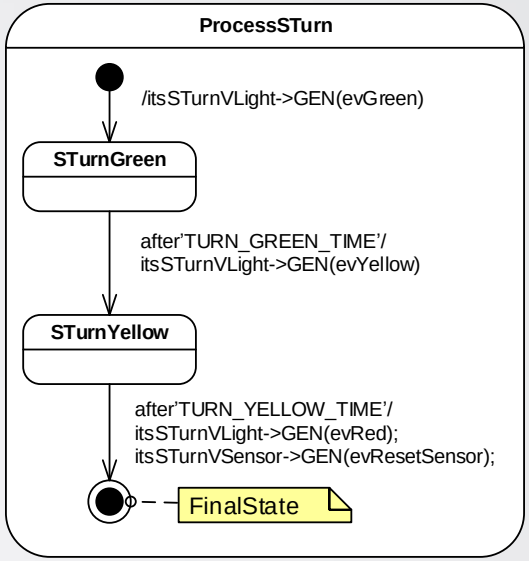
- Para soportar la vinculación entre estado submáquina y la submáquina se utiliza el concepto ConnectionPointReference.
- Cada uno de estos se corresponde con un punto de entrada o salida en la SM submáquina referenciada.
- Conceptualmente, es similar a una macro de un lenguaje de programación.
- Son semánticamente equivalentes a los estados compuestos.

Estado
Submáquina – Entrada y salida

- Puede ingresarse a un estado submáquina desde
  - su pseudoestado initial o,
  - cualquiera de sus puntos de entrada
- Puede abandonarse por consecuencia de
  - alcanzar su EstadoFinal,
  - el disparo de una transición grupal, originada en el estado submáquina o,
  - por medio de alguno de sus puntos de salida

Estado
EstadoFinal





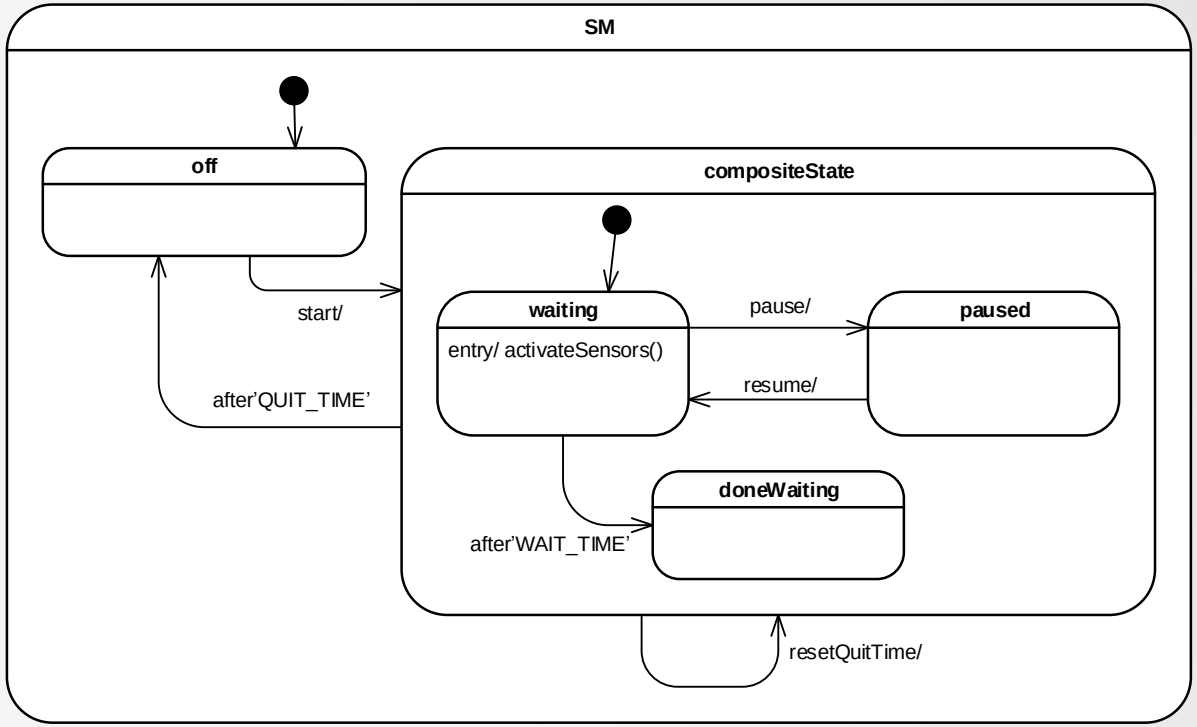
- Es un tipo especial de estado, que al ingresar al mismo, la región que lo contiene concluye.
- Si esta última, está contenida directamente en una SM y el resto de las regiones también finalizaron, entonces el comportamiento completo de la SM finaliza.

Eventos

- Un evento es algo que puede ocurrir en un determinado momento. Este puede tener muchas ocurrencias, las cuales pueden suceder en diferentes momentos.
- Los eventos que provocan una respuesta en el comportamiento, puede modelarse explícitamente en UML, incluyendo los
  - **TimeEvents** que ocurren en un momento específico o luego de un tiempo determinado,
  - ChangeEvents que ocurren cuando un valor *booleano* específico se vuelve verdadero y los
  - **MessageEvents** que ocurren al recibir un *mensaje*, que es una comunicación de un comportamiento con otro, solicitando la llamada de una operación (CallEvent) o la recepción de una señal (**SignalEvent**).

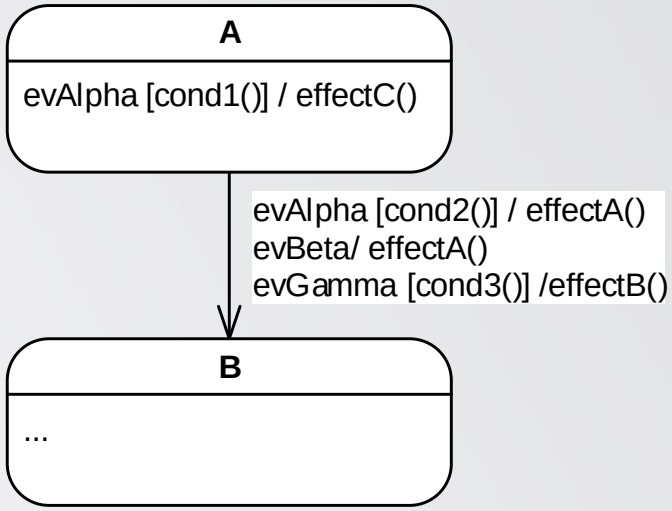
Eventos
TimeEvent

- Especifica un instante en el tiempo en el cual este ocurre.
- Puede ser absoluto o relativo.
- En una SM, si un estado contiene un disparador cuyo evento es TimeEvent relativo, el tiempo de inicio para dicho evento es aquel en el que la SM ingresó a dicho estado.



