# Capítulo 2
# Máquinas de Estado

2.1 Introdução

2.2 Estruturas das máquinas de estado

2.3 Máquina de estados finitos

2.4 Máquina de estados não determinísticos

2.5 Equivalência de máquinas de estados

2.6 Composição de máquinas de estados

# 2.1 Introdução

- Existe uma classe bastante larga de sistemas (modelos de espaço de estados) que pode ser caracterizada de acordo com o conceito de estado e com a ideia de que um sistema evolui através de uma sequência de mudanças ou transições de estado

- Um modelo de espaço de estados descreve o sistema através de um procedimento, dando a sequência de operações, passo-a-passo, para a evolução do sistema. Mostra como o sinal de entrada provoca alterações no estado e como o sinal de saída é produzido

- Estes sistemas operam tipicamente sobre fluxos de eventos (*event streams*), realizando uma lógica de controlo

# 2.2 Estruturas das máquinas de estado



- Descrição de um sistema como função envolve 3 entidades:
  - Sinais de entrada
  - Sinais de saída
  - A função de actualização do sinal de saída em resultado do sinal de entrada

$$F : SinalEntrada \rightarrow SinalSaída$$

- Para uma máquina de estados temos

$$FluxoEventos : Naturais_0 \rightarrow Símbolos$$

  - Símbolos é um conjunto arbitrário
  - O domínio destes sinais representa ordem mas não necessariamente tempo (sabemos que um evento ocorre antes ou depois de outro mas não sabemos qual o tempo entre eventos)
- Uma máquina de estados constrói o sinal de saída através de um símbolo num dado instante observando o símbolo de entrada num dado instante

# 2.2 Estruturas das máquinas de estado

- Definição de Máquina de Estados
  - Uma Máquina de Estados é um 5-tuple

$$MáquinaEstados = (Estados, Entradas, Saídas, Actualização, EstadoInicial)$$

    - *Estados* é o conjunto do espaço de estados
    - *Entradas* é o conjunto do alfabeto de entrada
    - *Saídas* é o conjunto do alfabeto de saída
    - $EstadoInicial \in Estados$
    - Regra de actualização

$$Actualização : Estados \times Entradas \rightarrow Estados \times Saídas$$

  - Denominado Modelo de Funções e Conjuntos

# 2.2 Estruturas das máquinas de estado

- Entradas e Saídas
  - Os conjuntos *Entradas* e *Saídas* são os conjuntos de todos os possíveis símbolos de entrada e de saída
  - O conjunto de sinais de entrada (saída) consiste de todas as sequências infinitas de símbolos de entrada (saída)

$$SinaisEntrada : [Naturais_0 \rightarrow Entradas]$$

$$SinaisSaída : [Naturais_0 \rightarrow Saídas]$$

  - Seja $x \in SinaisEntrada$ o sinal de entrada. Um dado símbolo nesse sinal pode ser escrito $x(n)$ para qualquer $n \in N_0$. A sequência de entrada pode ser escrita como

$$(x(0), x(1), \ldots, x(n), \ldots)$$

  - O índice $n$ refere-se ao número do passo e não ao tempo

# 2.2 Estruturas das máquinas de estado

- Regra de actualização
  - A regra de actualização torna possível que a máquina de estados construa o sinal de saída passo a passo pela observação passo a passo do sinal de entrada
  - Sendo $s(n) \in Estados$ o estado actual no passo $n$, e $x(n) \in Entradas$ o símbolo de entrada corrente, então o símbolo de saída corrente $y(n)$ e o próximo estado $s(n+1)$ é dado por

$$\forall n \geq 0 \ (s(n+1), y(n)) = Actualização(s(n), x(n))$$

    começando em $s(0)=EstadoInicial$
  - A máquina de estados define a função $F : SinaisEntrada \rightarrow SinaisSaída$ tal que para qualquer sinal de entrada $x \in SinaisEntrada$, o correspondente sinal de saída é dado por $y=F(x)$
  - Notemos que se o *EstadoInicial* muda, a função $F$ muda

# 2.2 Estruturas das máquinas de estado

- Cada avaliação da equação anterior é denominada **reacção** porque define a forma como a máquina de estados reage a um determinado símbolo de entrada

- Apenas um símbolo de saída é produzido por cada símbolo de entrada, não sendo necessário ter acesso a toda a sequência de entrada para produzir a saída

- Trata-se de um procedimento **causal** já que o estado seguinte e a saída actual dependem apenas do estado inicial e das entradas actual e passadas

# 2.2 Estruturas das máquinas de estado

– É por vezes conveniente decompor a *Actualização* em dois passos
  - Função para o estado seguinte

$$EstadoSeguinte : Estados \times Entradas \rightarrow Estados$$

  - Função de saída

$$Saída : Estados \times Entradas \rightarrow Saída$$

– Relacionando com a definição anterior

$$s(n+1) = EstadoSeguinte\big(s(n), x(n)\big)$$

$$y(n) = Saída\big(s(n), x(n)\big)$$

$$\big(s(n+1), y(n)\big) = Actualização\big(s(n), x(n)\big)$$

$$= \big(EstadoSeguinte\big(s(n), x(n)\big), Saída\big(s(n), x(n)\big)\big)$$

# 2.2 Estruturas das máquinas de estado

- *Stuttering (gaguejar)*
  - Uma máquina de estados por cada símbolo de entrada produz exactamente um símbolo de saída e uma mudança de estado (podendo mudar para o mesmo estado)
  - Portanto se não houver símbolo de entrada não há símbolo de saída nem mudança de estado
  - Por vezes torna-se necessário incluir um novo símbolo nos conjuntos de entrada e de saída denominado de **nulo** (*absent*)

$nulo \in Entradas$ e $nulo \in Saídas$

$$(s(n), nulo) = Actualização(s(n), nulo)$$

# 2.2 Estruturas das máquinas de estado

- Exemplo: Parquímetro
  - Parqueamento de 60 minutos
  - O parquímetro é alimentado com 10, 20 ou 50 cêntimos
  - O parquímetro mostra em minutos o tempo que falta para expirar
  - Cada cêntimo incrementa um minuto no tempo, até ao máximo de 60 minutos
  - Cada passagem de um minuto é assinalada
  - Quando o tempo restante é 0 o mostrador indica *Expirado*

# 2.2 Estruturas das máquinas de estado

- Modelo da máquina de estados
  - O conjunto de estados é

    $Estados = \{0,1,2,...,60\}$

  - O alfabeto de entrada e de saída são

    $Entradas = \{10c, 20c, 50c, minuto, nulo\}$

    $Saídas = \{Expirado, 1, 2, ..., 60, nulo\}$

  - Estado inicial

    $EstadoInicial = 0$

  - Função de actualização

    $Actualização : Estados \times Entradas \rightarrow Estados \times Saídas$

# 2.2 Estruturas das máquinas de estado

$$(s(n+1), y(n)) = \textit{Actualização}(s(n), x(n)) =$$

$$\begin{cases} (0, \textit{Expirado}) & \textit{se } x(n) = \textit{minuto} \wedge (s(n) = 0 \vee s(n) = 1) \\ (s(n)-1, s(n)-1) & x(n) = \textit{minuto} \wedge s(n) > 1 \\ (\min(s(n)+10,60), \min(s(n)+10,60)) & x(n) = 10c \\ (\min(s(n)+20,60), \min(s(n)+20,60)) & x(n) = 20c \\ (\min(s(n)+50,60), \min(s(n)+50,60)) & x(n) = 50c \\ (s(n), \textit{nulo}) & x(n) = \textit{nulo} \end{cases}$$

- Qual é a sequência de saída se a sequência de entrada for *(20c,minutox15,10c,minutox10,10c,minutox25,...)*?
- *(Expirado,20,19,...,6,5,15,14,...,6,5,15,14,...,Expirado,Expirado,...)*

# Capítulo 2
# Máquinas de Estado

# 2.3 Máquinas de estados finitos

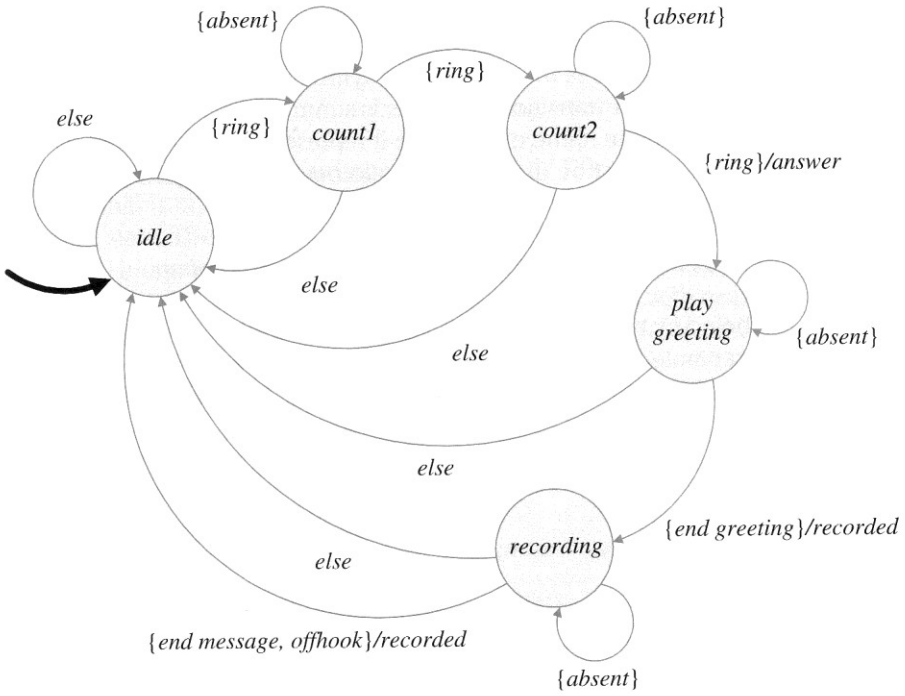
- Nas máquinas de estados finitos (*Finite State Machines* - FSM) o conjunto de estados é finito

