



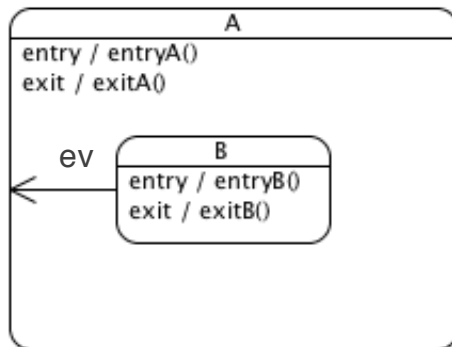
# Desenvolvimento de Sistemas Software

## Aula Teórica 10: Modelação de comportamento / Máquinas de Estado (II)



# Transições locais vs. transições externas

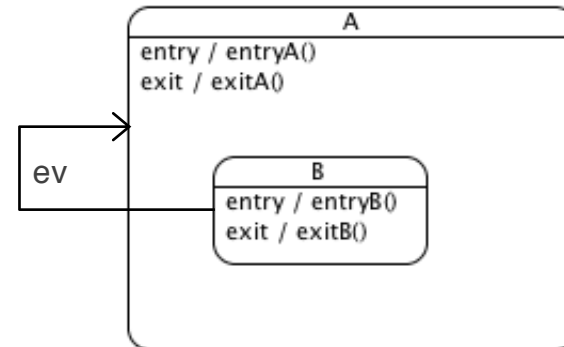
## Transições locais



Executa:  
1. exitB()  
2. ...

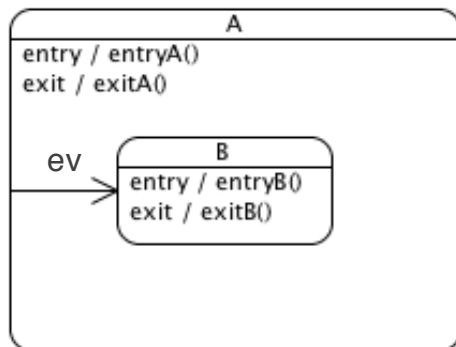
(sub-estado para super-estado)

## Transições externas



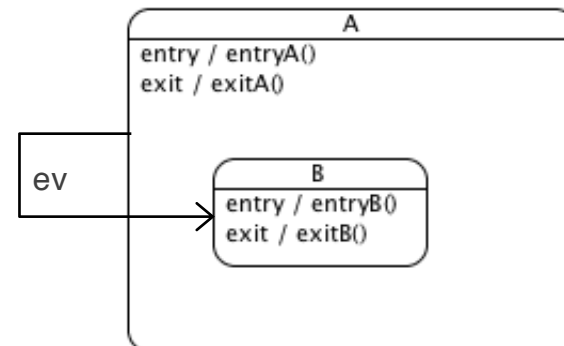
Executa:  
1. exitB()  
2. exitA()  
3. entryA()  
4. ...

(sub-estado para super-estado)



Executa:  
1. ...  
2. entryB()

(super-estado para sub-estado)



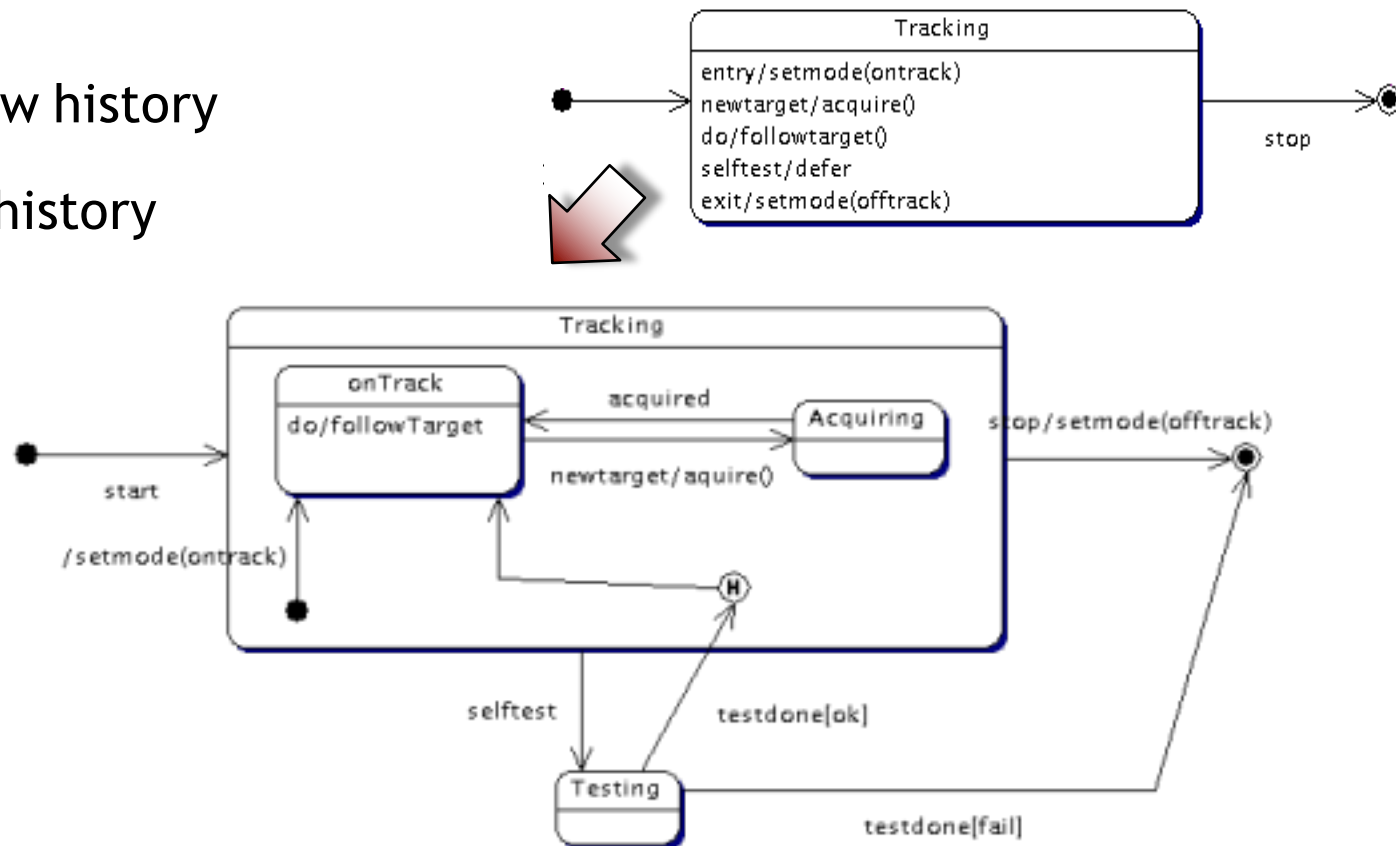
Executa:  
1. ...  
2. exitA()  
3. entryA()  
4. entryB()