Transición

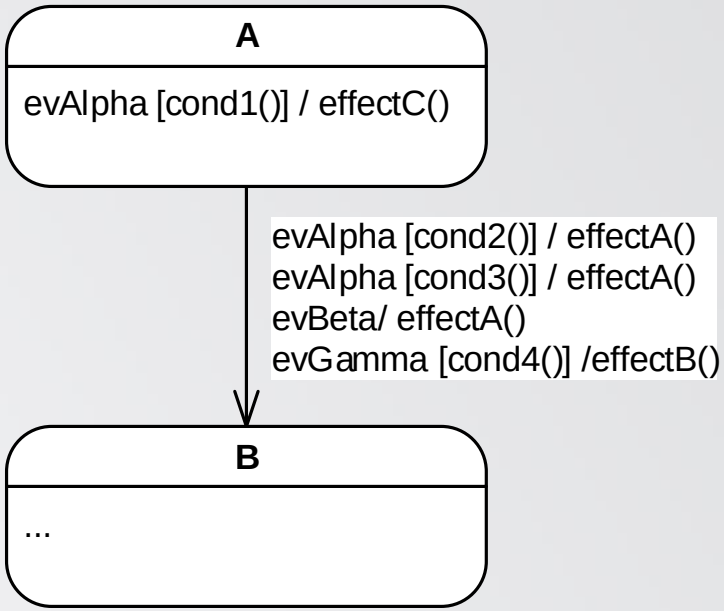
- Es un arco dirigido que se origina en un único vértice y termina en un único vértice, donde ambos pueden ser iguales.
- Especifica un fragmento válido del comportamiento de una SM.
- Puede tener asociado un comportamiento effect, que se ejecuta cuando la transición es recorrida.
- Transfiere la ejecución de la SM, de una configuración de estados estable a otra.
- Puede tener asociado un conjunto de disparadores (son excluyentes entre si), los cuales especifican un evento, cuya ocurrencia, una vez despachada, puede provocar (disparar) la transición.
- Está *habilitada* si la ocurrencia del evento despachada coincide con alguno de sus disparadores.



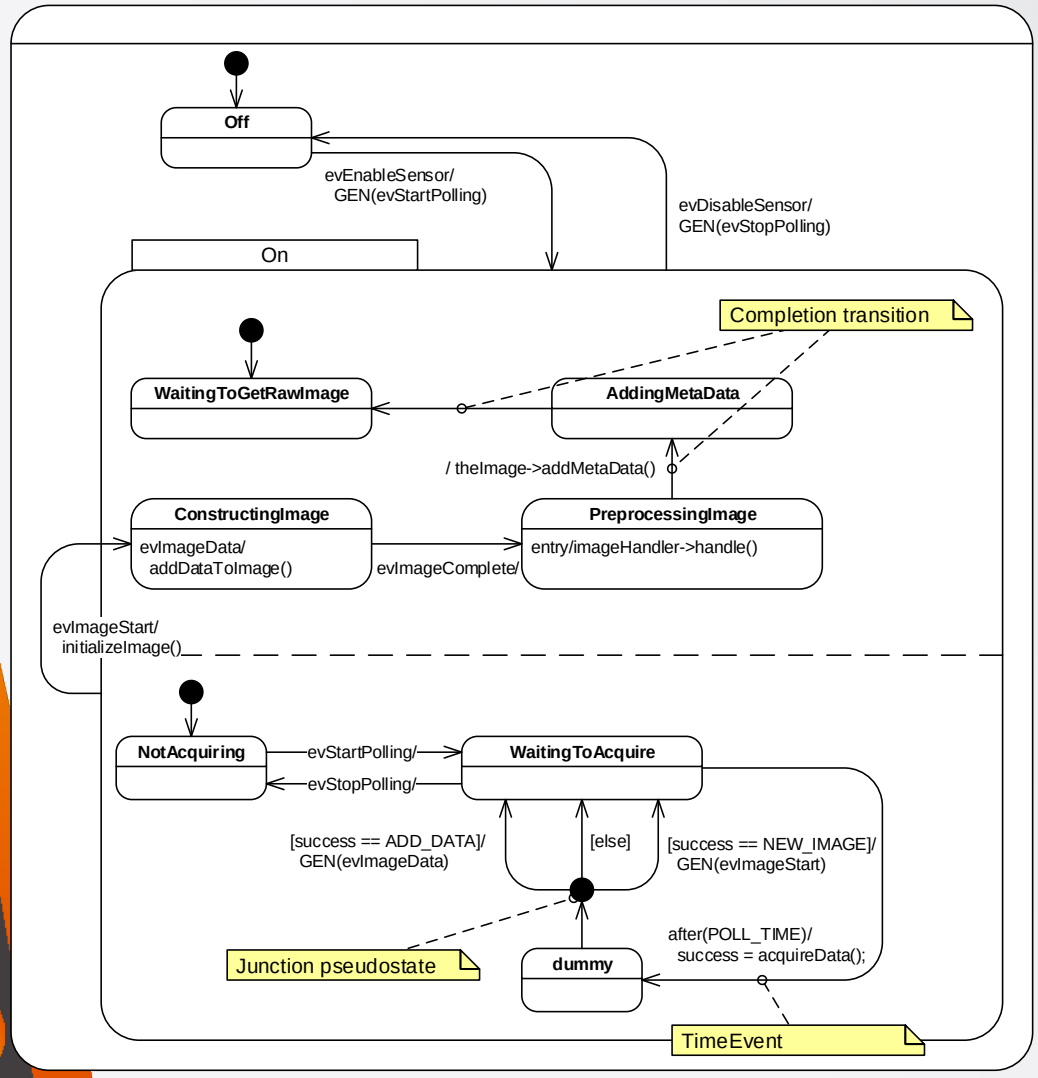
Transición
Guardas

- Una transición puede tener asociada una guarda.
- Las evaluadas como falsas inhabilitan sus transiciones.
- Se evalúan antes de habilitar la transición compuesta que la contiene, a menos que se origine en un pseudoestado choice.

- Una transición sin guarda, se la trata como si tuviera una siempre verdadera.
- No deben incluir expresiones que provoquen efectos laterales. Si es así, el modelo se considera incorrecto.