- Estas máquinas possuem técnicas analíticas poderosas dado que é possível explorar todas as possíveis sequências de estados
- O alfabeto de entrada e de saída destas máquinas permite representar uma grande variedade de situações

- Quando o número de estados é reduzido, quando os alfabetos de entrada e de saída são finitos e pequenos, podemos descrever a máquina de estados através de um **diagrama de transição de estados**

# 2.3 Máquinas de estados finitos

– Exemplo de um diagrama de transição de estados



# 2.3 Máquinas de estados finitos



- Diagrama de transição de estados
  - Cada bola representa um estado
  - Cada arco representa uma transição de um estado para outro
  - As bolas e os arcos são anotadas/etiquetadas
  - Cada bola é etiquetada com o nome do estado que representa
  - Cada arco é etiquetado pela entrada e pela saída
  - A etiqueta da entrada especifica que símbolo de entrada faz disparar a transição que lhe está associada
  - O símbolo de saída indicado é produzido como parte da transição entre estados (se não for indicado o símbolo de saída é produzido o símbolo nulo)
  - A execução de uma máquina de estados consiste numa sequência de reacções, na qual cada reacção envolve a transição de um estado para outro ao longo dos arcos

# 2.3 Máquinas de estados finitos



- Existe uma notação especial para certos arcos, **arco *else***, que são percorridos quando nenhum dos outros arcos saindo de um dado estado é percorrido (representando todas as outras possíveis entradas)
- Se este arco não estiver representado, e houver entradas não representadas, então por omissão a transição efectua-se para o próprio estado
- O estado inicial é indicado no diagrama
- A execução começa através do estado inicial. Partindo daí cada vez que chega um símbolo de entrada a máquina de estados reage, transitando para o estado seguinte de acordo com as etiquetas de entrada associadas a cada transição e produzindo um símbolo na saída
- Uma sequência de símbolos de entrada gera uma sequência de estados
- Sabendo a sequência de transições também sabemos a sequência de símbolos de saída

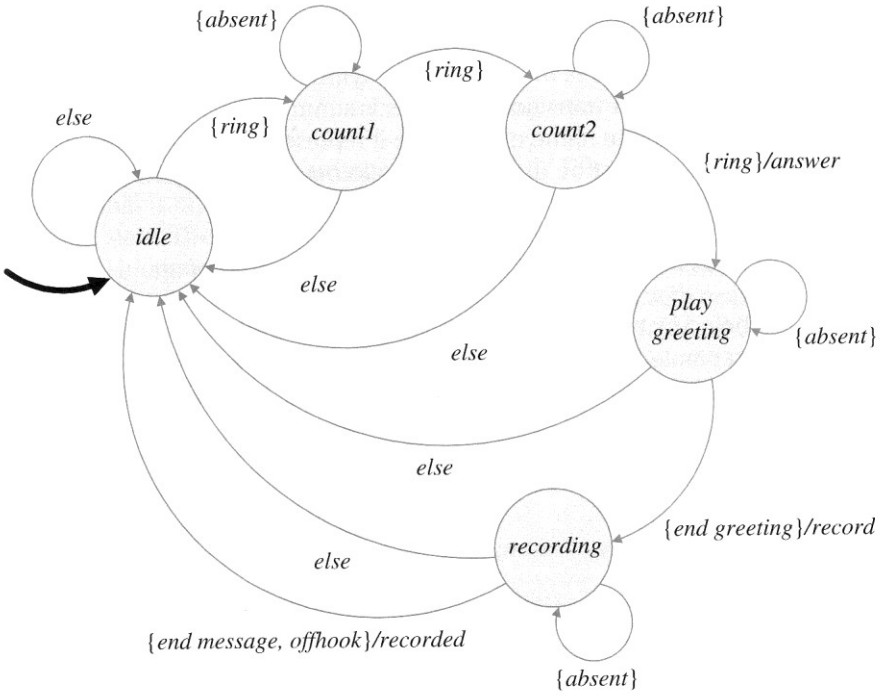
# 2.3 Máquinas de estados finitos



- Exemplo: Telefone com atendimento automático
  - Quando chega uma chamada o telefone toca
  - Se não atenderem até ao 3º toque a máquina responde; se atenderem a máquina não realiza nenhuma acção
  - Toca uma mensagem de boas vindas
  - Grava a mensagem
  - Depois da gravação desliga-se automaticamente

  - *Estados={idle,count1,count2,play greeting,recording}*
  - *Entrada={ring,offhook,end greeting,end message,absent}*
  - *Saídas={answer,record,recorded,absent}*

# 2.3 Máquinas de estados finitos




| States |
|---|
| *idle* — nothing is happening<br>*count1* — one ring has arrived<br>*count2* — two rings have arrived<br>*play greeting* — playing the greeting message<br>*recording* — recording the message |

| Inputs |
|---|
| *ring* — incoming ringing signal<br>*offhook* — a telephone extension is picked up<br>*end greeting* — greeting message is finished playing<br>*end message* — end of message detected (e.g., dialtone)<br>*absent* — no input of interest |

| Outputs |
|---|
| *answer* — answer the phone and start the greeting message<br>*record* — start recording the incoming message<br>*recorded* — recorded an incoming message<br>*absent* — default output when there is nothing interesting to say |

$$idle \xrightarrow{ring} count1 \xrightarrow{ring} count2 \xrightarrow{offhook} idle \ldots$$

$$idle \xrightarrow{ring} count1 \xrightarrow{ring} count2 \xrightarrow{ring} play\ greeting \xrightarrow{end\ greeting} recording \xrightarrow{end\ message} idle \ldots$$

# 2.3 Máquinas de estados finitos



- A máquina de estados deve sempre reagir a um símbolo de entrada - **propriedade da receptividade**
  - Com a introdução do arco *else* as máquinas de estado são sempre **receptivas**

- Uma mesma entrada poderá estar atribuída a mais de uma transição do estado corrente
  - A máquina é livre de escolher qual a transição a seguir entre aquelas que apresentam o mesmo símbolo de entrada
  - Mais do que um comportamento é admissível para a máquina
  - Uma máquina de estados com esta característica é denominada **não determinística**

# 2.3 Máquinas de estados finitos

- Tabela de actualização
  - Forma alternativa de representar uma máquina de estados
  - Representação tabular do diagrama de transição de estados

| Current state | (*next state*, *output symbol*) Under specified input symbol | | | | |
|---|---|---|---|---|---|
| | *ring* | *offhook* | *end greeting* | *end message* | *absent* |
| *idle* | (*count1*, *absent*) | (*idle*, *absent*) | (*idle*, *absent*) | (*idle*, *absent*) | (*idle*, *absent*) |
| *count1* | (*count2*, *absent*) | (*idle*, *absent*) | (*idle*, *absent*) | (*idle*, *absent*) | (*count1*, *absent*) |
| *count2* | (*play greeting*, *answer*) | (*idle*, *absent*) | (*idle*, *absent*) | (*idle*, *absent*) | (*count2*, *absent*) |
| *play greeting* | (*idle*, *absent*) | (*idle*, *absent*) | (*recording*, *record*) | (*idle*, *absent*) | (*play greeting*, *absent*) |
| *recording* | (*idle*, *absent*) | (*idle*, *recorded*) | (*idle*, *absent*) | (*idle*, *recorded*) | (*recording*, *absent*) |

# 2.3 Máquinas de estados finitos



- Máquinas de Mealy - símbolo de saída produzido durante a transição do estado
- Máquinas de Moore - símbolo de saída produzido enquanto a máquina se encontra no estado
- As máquinas de Mealy tornam-se mais úteis quando se realiza uma operação de composição síncrona

# 2.3 Máquinas de estados finitos

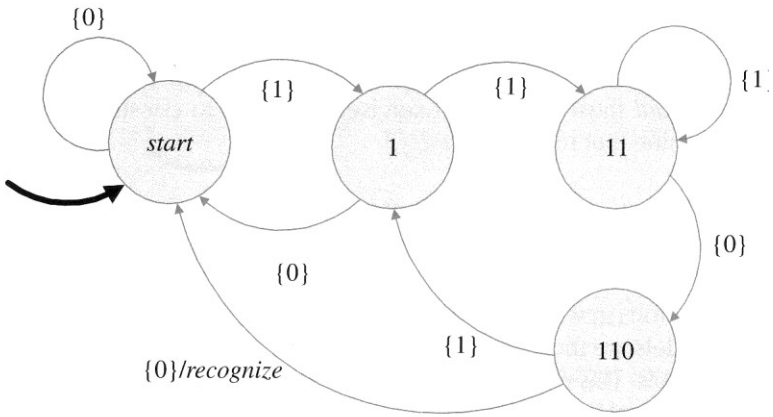
- Temos 3 formas de descrever uma máquina de estados:
  - a descrição dos conjuntos e funções, o diagrama de transição de estados e a tabela de actualizações
  - A tabela e o diagrama de transições só podem ser usados se os conjuntos (Estados, Entradas e Saídas) são finitos
  - A descrição através de conjuntos e funções é igualmente possível com espaços de estados finitos ou infinitos
- Sistema **determinado pelo estado**
  - Conhecendo o estado actual, podemos determinar o comportamento futuro do sistema para qualquer símbolo de entrada futuro
  - O estado actual sumaria a história do sistema
- Por vezes pretendemos saber se um dado problema pode ser realizado através de uma máquina de estados
  - Se o número de padrões necessários para descrever o sistema é finito consegue-se, usando um número de estados igual ao número de padrões, realizar através de uma máquina de estados