(super-estado para sub-estado)

# Pseudoestados de História

- Permitem modelar interrupções – actividade da máquina é retomada no estado em que se encontrava aquando da última saída

- $\textcircled{H}$  shallow history
- $\textcircled{H^*}$  deep history





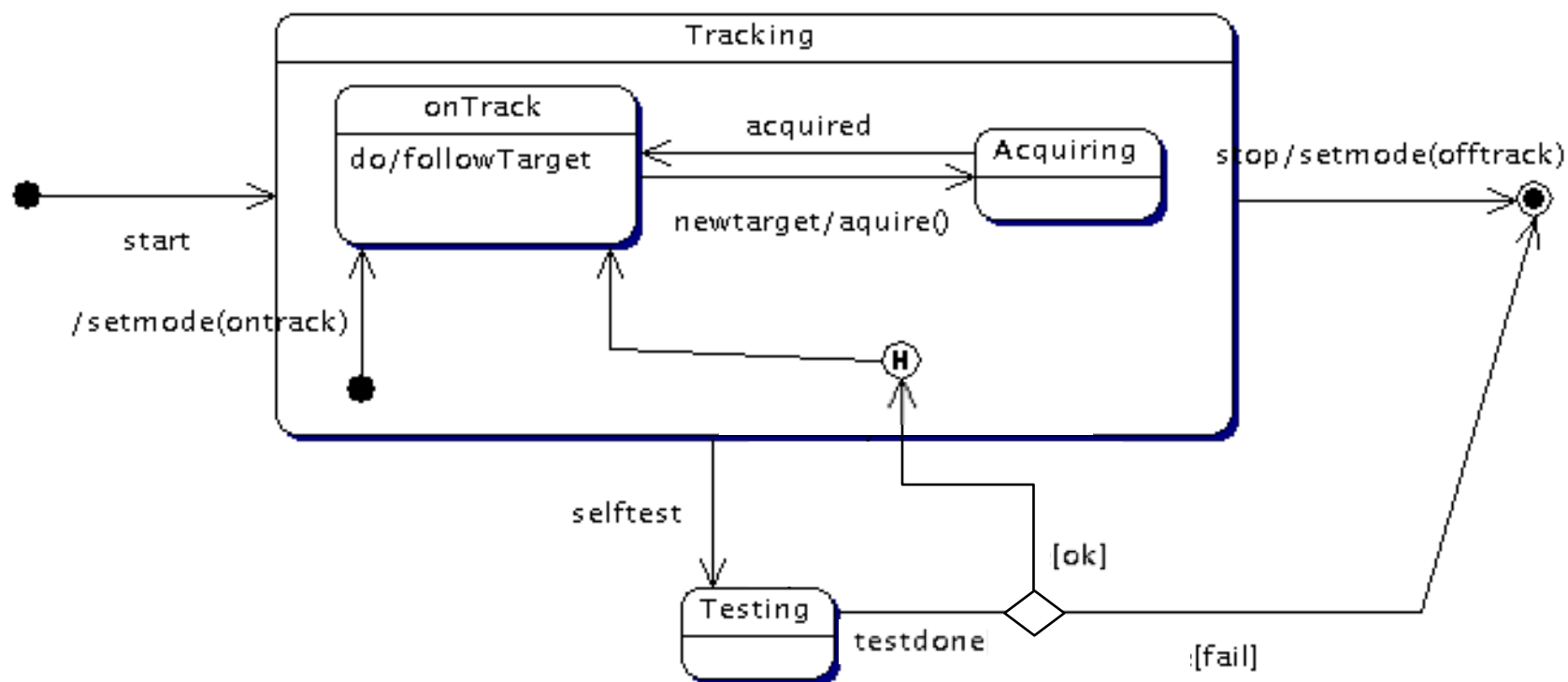
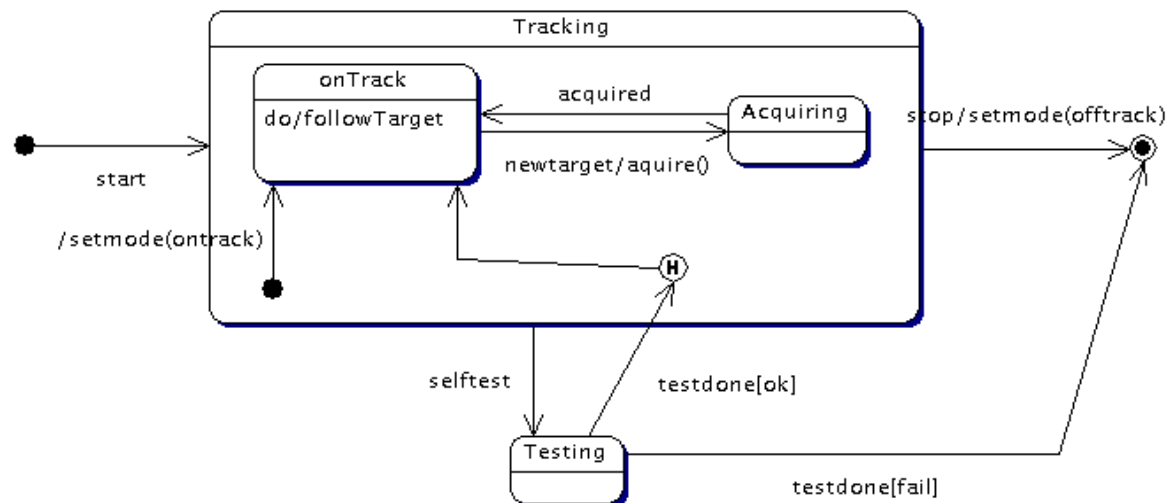
## Pseudoestado de Escolha