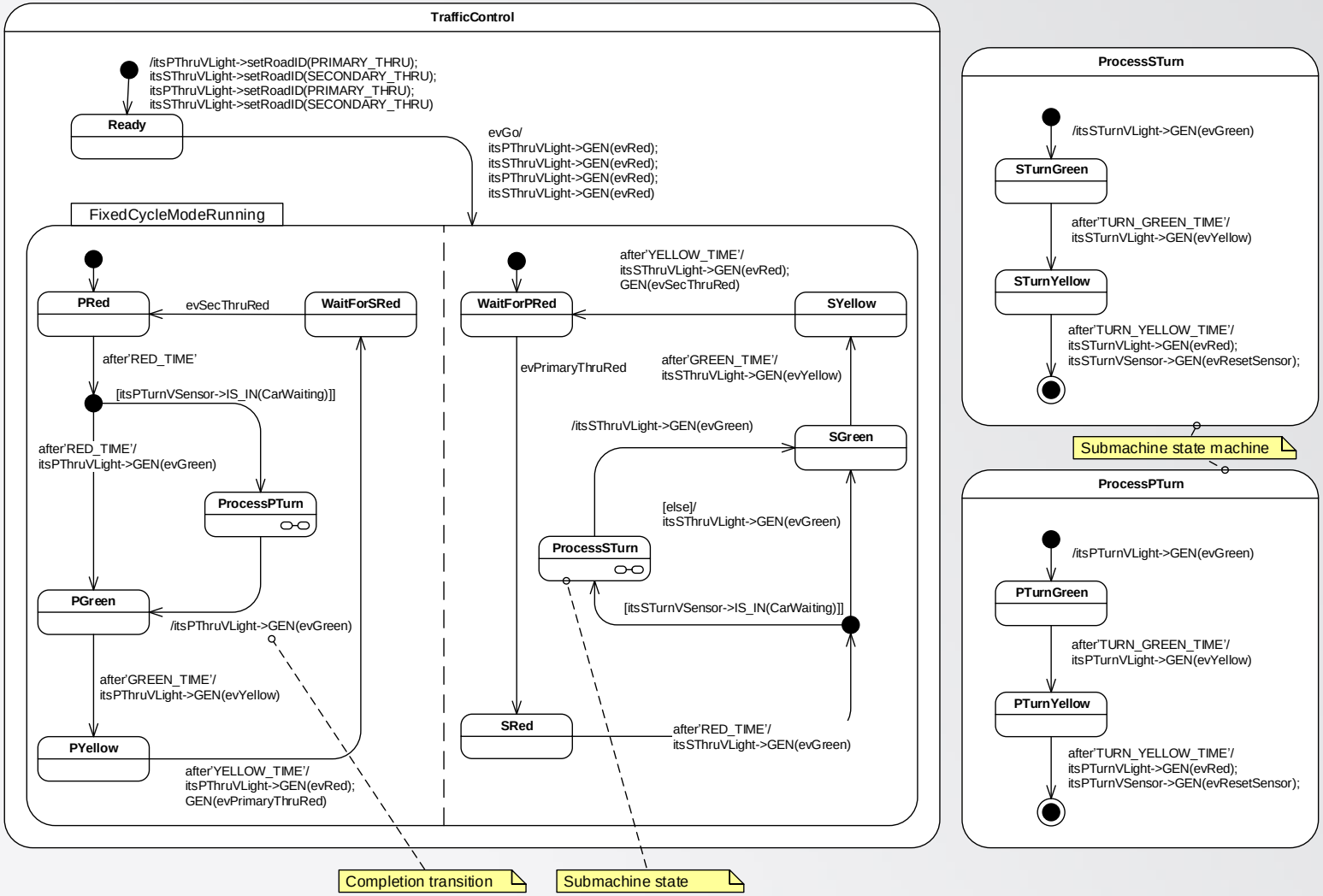


Transición
Terminación



- Es un tipo especial de transición, cuyo disparador está implícito.
- El evento que habilita dicho disparador se lo denomina *evento de terminación*.
- Este implica que han finalizado todos los comportamientos asociados con el estado origen de la transición de terminación.
- El evento de terminación tienen máxima prioridad de despacho.
- También se la conoce como “transición nula”.

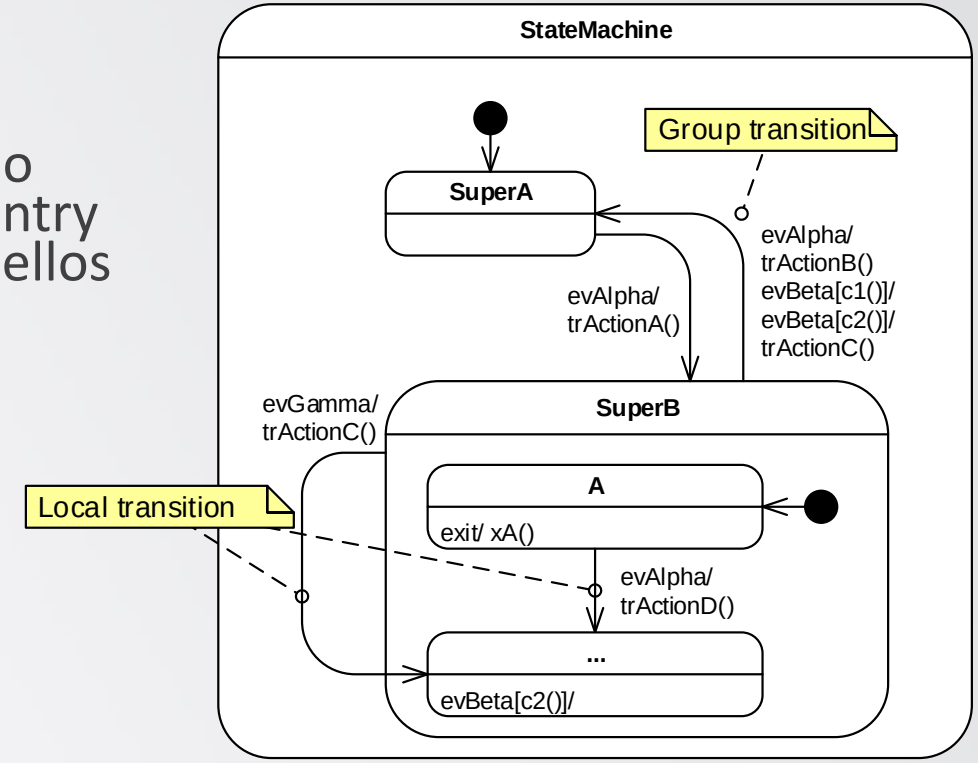
Transición
Terminación



Transición
Grupal o de alto nivel

- Son aquellas cuyo estado origen es un estado compuesto.
- Si es de tipo externa
  - provoca la salida de todos los subestados del estado compuesto, ejecutando todo comportamiento exit definido, a partir de sus estados más internos de la configuración de estados activa.

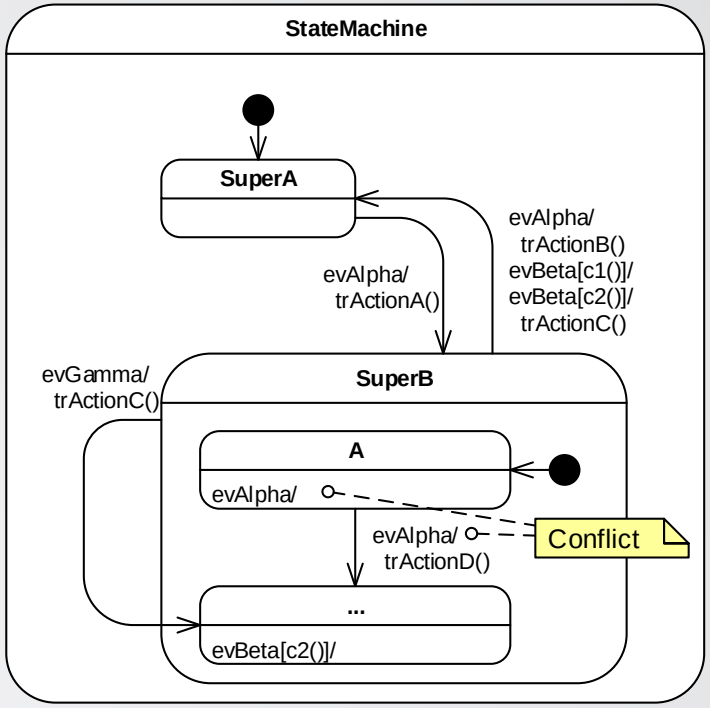
- Si es de tipo local
  - provoca la ejecución de los comportamientos exit del estado origen y los comportamientos entry del estado destino, pero no aquellos asociados con el estado que la contiene.