# 2.3 Máquinas de estados finitos

– Exemplo: Reconhecedor de código

- Reconhecedor de código em que os sinais de entrada e de saída são sequências de 0 e 1
- O sistema deverá produzir na saída *Recognize* sempre que detectar uma sequência 1100 na entrada

$$y(n) = \begin{cases} recognize & if\,(x(n-3), x(n-2), x(n-1), x(n)) = (1,1,0,0) \\ nulo & outros\,casos \end{cases}$$



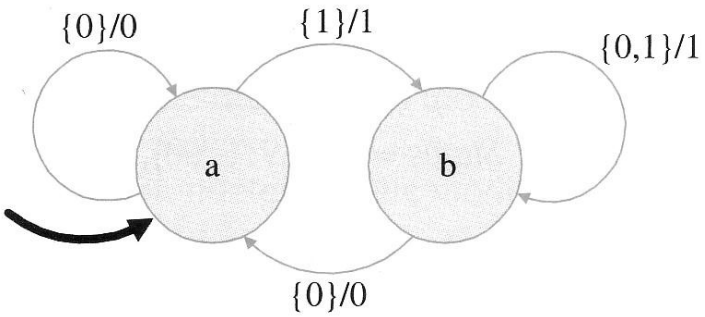
# Capítulo 2
# Máquinas de Estado

# 2.4 Máquinas de estados não determinísticas

- Uma máquina de estados não determinística é um modelo incompleto de um sistema, mas mais compacto e cuja informação poderá ser suficiente para analisar o comportamento do sistema
- Numa máquina determinística as entradas que etiquetam as transições de um dado estado são mutuamente exclusivas, ou seja não partilham símbolos comuns
- Nas máquinas não determinísticas pode haver sobreposição entre as etiquetas que representam as entradas
  - Um símbolo de entrada pode aparecer a etiquetar mais de uma transição do mesmo estado, o que significa que uma ou mais transições podem ser realizadas
  - É esta característica que torna a máquina não determinística

# 2.4 Máquinas de estados não determinísticas

- Exemplo
  - Estado inicial *a*
  - Transita para o estado *b* da primeira vez que encontra um 1
  - Fica no estado *b* um tempo arbitrário; se receber um 1 mantém-se no estado *b*, se receber 0 pode manter-se no *b* ou transitar para o *a*
  - A sequência de entrada (0,1,0,1,0,1,...)




gera ...

(a,a,b,a,b,a,b,...) (0,1,0,1,0,1,...)

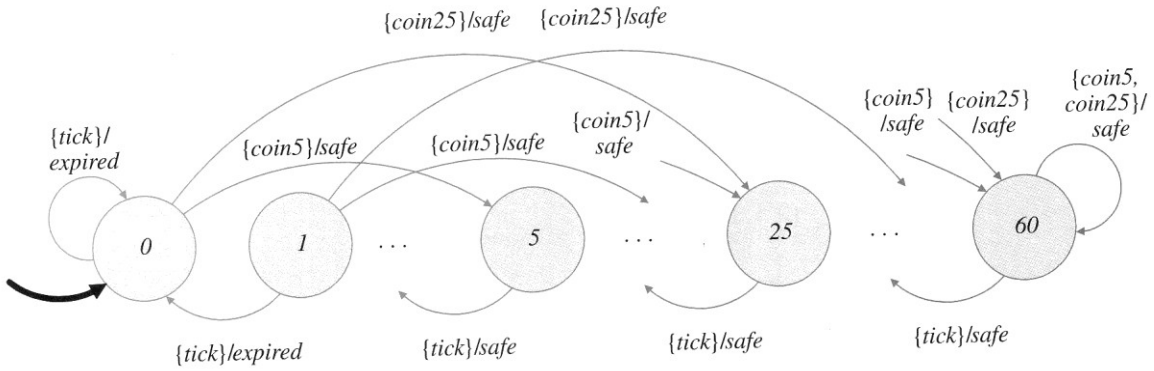
(a,a,b,b,b,a,b,...) (0,1,1,1,0,1,...)

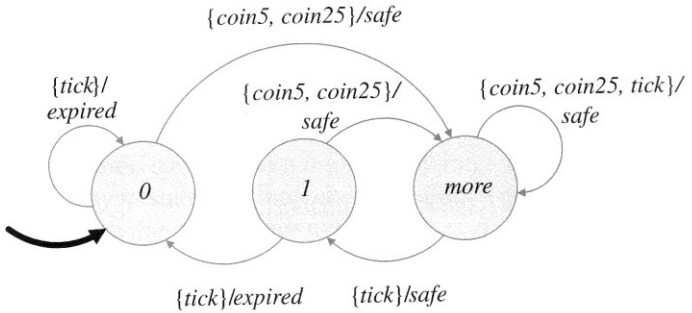
(a,a,b,a,b,b,b,...) (0,1,0,1,1,1,...)

(a,a,b,b,b,b,b,...) (0,1,1,1,1,1,...)

# 2.4 Máquinas de estados não determinísticas

- Exemplo: parquímetro



# 2.4 Máquinas de estados não determinísticas

- Função de actualização
  - Nas máquinas determinísticas se conhecemos o estado inicial e a sequência de entrada então toda a trajectória de estados e a sequência de saída pode ser determinada
    - *s(n)* e *x(n)* determinam univocamente *s(n+1)* e *y(n)*
  - Numa máquina não-determinística o estado seguinte não é completamente determinado pelo estado actual e o símbolo de entrada
    - Não a podemos caracterizar pela função *Actualização(s(n),x(n))* porque não há apenas um estado seguinte
  - Determinística $Actualização : Estados \times Entradas \rightarrow Estados \times Saídas$
  - Não determinística

$$PossíveisActualizações : Estados \times Entradas \rightarrow P(Estados \times Saídas)$$

onde $P(Estados \times Saídas)$ representa todos os subconjuntos possíveis

# 2.4 Máquinas de estados não determinísticas



- Para uma máquina de estados ser receptiva é necessário que

$$\forall s(n) \in Estados \;\; e \;\; \forall x(n) \in Entradas \;\; PossíveisActualizações\big(s(n), x(n)\big) \neq \phi$$

- Uma máquina de estados não determinística selecciona arbitrariamente o estado seguinte e o símbolo de saída corrente das *PossíveisActualizações* dado o estado corrente e o símbolo de entrada
- Nada é dito como a selecção é realizada
- Definição da máquina de estados não determinística num 5-tuple

$$MáquinaEstados = \big(Estados, Entradas, Saídas, PossíveisActualizações, EstadoInicial\big)$$

- Uma máquina determinística é um caso especial de uma não determinística

$$PossíveisActualizações\big(s(n), x(n)\big) = \big\{Actualização\big(s(n), x(n)\big)\big\}$$

# 2.4 Máquinas de estados não determinísticas

- Comportamento
  - Definimos o comportamento de uma máquina como o par *(x,y)* tal que *y=F(x)*, ou seja o possível par entrada-saída
  - Definimos o conjunto

$$Comportamentos \subset SinaisEntrada \times SinaisSaída$$

onde

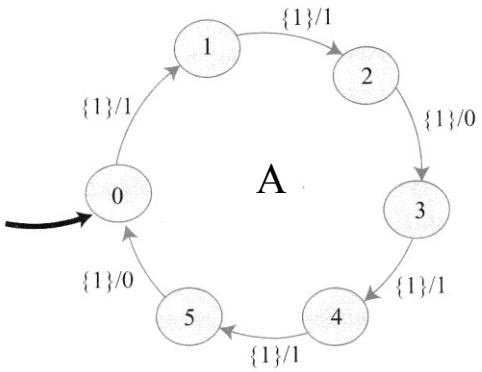
$$Comportamentos = \{(x,y) \in SinaisEntrada \times SinaisSaída \mid y \text{ é uma sequência de saída possível para a sequência de entrada } x\}$$

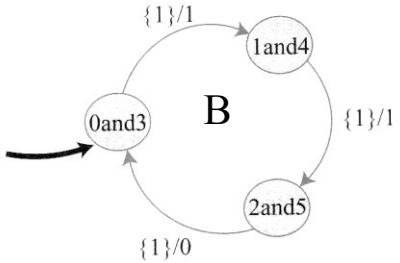
  - Máquina determinística para cada $x \in SinaisEntrada$ existe um $y \in SinaisSaída$ tal que *(x,y)* é um comportamento
  - Para uma máq. determ. o conjunto comportamento é o gráfico de *F*
  - Para uma máq. não deter. para um dado $x \in SinaisEntrada$ pode haver mais de um $y \in SinaisSaída$ tal que *(x,y)* é um comportamento