- Ramificação condicional (dinâmica!) em função do valor de uma expressão.
- Decisão pode ser uma função de acções anteriores.
- Caso mais que uma guarda verdadeira, a escolha é não determinística.
- Se nenhuma guarda for verdadeira, o modelo está mal formado ([else]!)

Desenvolvimento de Sistemas Software

- pseudoestado de Escolha

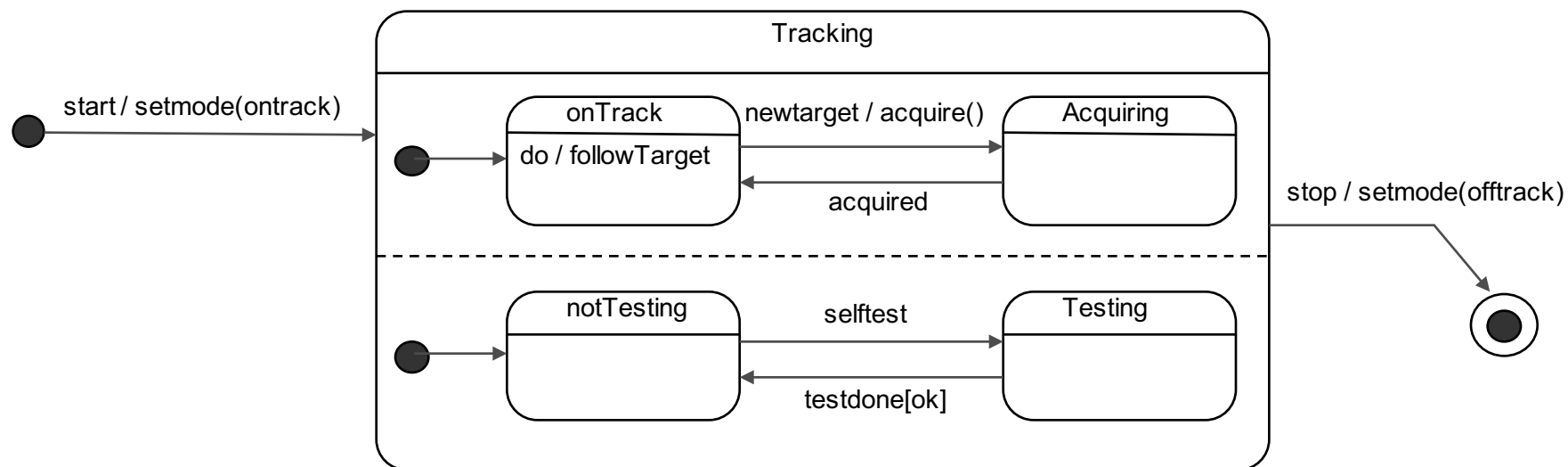




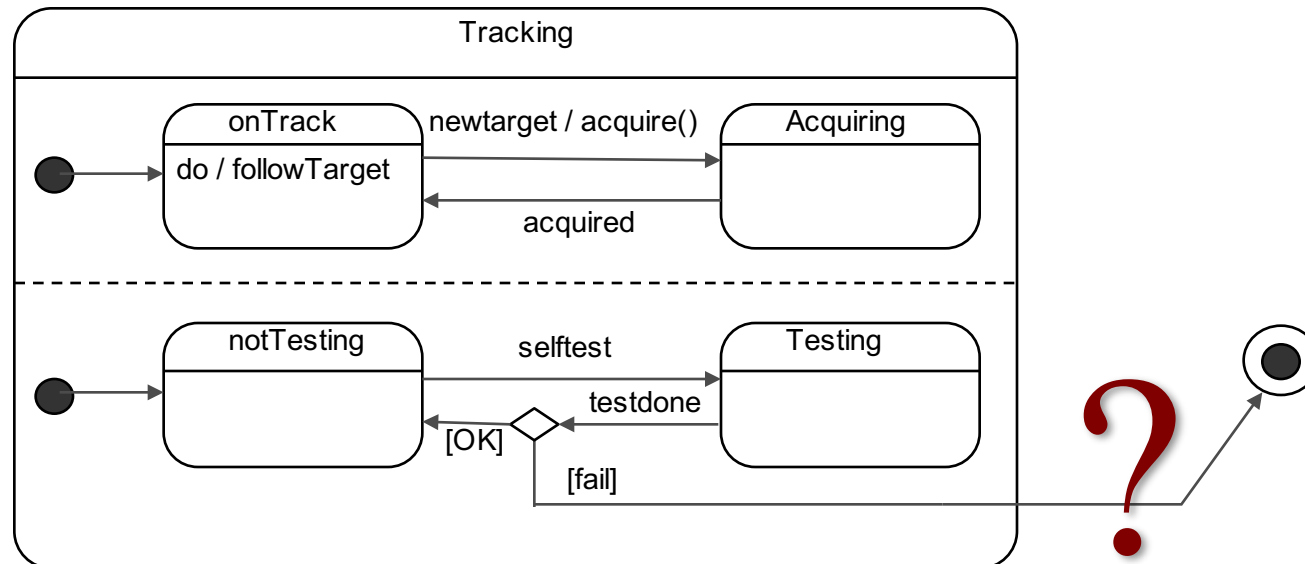


## Estados com concorrência...

- Um estado pode ser dividido em “regiões” ortogonais
- Cada região contém um sub-diagrama
- Os diagramas das regiões são *executados* de forma concorrente