Transición
Conflictivas

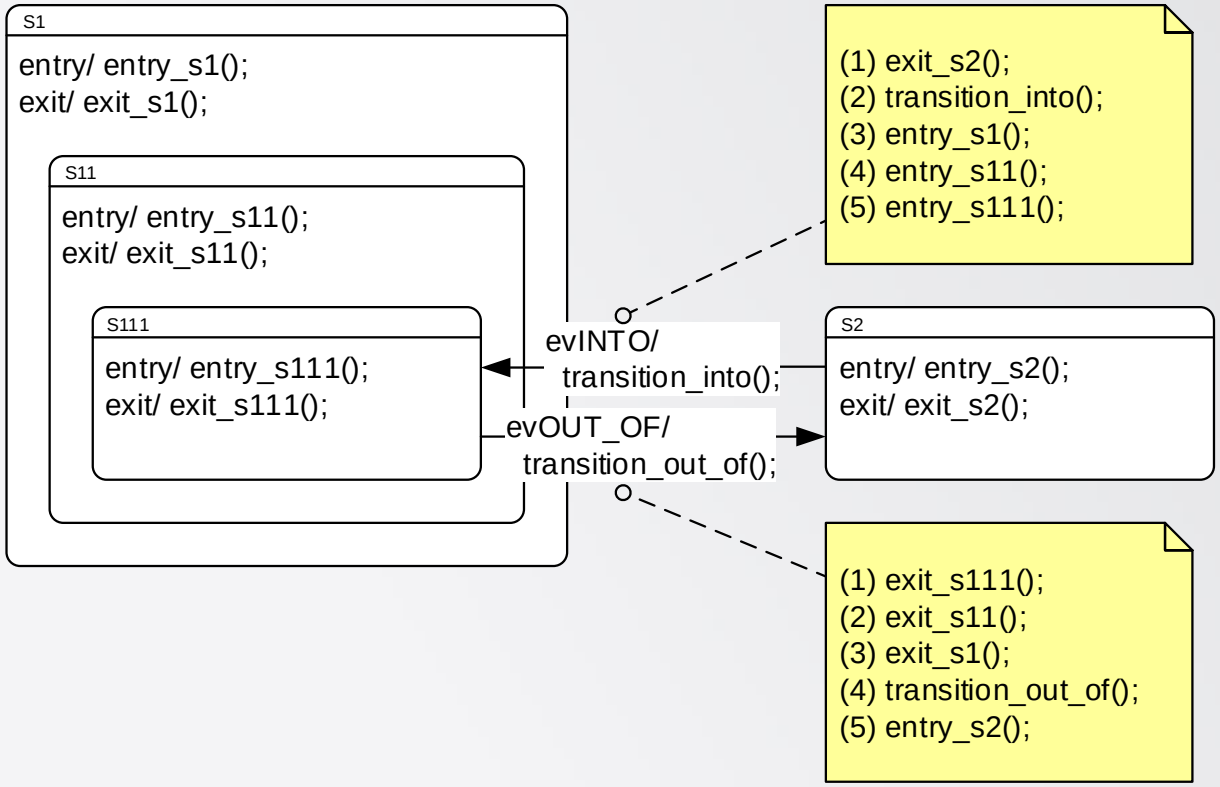
- Es posible que en una SM se habiliten simultáneamente más de una transición. En cuyo caso, tales transiciones pueden estar en conflicto entre sí.
- Dos transiciones se dicen estar en conflicto, si ambas abandonan el mismo estado.
- Pueden dispararse transiciones en simultáneo, sólo si estas ocurren en regiones mutuamente excluyentes.

- Una transición **interna** entra únicamente en conflicto con aquellas transiciones que provocan la salida del estado en cuestión.
- En situaciones en las que hay transiciones en conflicto, la selección de cuál transición se disparará, se basa en parte, en una prioridad implícita, basada en su posición relativa en la jerarquía de estados. La prioridad de una transición se define en función de su estado origen.



Transición
Secuencia de ejecución

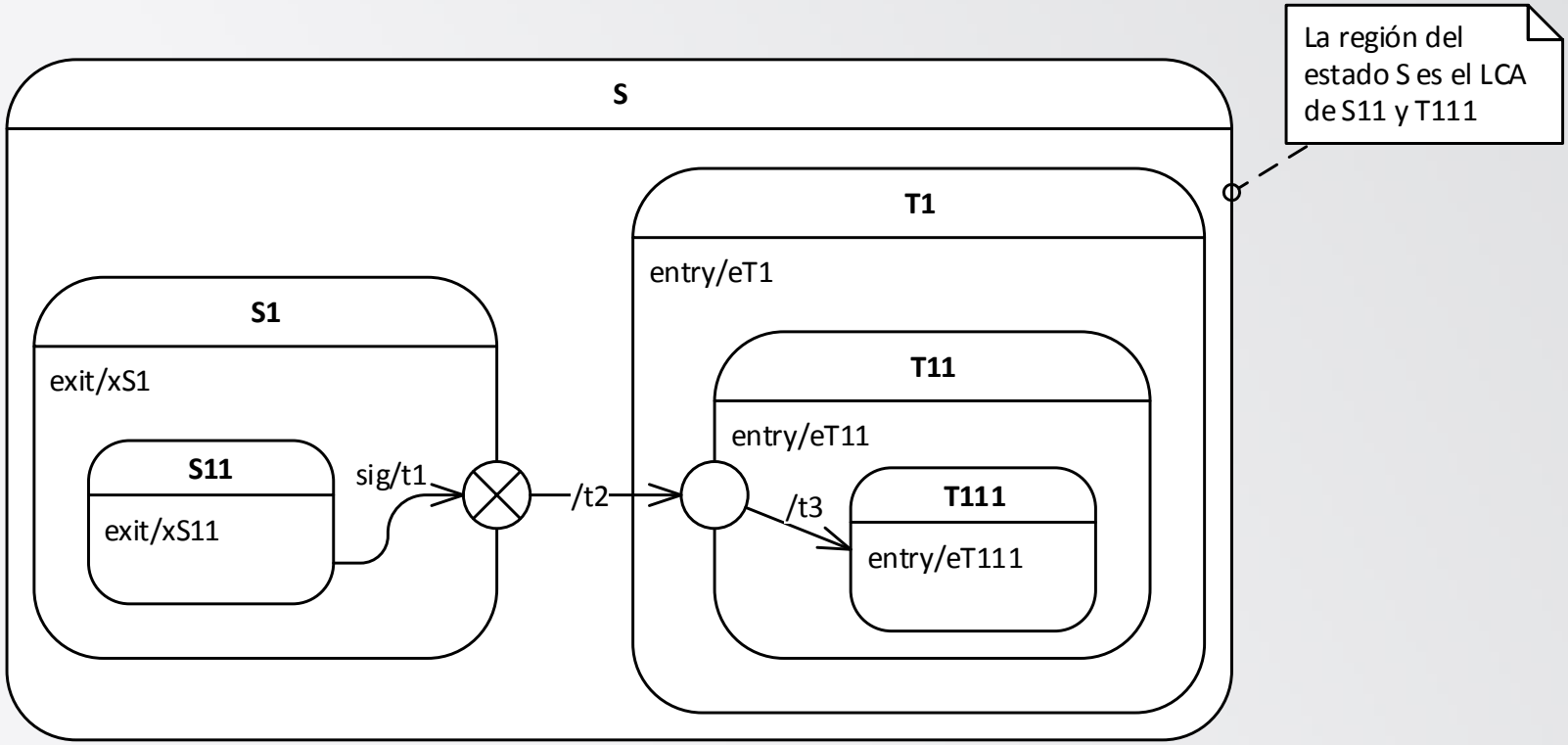
- Orden de ejecución: salida-transición-entrada.
- Acciones de entrada - ejecutadas desde afuera hacia adentro.
- Acciones de salida - ejecutadas desde adentro hacia afuera.



Transición
Secuencia de ejecución

- Secuencia de acciones ejecutadas:

```
xS11, t1, xS1, t2, eT1, eT11, t3, eT111
```



Transición
Secuencia de ejecución

- Toda transición, excepto las internas y las locales, provocan la salida del estado origen (principal) y la entrada al estado destino (principal).
- Secuencia
  - Partiendo del origen principal, se abandonan los estados contenidos en este último.
  - La salida de estados continua hasta alcanzar la primera región que contiene, directa o indirectamente, tanto el estado origen principal como el estado destino principal. Esta región se la conoce como su LCA (Least Common Ancestor).
  - Encontrado su LCA, se ejecuta el comportamiento effect de la transición que conecta la sub-configuración de estados origen con la sub-configuración de estados destino.
  - Se ingresa a la configuración de estados que contiene el estado destino principal, comenzando con el estado más externo en la región LCA, que contiene el estado destino principal.