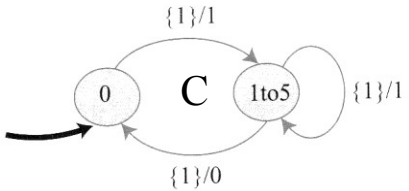
# Capítulo 2
# Máquinas de Estado

# 2.5 Equivalência de máquinas de estado

- Duas máquinas de estado diferentes, com os mesmos alfabetos, podem ser equivalentes no sentido em que para a mesma sequência de entrada produzem a mesma sequência de saída
- Exemplo:
  - 3 máquinas com o mesmo alfabeto de entrada e de saída
  - **A** e **B** produzem uma sequência alternada de 110 quando recebem uma sequência de 1's na entrada
  - **C** é não determinística, pode produzir qualquer sequência alternada de 1's (>0) e um 0



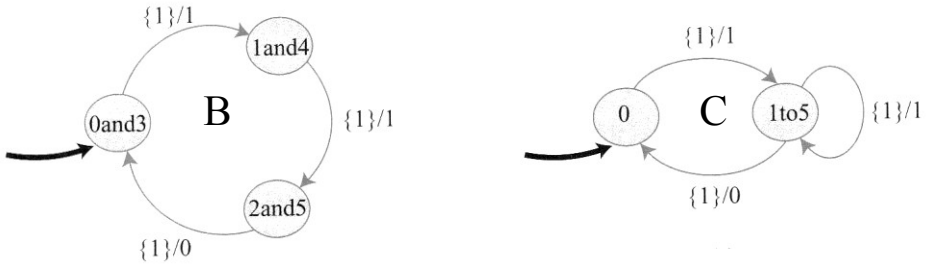



# 2.5 Equivalência de máquinas de estado

- Queremos mostrar que a máquina **C** simula **B** e **A**, e que a máquina **B** é equivalente a **A**
- Jogo da simulação com duas máquinas
  - As máquinas começam no mesmo estado inicial
  - A primeira máquina reage ao símbolo de entrada
    - Se a máquina for não determinística existe mais do que uma possível reacção
  - A segunda máquina reage ao mesmo símbolo de entrada de modo a produzir o mesmo símbolo de saída
    - Se a máquina for não determinística é livre de escolher das reacções possíveis qualquer uma que esteja de acordo com o símbolo de saída da primeira máquina e que permita continuar a gerar os mesmo símbolos de saída
  - A segunda máquina simula a primeira máquina se consegue sempre gerar um símbolo de saída igual ao da primeira máquina
- Se a relação de simulação ocorre em ambos os sentidos então as máquinas são equivalentes

# 2.5 Equivalência de máquinas de estado

– Exemplo: determinar se **C** simula **B**




- Máquinas nos estados iniciais

$$S_0=(0and3,0) \in Estados_B \times Estados_C$$

- A máquina **B** reage primeiro (está a ser simulada)

$$S_1=(1and4,1to5) \in Estados_B \times Estados_C$$

$$S_2=(2and5,1to5) \in Estados_B \times Estados_C$$

- Volta a $S_0 \Rightarrow$ **C** simula **B**
- A **relação de simulação** estabelece a correspondência entre as 2 máquinas

$$S_{B,C}=\{S_0,S_1,S_2\} \subset Estados_B \times Estados_C$$

– A simulação é usada para estabelecer uma relação entre um modelo mais abstracto e um modelo mais detalhado

# 2.5 Equivalência de máquinas de estado

- Estratégia de simulação
  - Suponha-se que temos duas máquinas **X** e **Y** definidas por

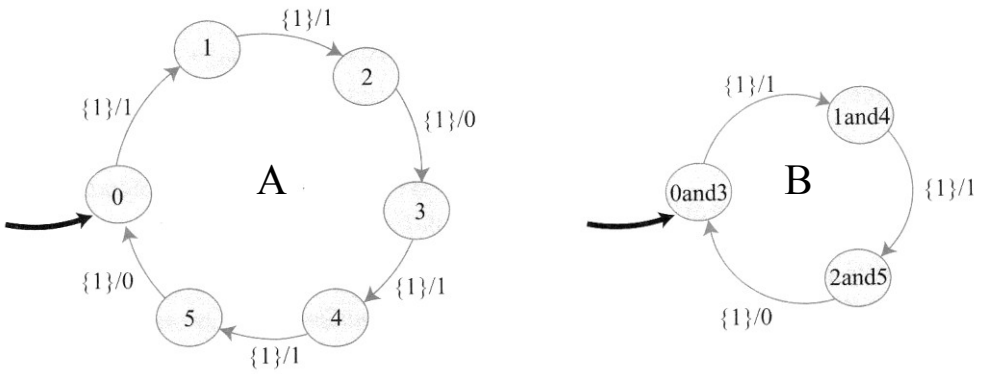
    $X=(Estados_X, Entradas, Saídas, PossíveisActualizações_X, EstadoInicial_X)$

    e

    $Y=(Estados_Y, Entradas, Saídas, PossíveisActualizações_Y, EstadoInicial_Y)$
  - **Y** simula **X** se existe um subconjunto $S \subset Estados_X \times Estados_Y$ tal que
    1. $(EstadoInicial_X, EstadoInicial_Y) \in S$
    2. Se $(s_X(n), s_Y(n)) \in S$, então $\forall x(n) \in Entradas$, e $\forall (s_X(n+1), y_X(n)) \in PossíveisActualizações_X(s_X(n), x(n))$, existe $(s_Y(n+1), y_Y(n)) \in PossíveisActualizações_Y(s_Y(n), x(n))$ tal que
       1. $(s_X(n+1), s_Y(n+1)) \in S$
       2. $y_X(n)=y_Y(n)$
  - O conjunto $S$ se existir é denominado a relação de simulação, estabelecendo uma correspondência entre os estados das duas máquinas

# 2.5 Equivalência de máquinas de estado

– Exemplo: determinar se **B** simula **A**




- A relação de simulação é o sub-conjunto

$$S_{A,B} \subset \{0,1,2,3,4,5\} \times \{0and3,1and4,2and5\}$$

- **A** reage primeiro (está a ser simulada)
- Sequência de estados:

$S_{A,B}=\{(0,0and3),(1,1and4),(2,2and5),(3,0and3),(4,1and4),(5,2and5)\}$

- Conclui-se que **B** simula **A** e é fácil de verificar que **A** simula **B**
- As máquinas **A** e **B** são equivalentes

# 2.5 Equivalência de máquinas de estado

– A relação de simulação é transitiva

- Mostrámos que **C** simula **B** e que **B** simula **A**

$$S_{B,C} \subset Estados_B \times Estados_C \qquad S_{A,B} \subset Estados_A \times Estados_B$$

então

$$S_{A,C}=\{(S_A,S_C) \in Estados_A \times Estados_C \mid \exists S_B \in Estados_B \text{ onde } (S_A,S_B) \in S_{A,B} \text{ e } (S_B,S_C) \in S_{B,C}\}$$

é a relação de simulação que mostra que **C** simula **A**

# 2.5 Equivalência de máquinas de estado



– Exemplo: aplicação da propriedade transitiva

- **B** simula **A**

  $S_{A,B}=\{(0,0and3),(1,1and4),(2,2and5),(3,0and3),(4,1and4),(5,2and5)\}$

- **C** simula **B**

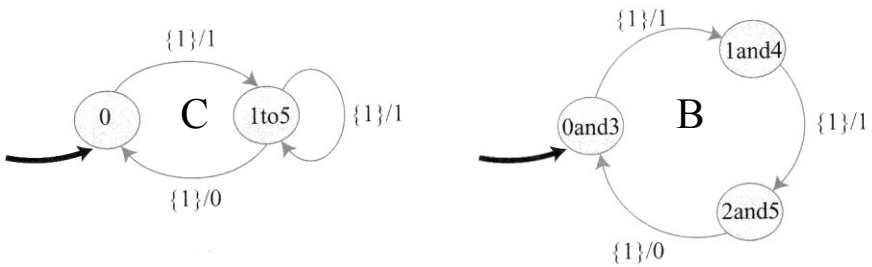
  $S_{B,C}=\{(0and3,0),(1and4,1to5),(2and5,1to5)\}$