# Pseudoestado de terminação

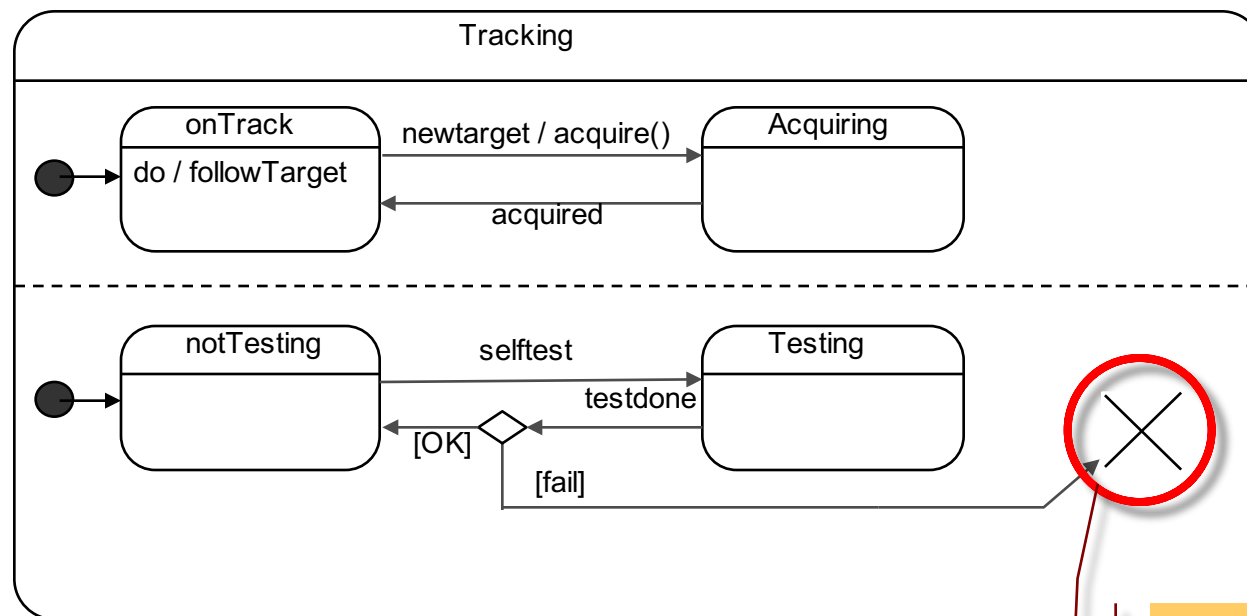






# Pseudoestado de terminação

- Indica que a execução da máquina de estados termina.
- Não são executadas acções de saída a não ser as da transição para o pseudoestado de terminação

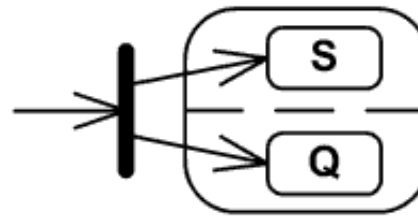


pseudoestado de  
Terminação

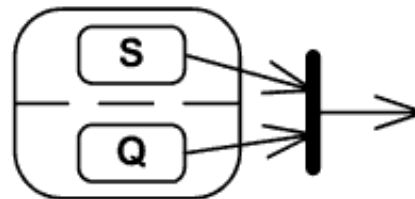


## Pseudoestados *fork* e *join*

- Permitem gerir concorrência.
- Fork - divide uma transição de entrada em duas ou mais transições
  - Transições de saída têm que terminar em regiões ortogonais distintas