Procesamiento de eventos
Paradigma "ejecutar hasta finalizar"

- Tras la creación, una SM llevará a cabo su inicialización durante la cual ejecuta una transición compuesta inicial, para luego ingresar en un *punto de espera*, representado por una configuración de estados estable.
- Esta condición se mantiene hasta que se despacha un evento almacenado en su event pool.
- En cuyo caso el evento se evalúa y, si coincide con un disparador válido de la SM, y hay al menos una transición habilitada, se ejecuta un *paso* de la máquina de estados.
- Que implica ejecutar una transición compuesta y finalizar en una configuración de estados estable, es decir, en el siguiente punto de espera.
- Este ciclo se lo conoce como **RTC** o **Run To Completion**.
- RTC significa que se despacha un evento por vez.

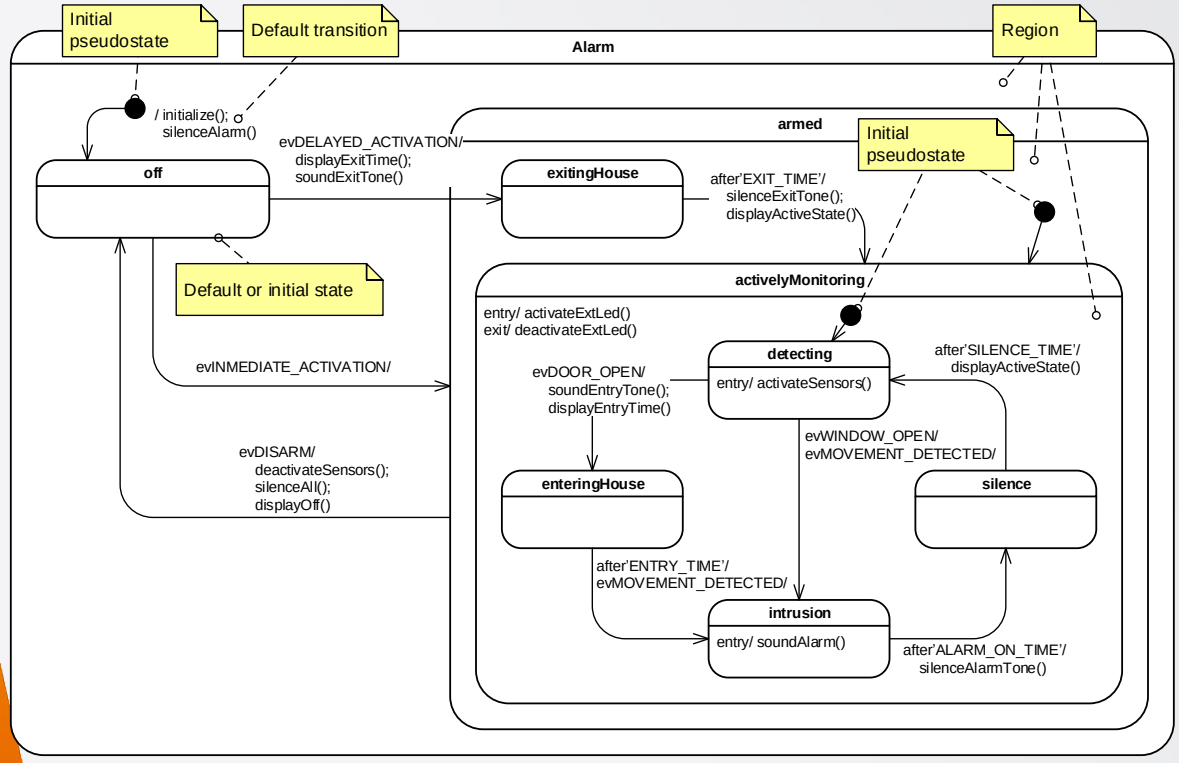
Procesamiento de eventos
Paradigma “ejecutar hasta finalizar”

- Si ninguna transición está habilitada y el correspondiente tipo de evento no se encuentra en las listas de disparadores diferidos de la configuración de estados activa, se descarta la ocurrencia del evento despachado y el paso RTC se resuelve de manera trivial.
- **Para la implementación**: una SM puede interrumpirse, ya que en una implementación particular, un hilo de ejecución que maneja un paso de la SM, puede suspenderse, permitiendo la ejecución de hilos de mayor prioridad, y una vez que se le asigne el procesador nuevamente, puede reanudar de manera segura su ejecución y así completar el procesamiento del evento.

Pseudoestado

- Es una abstracción, en el grafo de SM, que engloba diferentes tipos de vértices *transitorios*.
- Generalmente, se utilizan para conectar diferentes transiciones para lograr una más compleja, denominada *transición compuesta*.
- La semántica específica de un pseudoestado depende de su tipo.

Pseudoestado
Initial

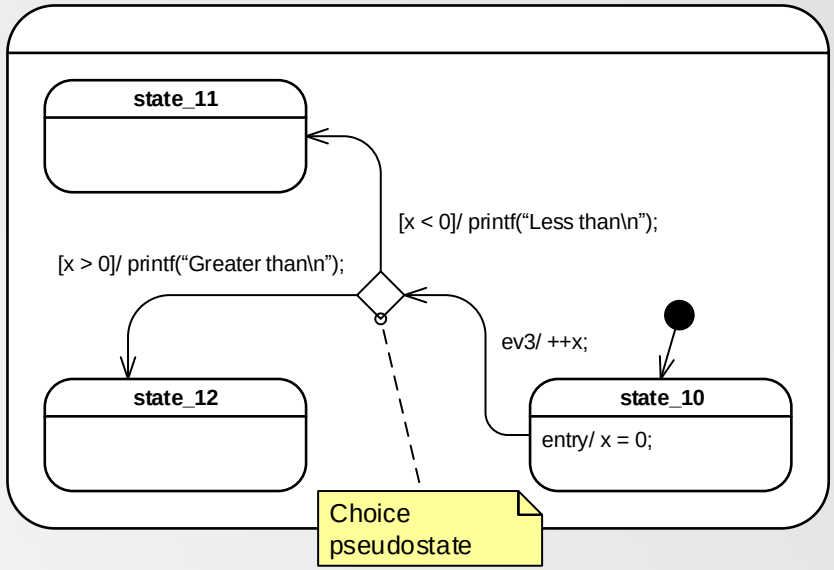


- Representa un punto de partida de una región, cuando se la activa por defecto.
- Es el origen de una única transición, la cual puede tener asociado un comportamiento effect, pero no así un disparador o una guarda.
- Una región puede tener asociado, a lo sumo, un vértice initial.