- Podemos concluir que **C** simula **A**

  $S_{A,C}=\{(0,0),(1,1to5),(2,1to5),(3,0),(4,1to5),(5,1to5)\}$

# 2.5 Equivalência de máquinas de estado

– Exemplo: determinar se **B** simula **C**




- A relação de simulação é o sub-conjunto

$$S_{C,B} \subset \{0,1to5\} \times \{0and3,1and4,2and5\}$$

- **C** reage primeiro (está a ser simulada)
- Sequência de estados:

$$S_{C,B}=\{(0,0and3),(1to5,1and4),...\}$$

- Conclui-se que **B** não simula **C**

# Capítulo 2
# Máquinas de Estado

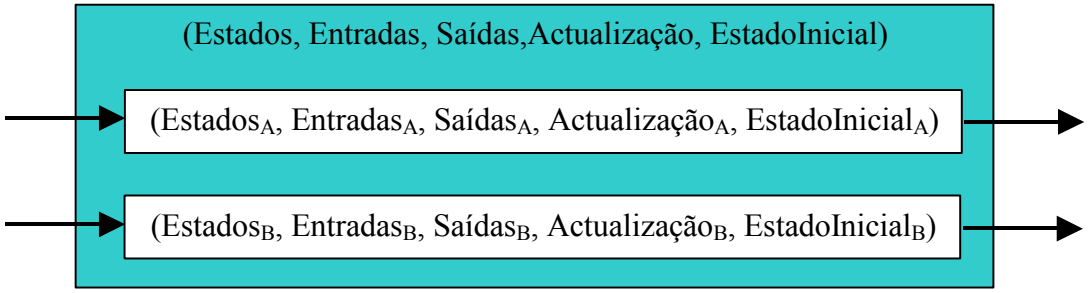
# 2.6 Composição de máquinas de estado

- Analisámos os sistemas como funções, logo a composição de sistemas é uma composição de funções
- Pretendemos definir uma nova máquina de estados que descreve a composição de várias máquinas de estado
- Vamos trabalhar com composição **síncrona**, ou seja, cada máquina de estados na composição reage simultaneamente ou instantaneamente
- A reacção da composição consiste num conjunto de reacções simultâneas de cada uma das máquinas componentes
- Um sistema que reage apenas em resposta a estímulos externos é denominado de **reactivo**
- Como a composição é síncrona estes sistemas são denominados de **reactivos síncronos**
- As reacções são instantâneas, o símbolo de saída de uma máquina acontece de uma forma simultânea com o símbolo de entrada

# 2.6 Composição de máquinas de estado



- Composição lado-a-lado




  - As duas máquinas de estado não interferem entre si
  - Pretende-se definir uma única máquina de estados representando a operação síncrona das duas máquinas de estado componentes
  - Poderemos desejar que apenas uma máquina reaja
  - Se trocarmos a ordem das máquinas obtemos uma máquina diferente mas equivalente

# 2.6 Composição de máquinas de estado

– Definição da composição de máquinas lado-a-lado

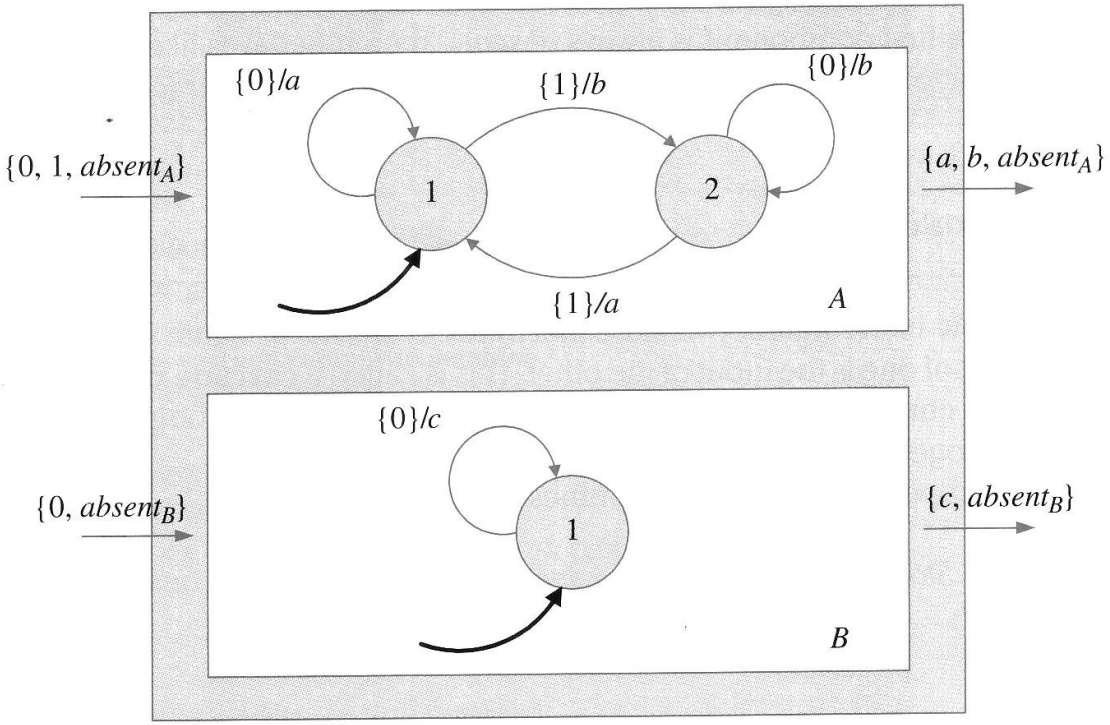
- $Estados = Estados_A \times Estados_B$
- $Entradas = Entradas_A \times Entradas_B$
- $Saídas = Saídas_A \times Saídas_B$
- $EstadoInicial = (EstadoInicial_A , EstadoInicial_B)$
- $((s_A(n+1),s_B(n+1)),(y_A(n),y_B(n))) = Actualização((s_A(n),s_B(n)),(x_A(n),x_B(n)))$

*onde*

- $(s_A(n+1),y_A(n)) = Actualização_A\ (s_A(n),x_A(n))$
- $(s_B(n+1),y_B(n)) = Actualização_B\ (s_B(n),x_B(n))$

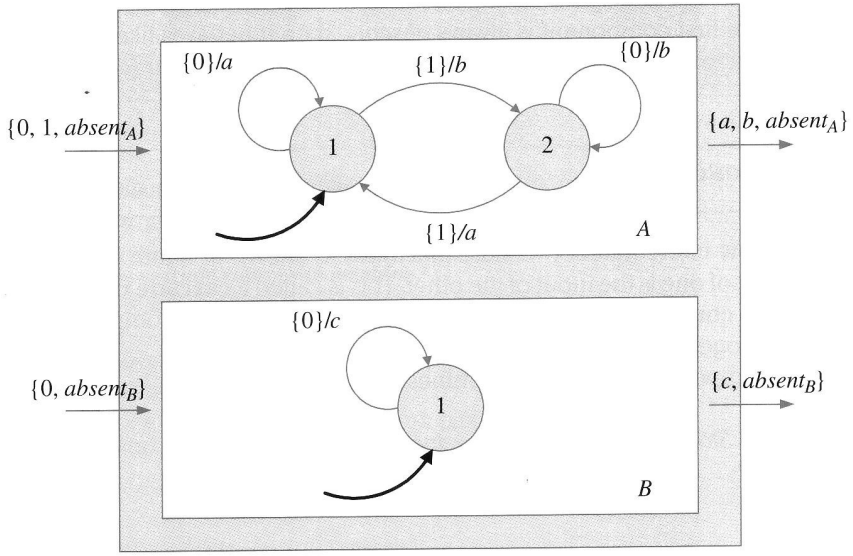
# 2.6 Composição de máquinas de estado

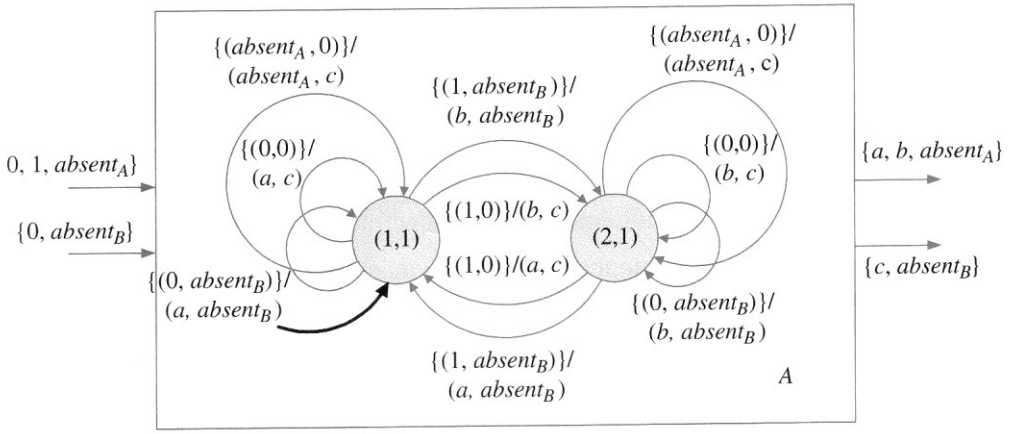
– Exemplo:



# 2.6 Composição de máquinas de estado

- $Estados = Estados_A \times Estados_B = \{(1,1),(2,1)\}$
- $Entradas = Entradas_A \times Entradas_B =$
  $\{(0,0),(1,0),(nulo_A,0),(0,nulo_B),(1,nulo_B),(nulo_A,nulo_B)\}$
- $Saídas = Saídas_A \times Saídas_B =$
  $\{(a,c),(b,c),(nulo_A,c),(a,nulo_B),(b,nulo_B),(nulo_A,nulo_B)\}$
- $EstadoInicial = (EstadoInicial_A , EstadoInicial_B) = (1,1)$