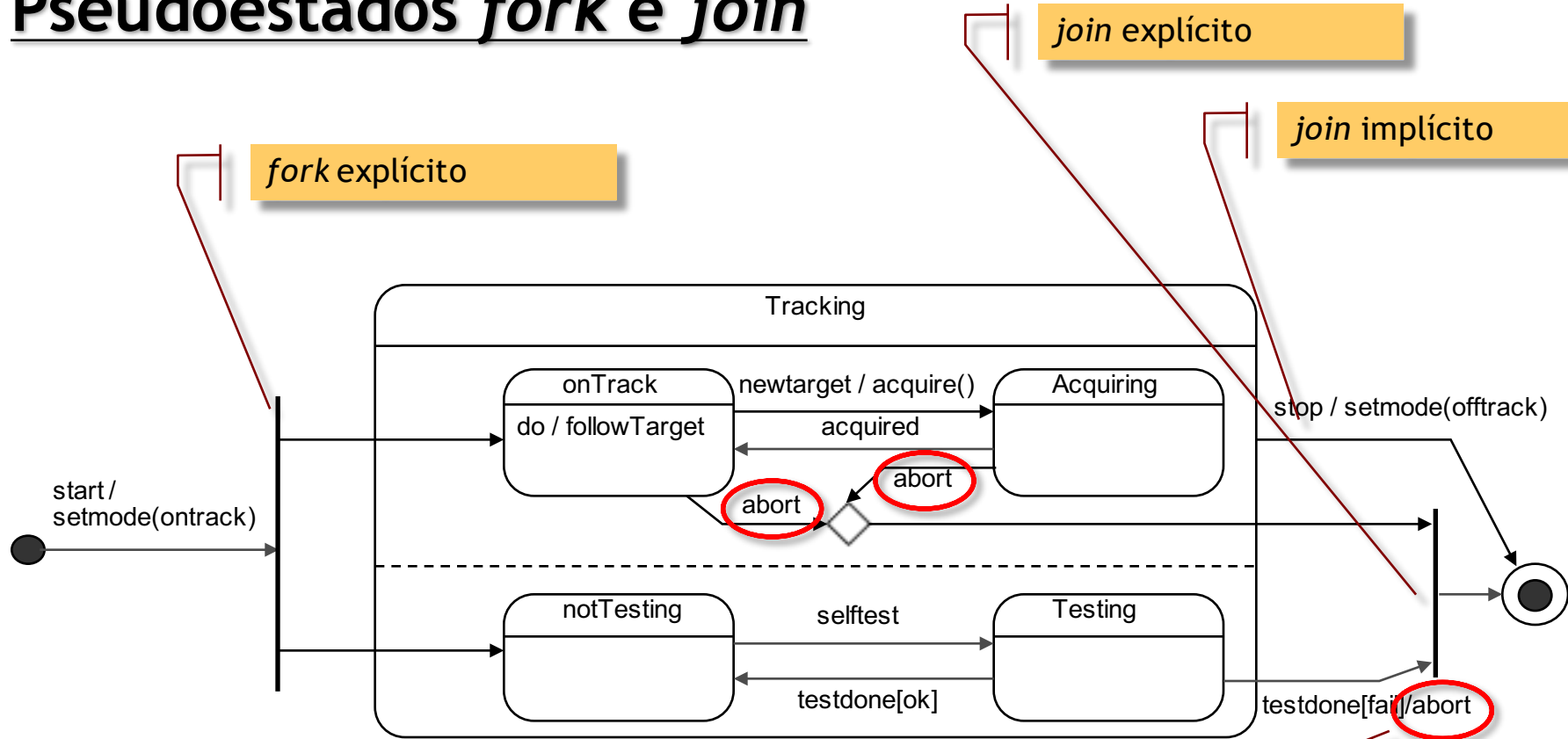


- Join - funde duas ou mais transições de entrada numa só transição de saída
  - Transições de entrada têm que originar em regiões ortogonais distinta





# Pseudoestados *fork* e *join*

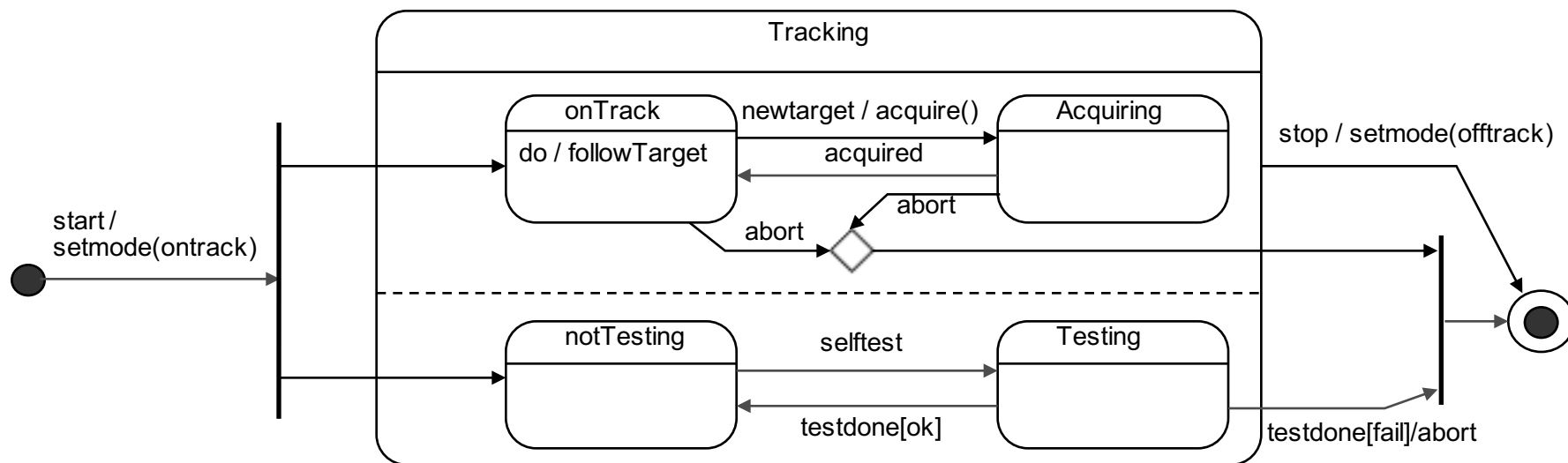


señal de *abort* utilizado para sincronizar con a outra sub-máquina





## Pontos de entrada e saída

- Como fazer para "esconder os detalhes" do estado Tracking?
- Transição a partir de sub-estados levantam problemas...