Pseudoestado
Junction

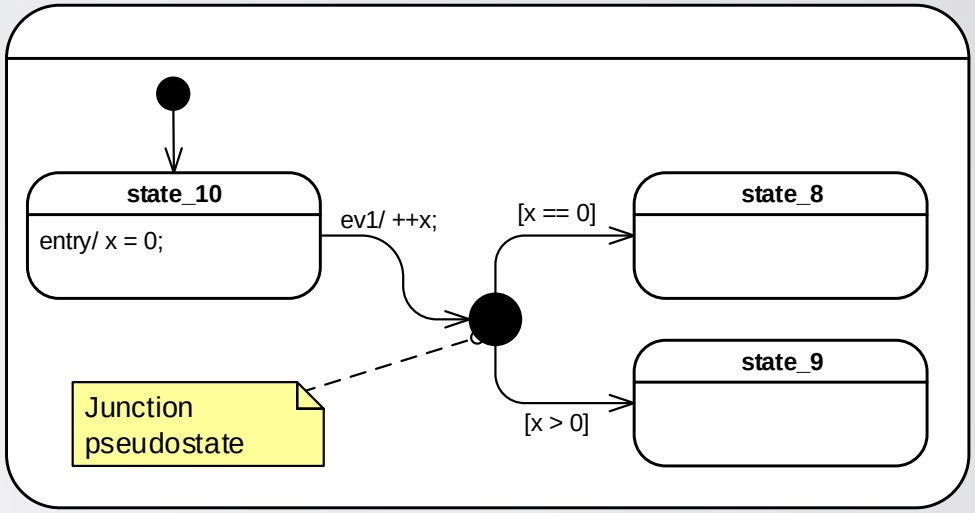
- Conecta diferentes transiciones en caminos compuestos entre estados.
- Puede utilizarse para
  - combinar diferentes transiciones entrantes en una única transición saliente, que representa un camino continuo y compartido, o
  - dividir una transición entrante en múltiples transiciones salientes con diferentes guardas. Estas se evalúan ***antes*** de ejecutar una transición compuesta que contiene este pseudoestado. La transición saliente se la conoce como rama condicional *estática*.

- Si todas las guardas son falsas, se inhibe la transición compuesta.
- En ciertas situaciones, puede utilizarse la guarda “else”.
- Si más de una guarda es verdadera, se elige una de estas. La elección no está definida.
- También se lo conoce como pseudoestado *conditional*.

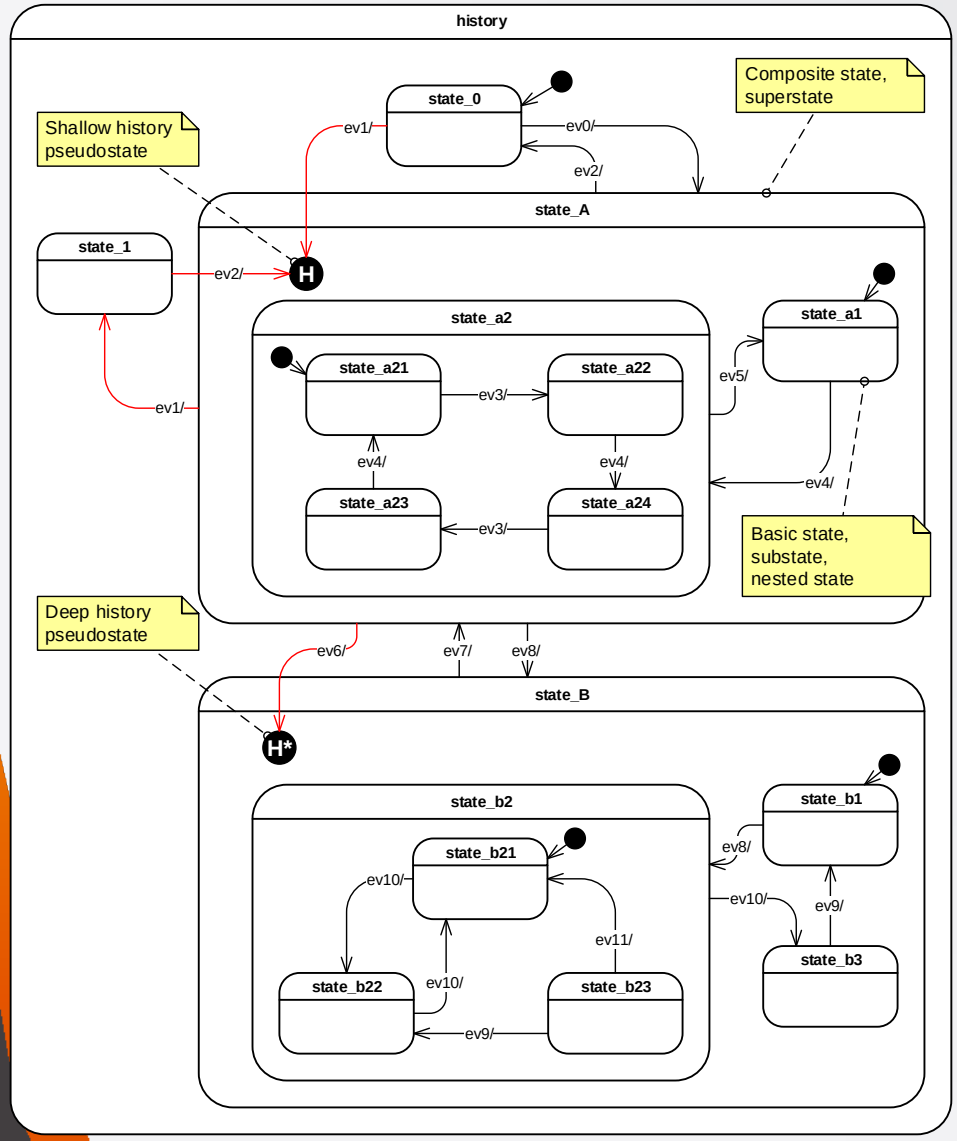


Pseudoestado
Choice

- Similar al pseudoestado junction, con la diferencia que las guardas de todas las transiciones salientes se evalúan *dinámicamente*, cuando el recorrido de la transición compuesta alcanza este pseudoestado.
- Si todas las guardas son falsas, el modelo se considera incorrecto.
  - Se recomienda, el uso de la guarda "else".
  - Si más de una guarda es verdadera, se elige una de estas. La elección no está definida.
  - También se lo conoce como pseudoestado *branch*.



Pseudoestado
History



- Si el estado activo es a24 y recibe ev1 seguido de ev2, **¿cuál es el próximo estado activo?**.
- Si el estado activo es b23, y recibe el evento ev7 seguido de ev6, **¿cuál es el próximo estado?**. En el mismo contexto, si recibe ev7 seguido de ev8, **¿cuál es el estado activo final?**.