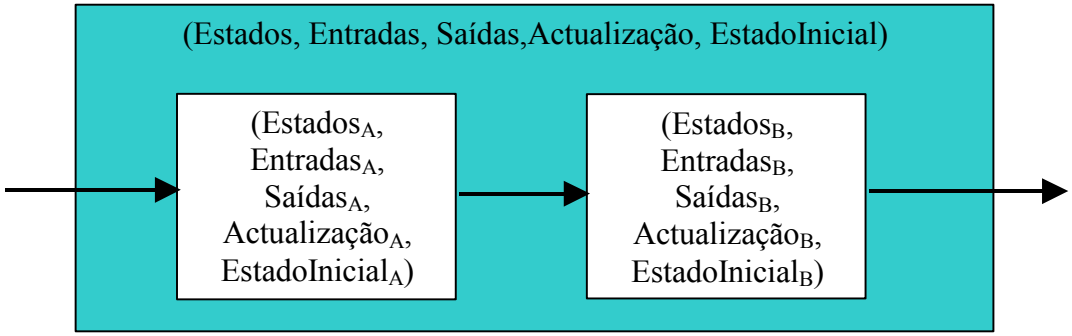


# 2.6 Composição de máquinas de estado



- Composição em cascata




  - Também designada por ligação em série

# 2.6 Composição de máquinas de estado

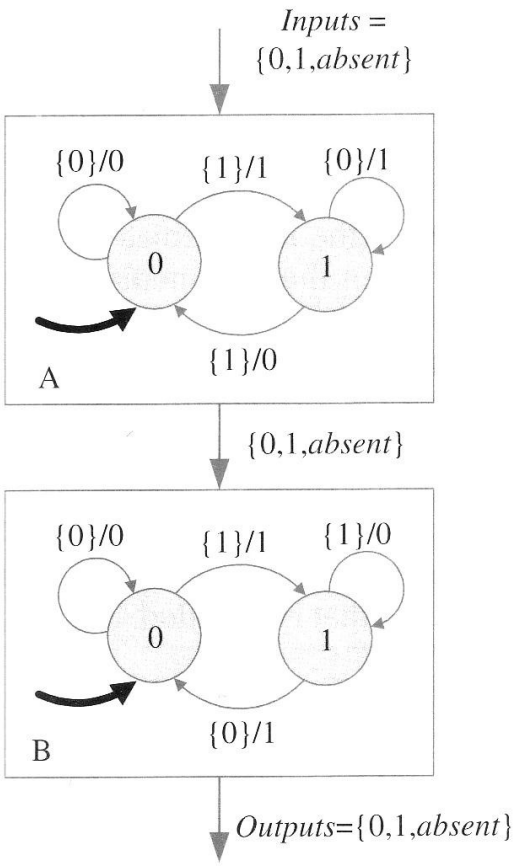
- Assumpções sobre as máquinas componentes
  - $Saídas_A \subset Entradas_B$
- Definição da composição de máquinas em cascata
  - $Estados = Estados_A \times Estados_B$
  - $Entradas = Entradas_A$
  - $Saídas = Saídas_B$
  - $EstadoInicial = (EstadoInicial_A , EstadoInicial_B)$
  - $((s_A(n+1),s_B(n+1)), y_B(n))=Actualização((s_A(n),s_B(n)),x(n))$

  *onde*

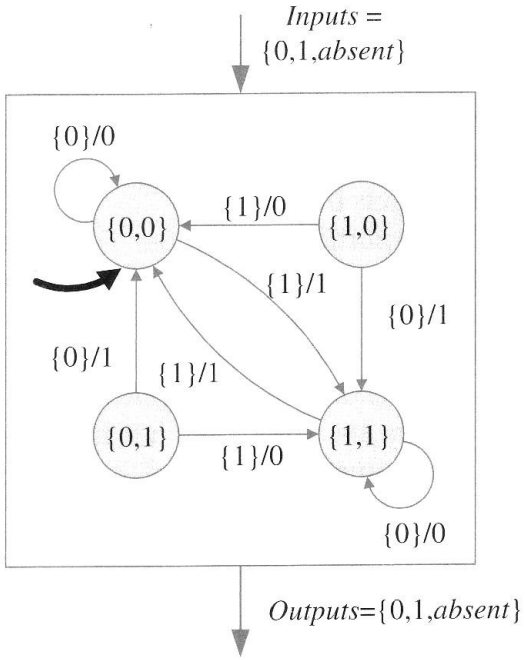
  - $(s_A(n+1),y_A(n))=Actualização_A\ (s_A(n),x(n))$
  - $(s_B(n+1),y_B(n))=Actualização_B\ (s_B(n),y_A(n))$

# 2.6 Composição de máquinas de estado

– Exemplo



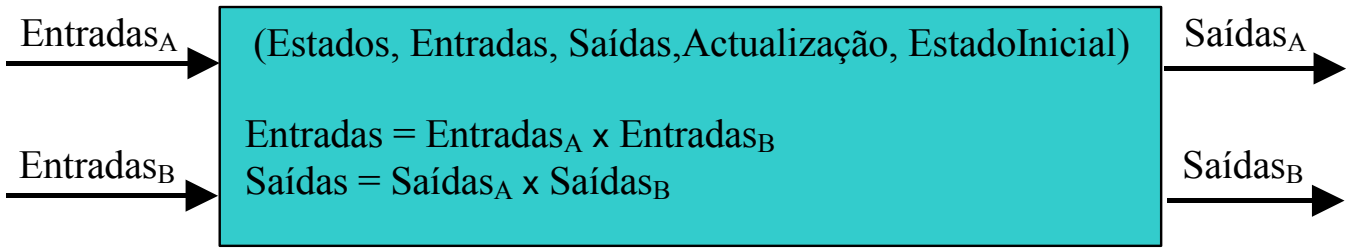

- *Estados = Estados$_A$ x Estados$_B$ = {(0,0),(0,1),(1,0),(1,1)}*
- *Entradas = Saídas = {0,1,nulo}*
- *EstadoInicial = (0,0)*



# 2.6 Composição de máquinas de estado

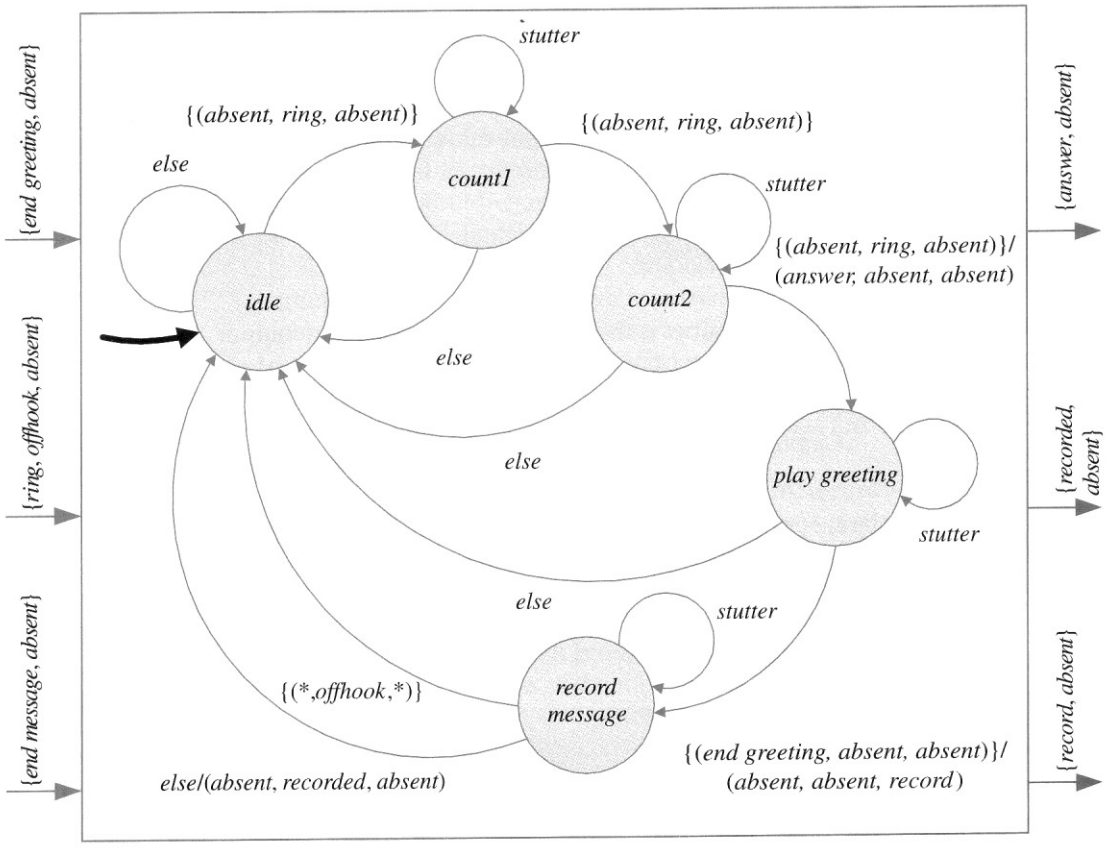


- Entradas e saídas sob a forma de produto
  - Pretende-se modelar a situação em que as entradas são seleccionadas de partes distintas e as saídas são enviadas para partes distintas