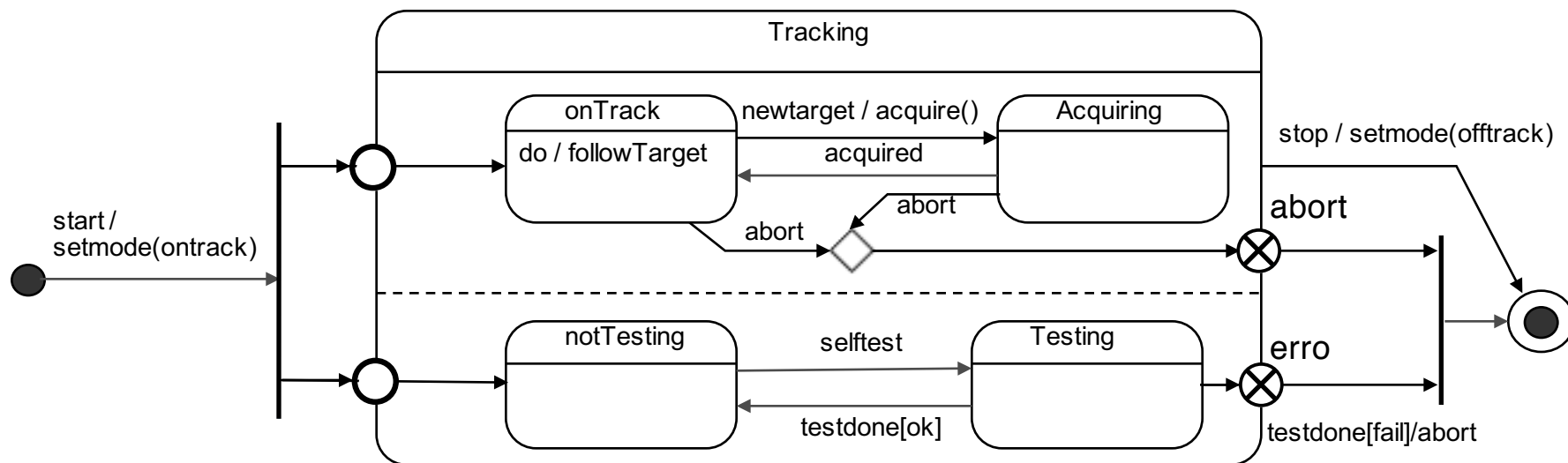
# Pseudoestados Ponto de entrada e Ponto de saída

- Ponto de entrada 
  - Permite definir um ponto de entrada numa máquina de estados ou num estado composto
  - O ponto de entrada é identificado por nome
  - O ponto de entrada transita para um estado interno que poderá ser diferente do definido pelo estado inicial
- Ponto de saída 
  - Permite definir um ponto de saída alternativo ao estado final
  - O ponto de saída é identificado por nome



## Pontos de entrada e saída

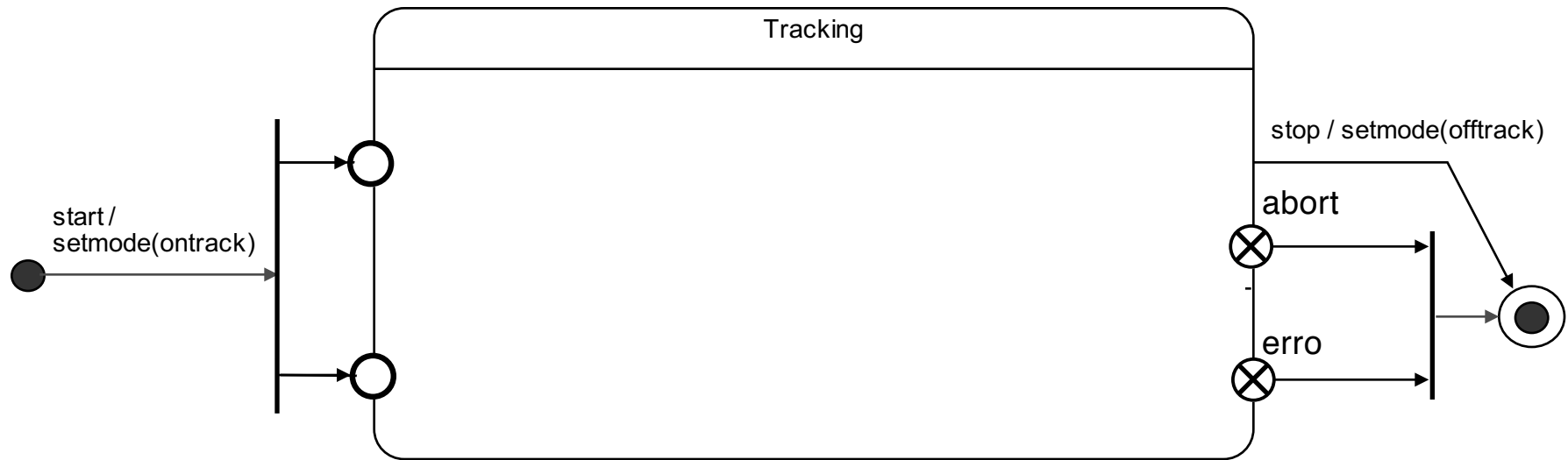
- Como fazer para "esconder os detalhes" do estado Tracking?
- Transição a partir de sub-estados levantam problemas...





## Pontos de entrada e saída

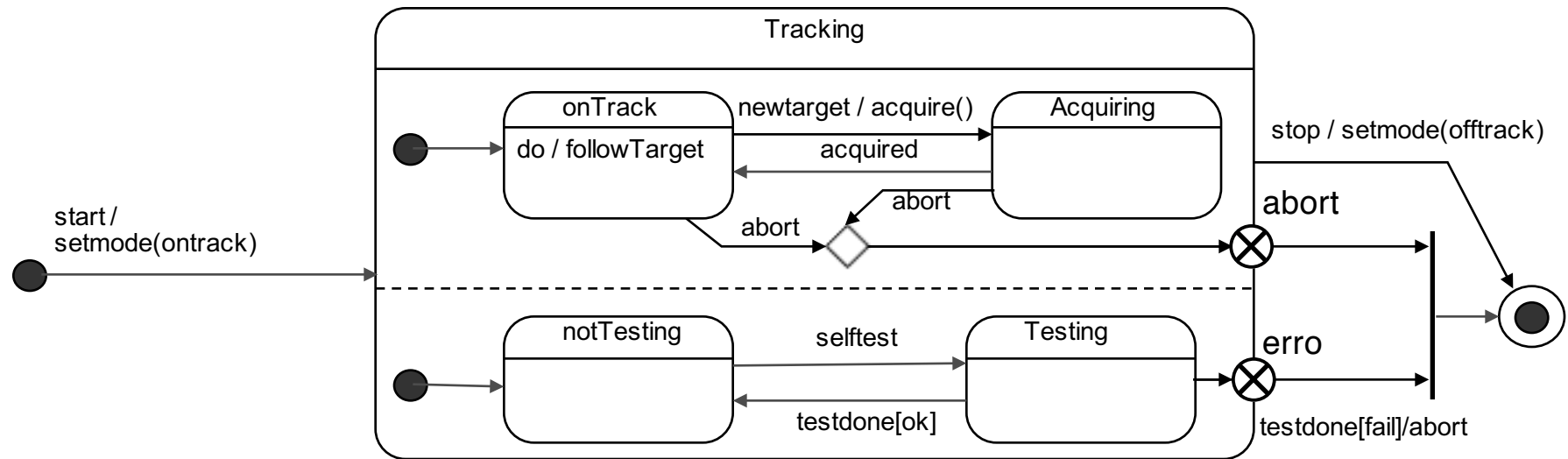
- Como fazer para "esconder os detalhes" do estado Tracking?
- Transição a partir de sub-estados levantam problemas...