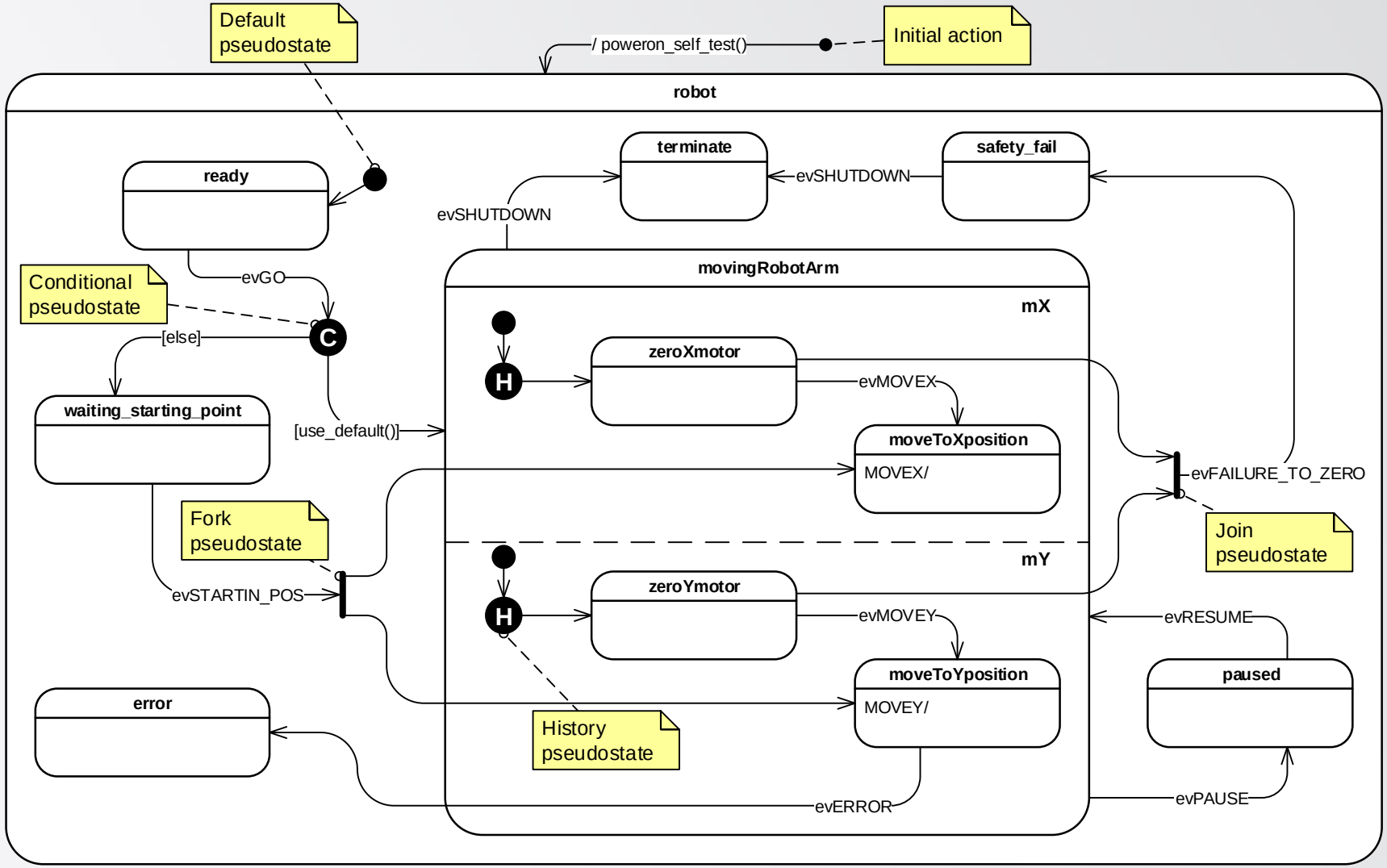
Pseudoestado
Deep History

- Es una clase de variable, que representa la configuración de estados activa más reciente de la región a la cual pertenece.
- Cuando es destino de una transición, implica restaurar la región a la configuración de estados retenida en la historia, cumpliendo la semántica para entrar a un estado.
- Ejecutando en el orden apropiado, los comportamientos entry de todos los estados de la configuración restaurada, comenzando por el estado más externo.
- A lo sumo, puede definirse una transición saliente de este pseudoestado, que tenga por destino un estado de la región.
- Puede definirse sólo para estados compuestos, a lo sumo, uno por cada región.

Pseudoestado
Shallow History

- Es una clase de variable, que representa el subestado activo más reciente de la región a la cual pertenece, pero no así sus subestados.
- Cuando es destino de una transición, implica restaurar la región al subestado retenido en la historia, cumpliendo la semántica para entrar a un estado.
- A lo sumo, puede definirse una transición saliente de este pseudoestado, que tenga por destino un estado de la región, denominado *estado shallow history por defecto*.
- Puede definirse sólo para estados compuestos, a lo sumo, uno por cada región.

Pseudoestado
Join & Fork



Al igual que los estados ortogonales, son complejos de implementar

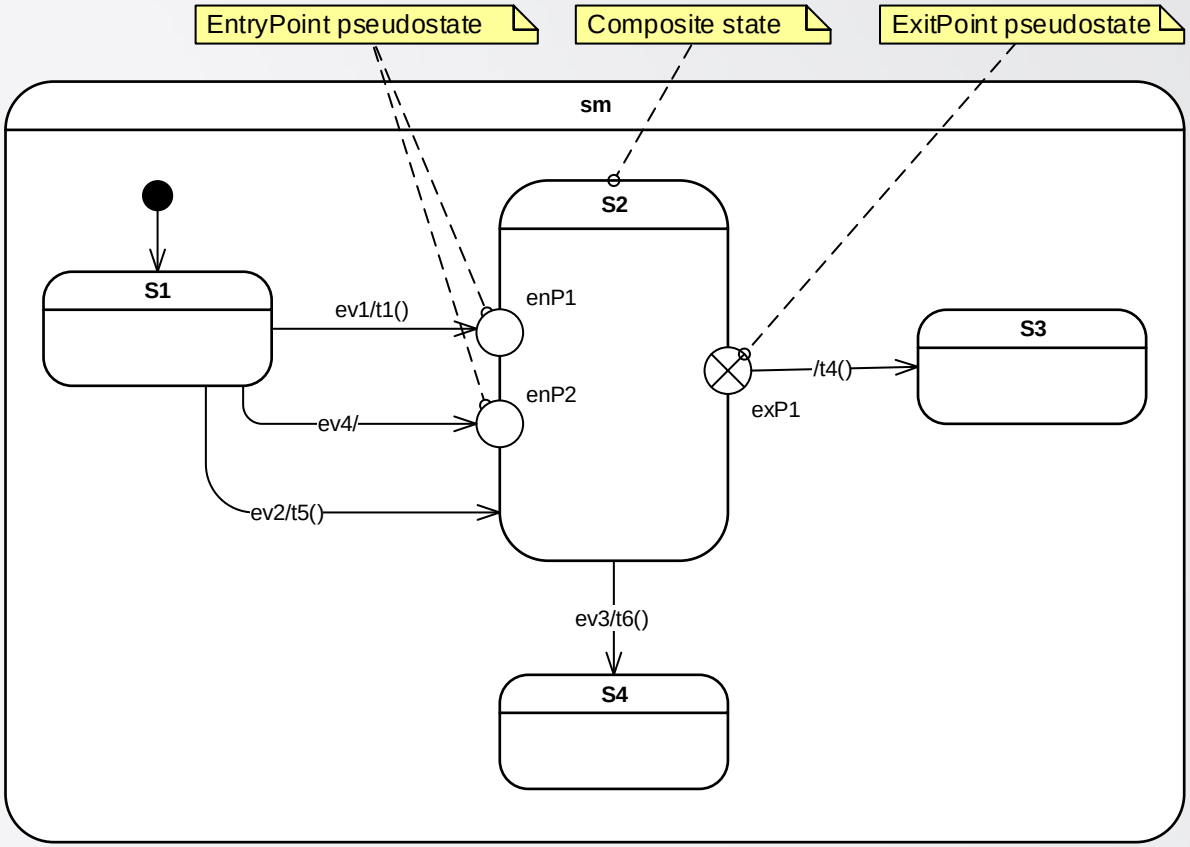
Pseudoestado
Join

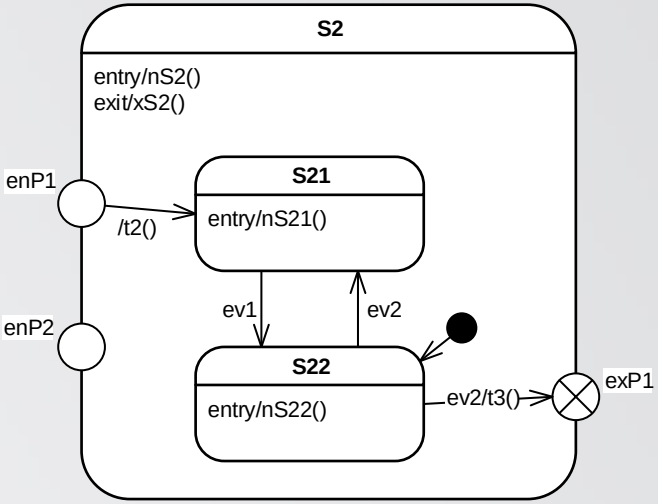
- Actúa como un vértice común para dos o más transiciones, que se originan en vértices de diferentes regiones ortogonales, de un mismo estado compuesto.
- Sus transiciones entrantes no pueden tener ni guardas ni disparadores.
- Cumple la función de **sincronización**, ya que todas las transiciones entrantes deben completarse para que continúe la ejecución a través de su transición saliente.
- Sólo puede definirse una única transición saliente.