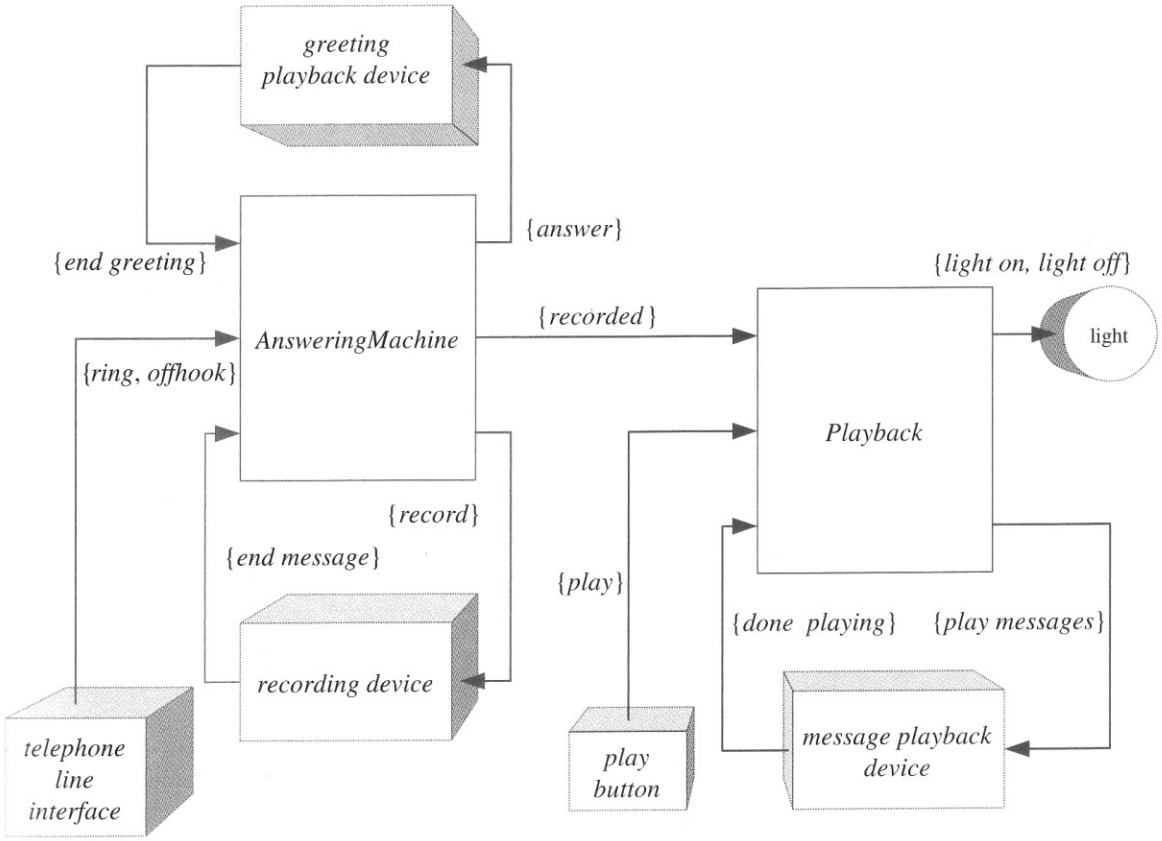
# 2.6 Composição de máquinas de estado

– Exemplo



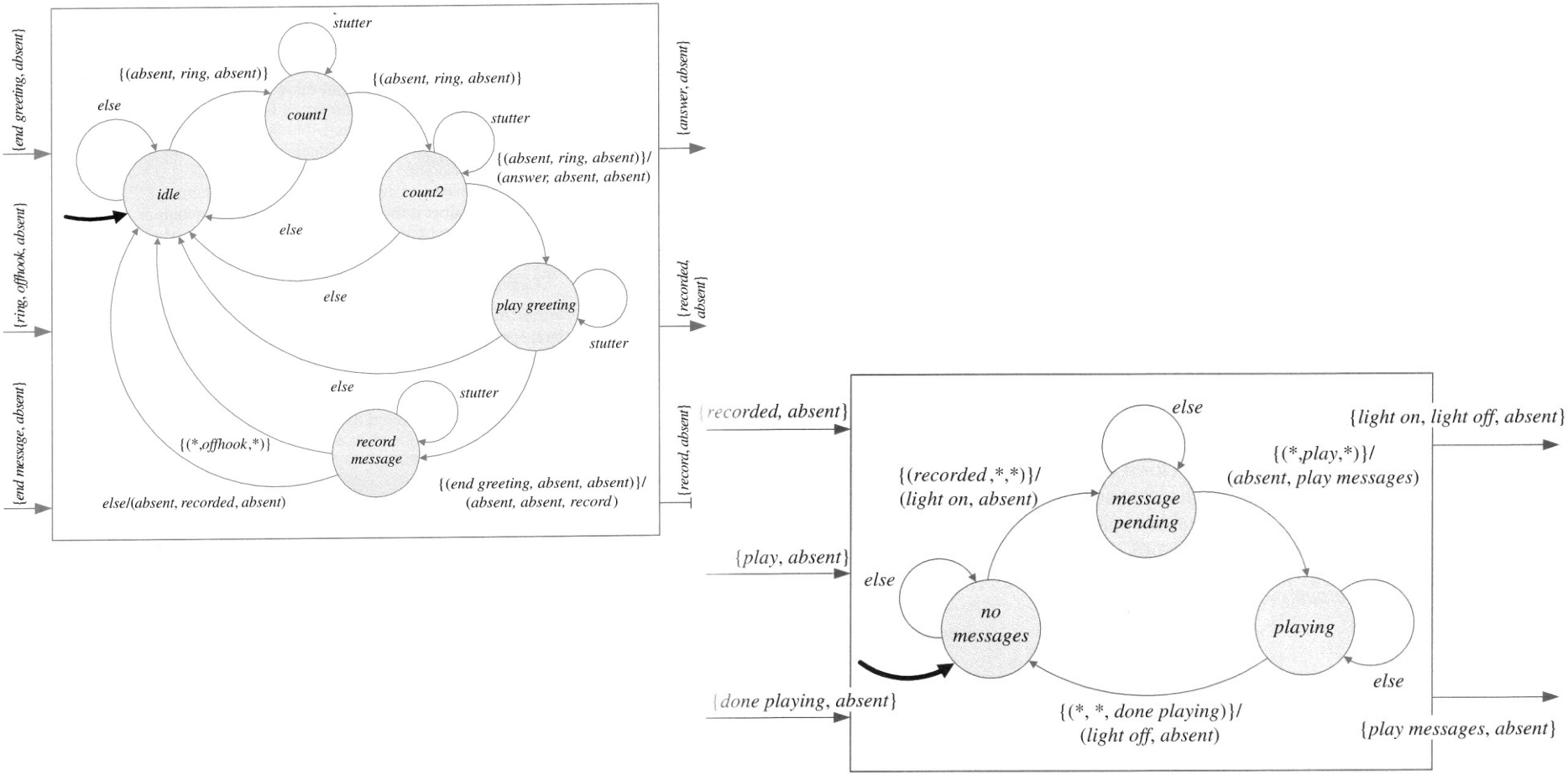
# 2.6 Composição de máquinas de estado

- Exemplo



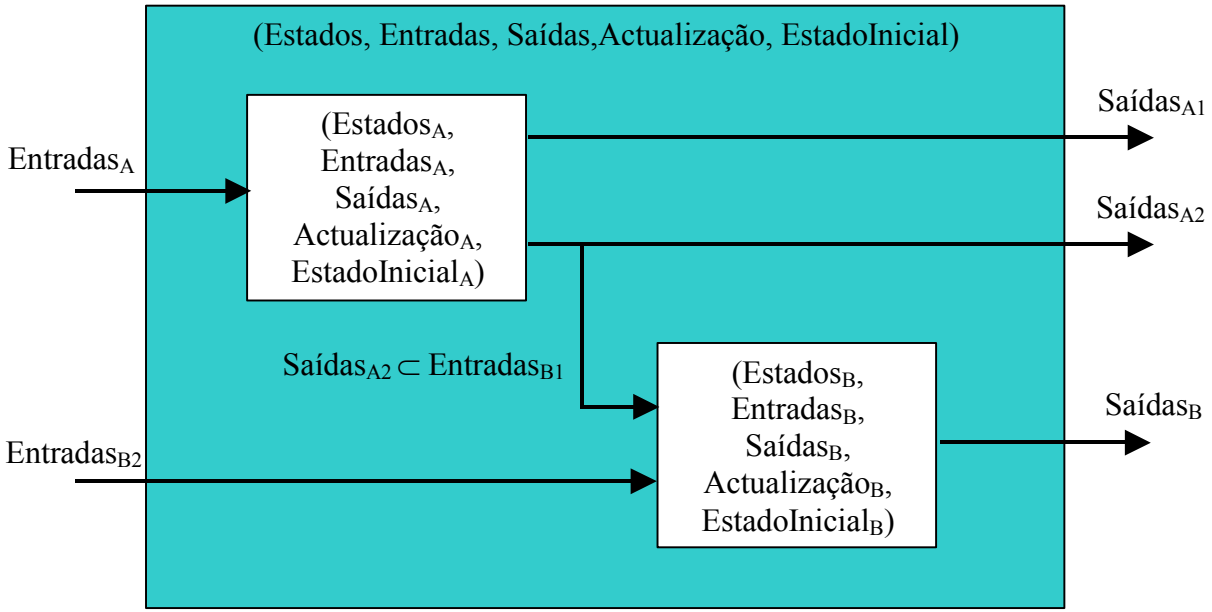
# 2.6 Composição de máquinas de estado





# 2.6 Composição de máquinas de estado

- Composição para a frente genérica




  - Exercício: defina a composição das máquinas para este exemplo

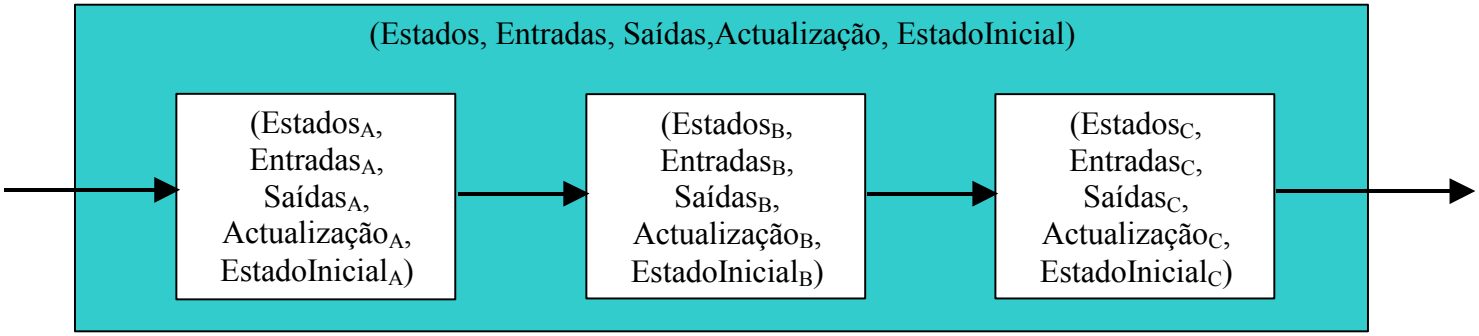
# 2.6 Composição de máquinas de estado

- Assumpções sobre as máquinas componentes
  - $Saídas_{A2} \subset Entradas_{B1}$
- Definição da composição para a frente genérica
  - $Estados = Estados_A \times Estados_B$
  - $Entradas = Entradas_A \times Entradas_{B2}$
  - $Saídas = Saídas_{A1} \times Saídas_{A2} \times Saídas_B$
  - $EstadoInicial = (EstadoInicial_A , EstadoInicial_B)$
  - $((s_A(n+1),s_B(n+1)), (y_{A1}(n),y_{A2}(n),y_B(n))) =$
    $Actualização((s_A(n),s_B(n)),(x_A(n), x_{B2}(n)))$
    *onde*
  - $(s_A(n+1),(y_{A1}(n), y_{A2}(n))) = Actualização_A(s_A(n),x_A(n))$
  - $(s_B(n+1),y_B(n)) = Actualização_B(s_B(n),(y_{A2}(n),x_{B2}(n)))$

# 2.6 Composição de máquinas de estado



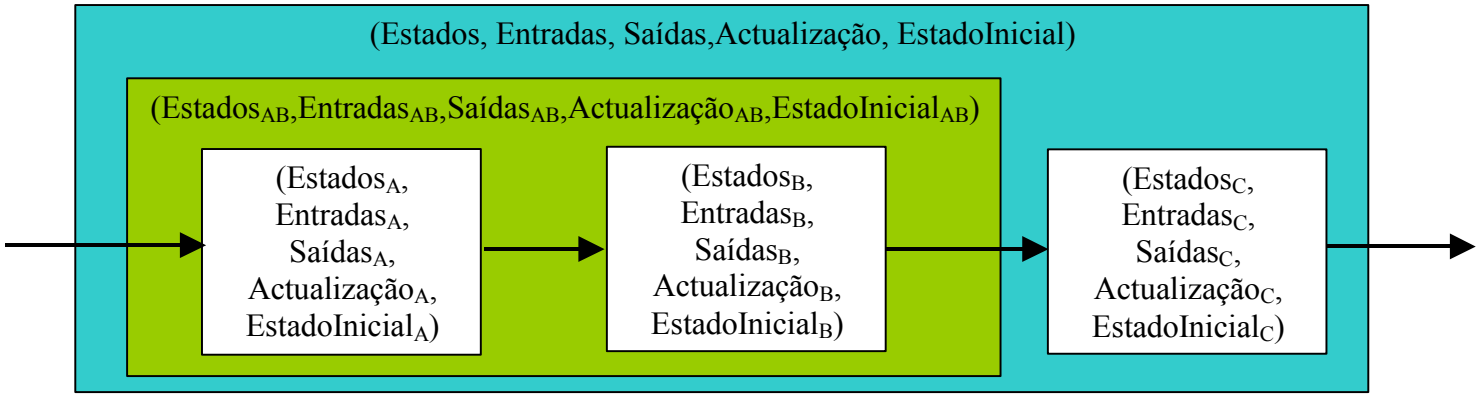