## Pontos de entrada e saída

- Como fazer para "esconder os detalhes" do estado Tracking?
- Transição a partir de sub-estados levantam problemas...

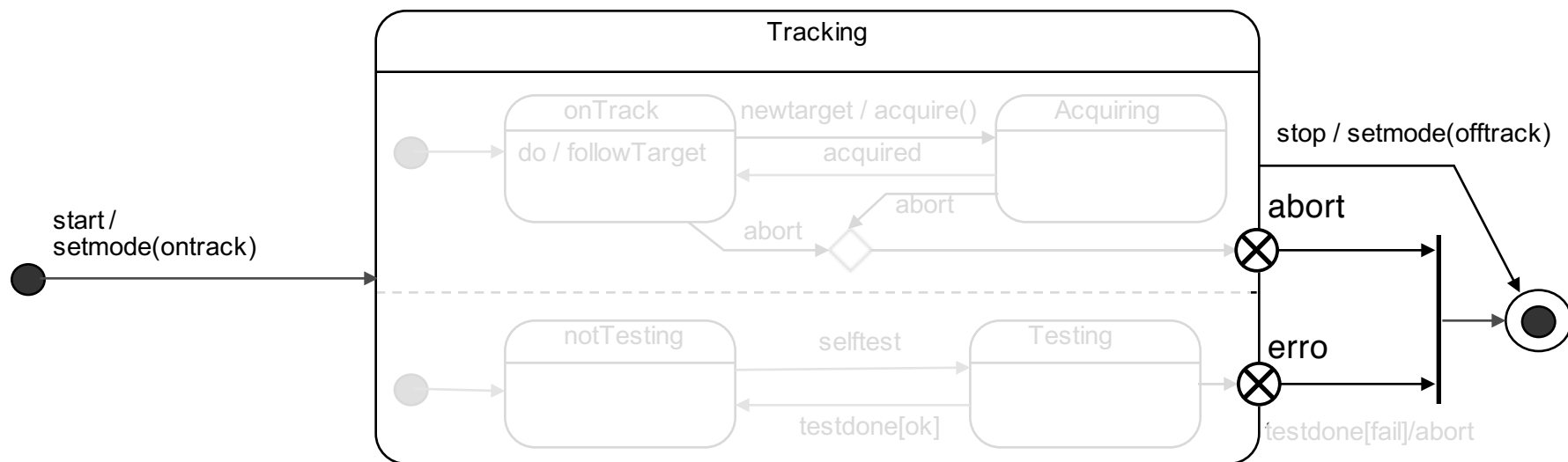






## Pontos de entrada e saída

- Como fazer para "esconder os detalhes" do estado Tracking?
- Transição a partir de sub-estados levantam problemas...