Pseudoestado
Fork

- Divide una transición entrante en dos o más transiciones, que terminan en vértices de diferentes regiones ortogonales, de un mismo estado compuesto.
- Sus transiciones salientes no puede tener ni disparadores ni guardas.

Pseudoestado
EntryPoint & ExitPoint





Pseudoestado
EntryPoint

- Representa un punto de entrada a una SM o un estado compuesto, que permite encapsular el interior de estos.
- A lo sumo puede tener asociada una transición saliente. Si no está definida se ingresa por defecto.
- Si intervienen diferentes regiones, actúa como un pseudoestado fork.
- Si el estado que lo contiene tiene asociado un comportamiento entry, este se ejecuta **antes** que cualquier comportamiento asociado con la transición saliente.

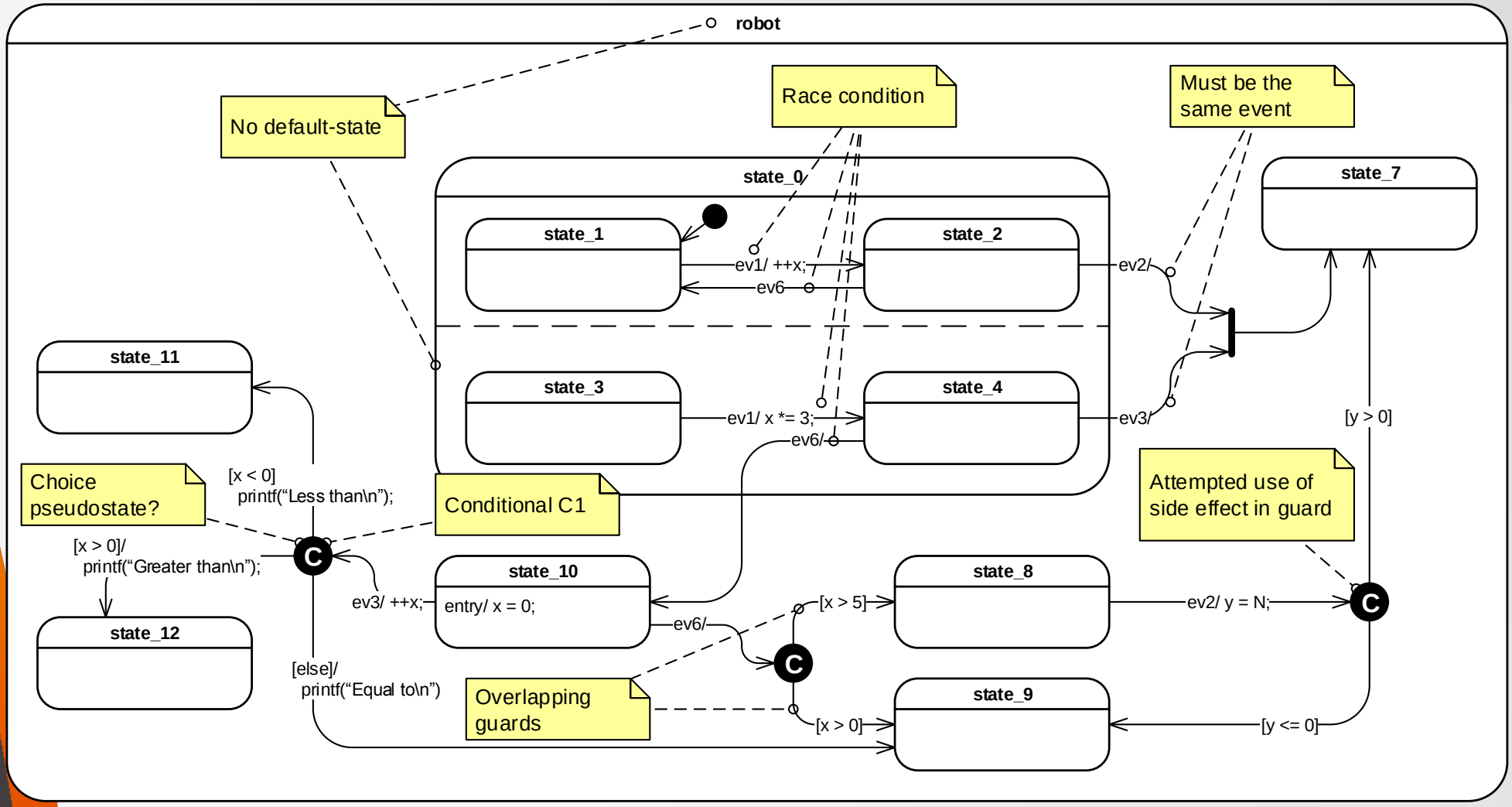
Pseudoestado
ExitPoint

- Representa un punto de salida de una SM o un estado compuesto, que permite encapsular el interior de estos.
- Las transiciones que terminan en estos implican la salida de sus contenedores.
- Si las transiciones que terminan en este, provienen de diferentes regiones ortogonales dentro del estado, entonces este se comporta como uno del tipo join.

Pseudoestado
Terminate

- Al ingresar a este, la ejecución de la SM finaliza inmediatamente, sin salir de sus estados activos ni ejecutar los comportamientos exit.
- Asimismo, se cancelan automáticamente las actividades en ejecución.
- Es equivalente a invocar la destrucción del objeto que aloja dicho comportamiento.

Diagramas “mal formados”



Diagramas “mal formados”

- Si el estado actual es state_10 y recibe el evento ev3, **¿cuál es el estado final de la transición?**.
- Si en lugar de C1 utilizamos un pseudoestado branch, **¿cuál es el estado final de la transición?**.
- Si todas las guardas son verdaderas, **¿cuál es la transición esperada luego de un conditional o branch?**.
- **¿Si todas las guardas son falsas?**
- **¿Cómo podría salvar la condición de carrera en el estado state_0 recibiendo el evento ev1?**

Preguntas

Referencias

[1] D. Harel, "Statecharts: A Visual Formalism for Complex Systems", Sci. Comput. Programming 8 (1987), pp. 231–274.

[2] RKH, “RKH Sourceforge download site,”, http://sourceforge.net/projects/rkh-reactivesys/ , August 7, 2010.

[3] Object Management Group, “Unified Modeling Language: Superstructure version 2.1.1,” formal/2007-02-05, Febrary 2007.

[4] B. P. Douglass, “Design Patterns for Embedded Systems in C,”, Elseiver, October 7, 2010

[5] B. P. Douglass, “Real-Time UML: Advances in the UML for Real-Time Systems (3rd Edition),”, Elseiver, October 4, 2006

[6] B. P. Douglass, “Real-Time Agility: The Harmony/ESW Method for Real-Time and Embedded Systems Development,”, Elseiver, June 19, 2009

[7] M. Samek, “Practical UML Statecharts in C/C++, Second Edition: Event-Driven Programming for Embedded Systems,” Elsevier, October 1, 2008.