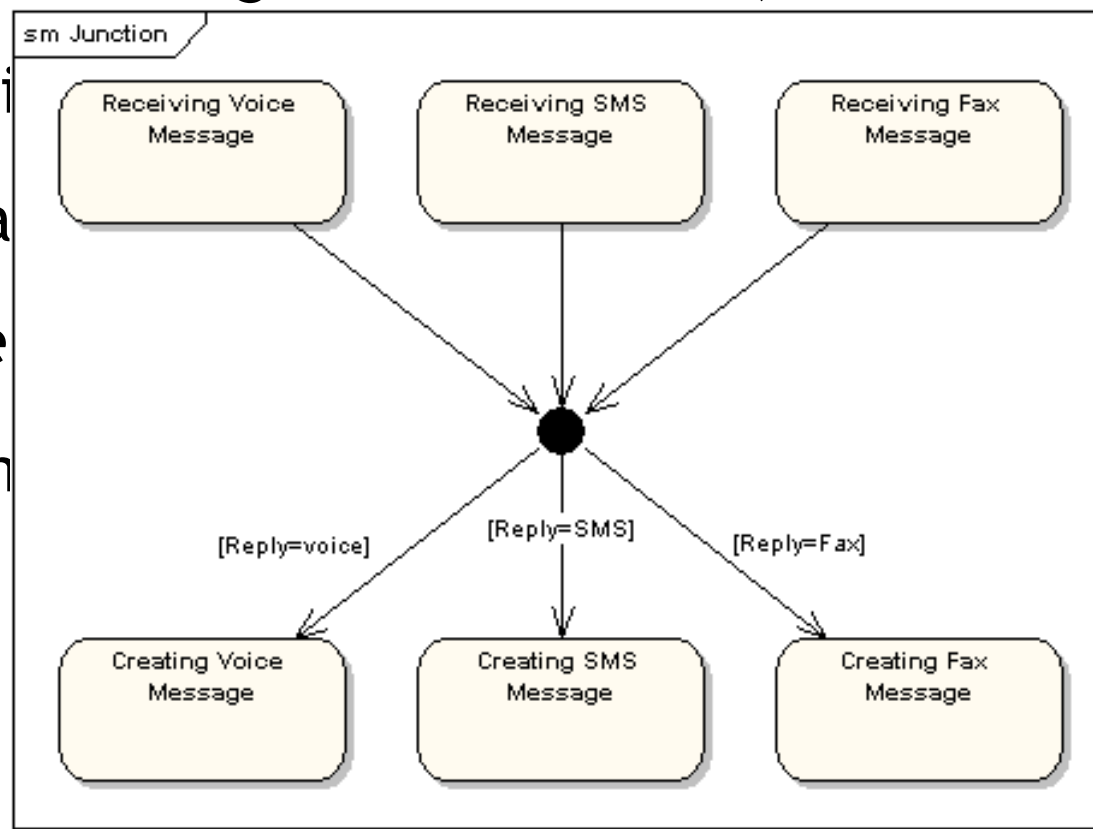


## Pseudo-estado de Junção

- Ramificação condicional (estática!) em função do valor de uma expressão.
- Caso mais que uma guarda verdadeira, a escolha é não determinística.
- Se nenhuma guarda for verdadeira, o modelo está mal formado ([else]!)
- Útil para simplificar diagramas, factorizando transições

## Pseudo-estado de Junção

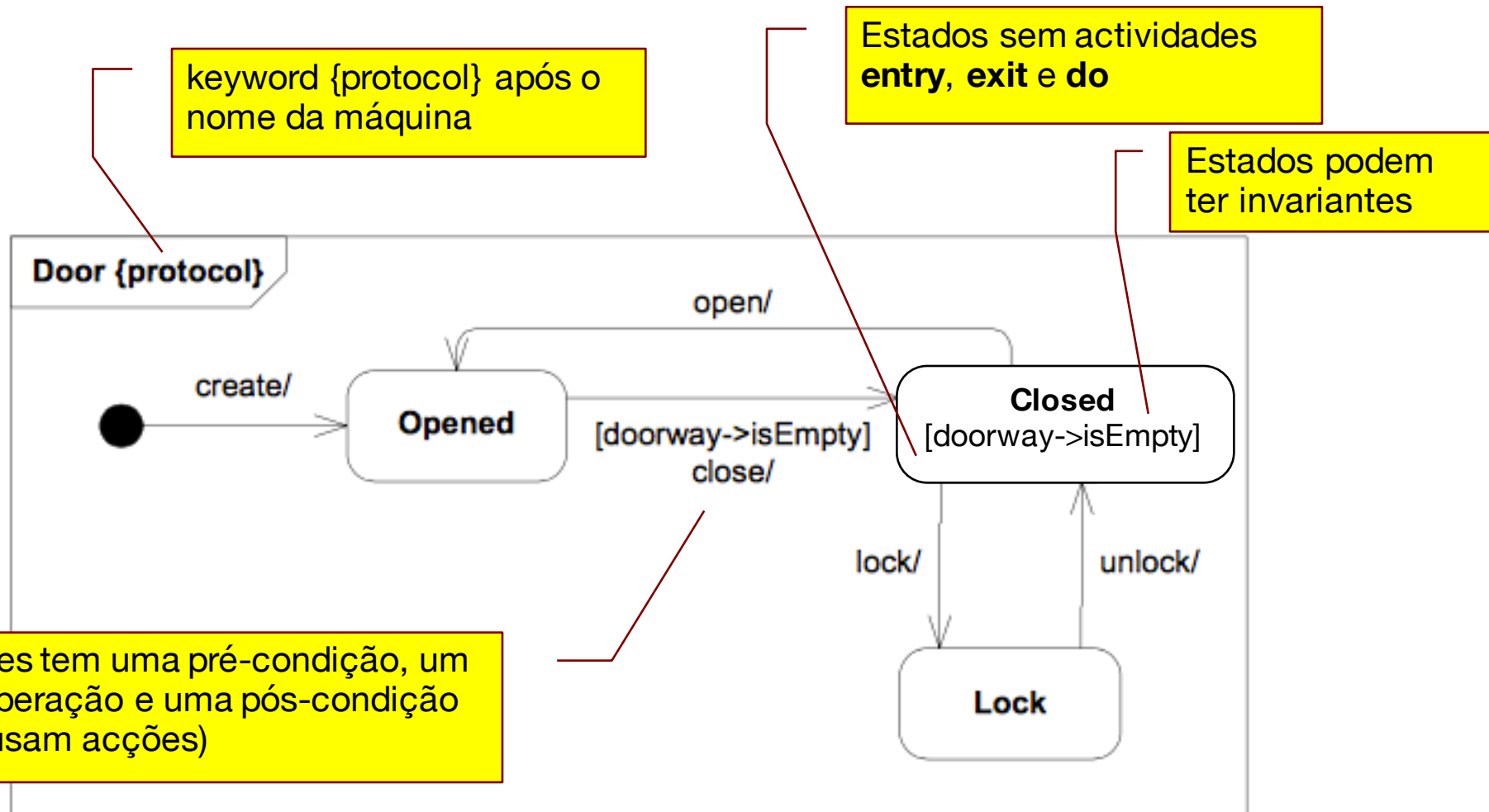
- Ramificação condicional (estática!) em função do valor de uma expressão.
- Caso mais que uma guarda verdadeira, a escolha é não determinística
- Se nenhuma guarda verdadeira, o estado é formado ([e])
- Útil para simular





# Protocol State Machines

- Especificam que operações podem ser invocadas em cada estado e em que condições - a sequências válidas de invocação das operações.





# Diagramas de Estado (*Statecharts*)

## Sumário

- Mais sobre transições
- Transições vs. actividades internas
- Regiões concorrentes
- Mais pseudoestados: História, Escolha, *Fork*, *Join*, Terminação, Pontos de Entrada e Saída, Junção