- Composição hierárquica

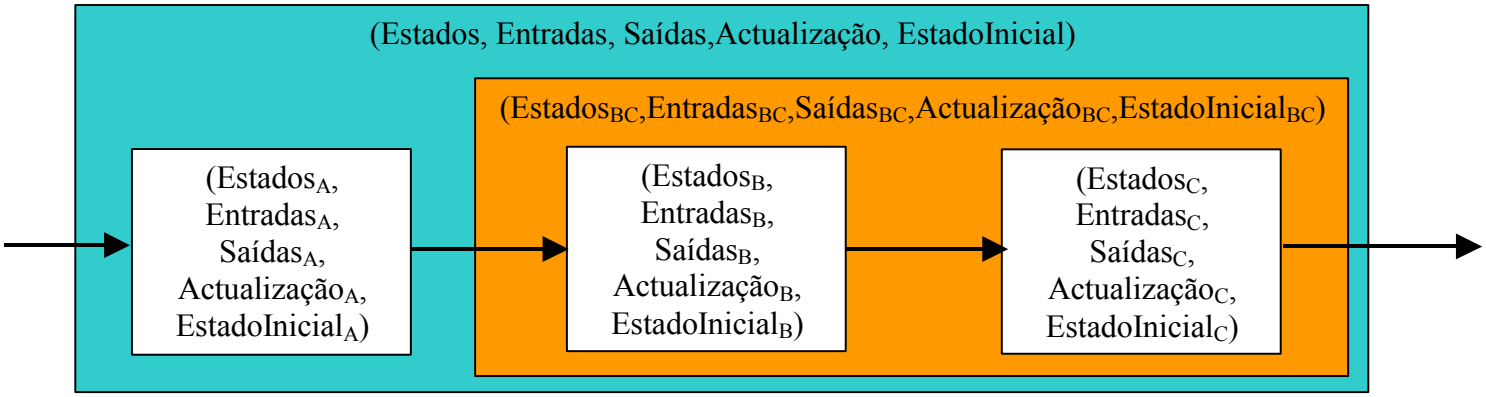



  - Diferentes estratégias para composição das máquinas
  - Essas estratégias geram máquinas de estado diferentes mas equivalentes

# 2.6 Composição de máquinas de estado



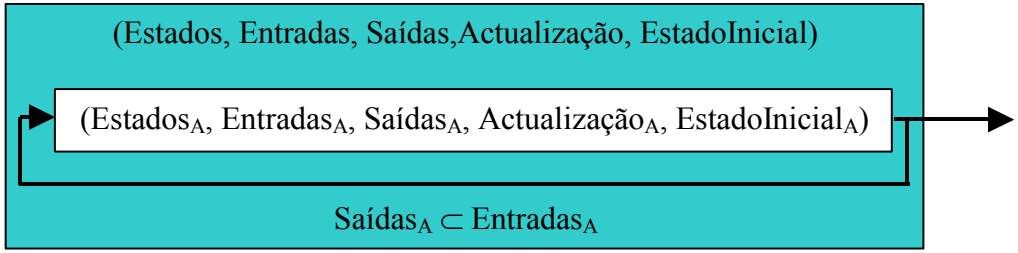


# 2.6 Composição de máquinas de estado



- Composição de máquinas com retroacção (feedback)
  - A saída de uma máquina de estados é ligada à entrada da mesma máquina de estados
  - A saída é uma consequência instantânea da entrada
  - Numa máquina com retroacção a saída depende da entrada que depende da saída ...



# 2.6 Composição de máquinas de estado



– Problema semelhante quando pretendemos determinar $x$ tal que

$$x = f(x)$$

para uma dada função $f$.

- A solução desta equação, se existir, é denominada de ponto fixo
- Pode não ter ponto fixo, pode ter um ponto fixo ou vários pontos fixos
- Exemplo:
  - $f(x)=1+x^2$ não tem solução em R
  - $f(x)=1-x$ solução única em $x=0,5$
  - $f(x)=x^2$ que tem duas soluções em $x=0$ e $x=1$

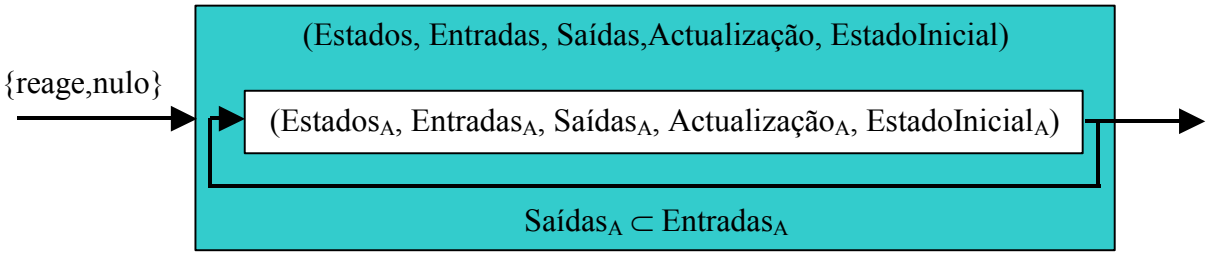
# 2.6 Composição de máquinas de estado



- Uma composição com retroacção sem ponto fixo nalgum estado atingível, é uma composição mal formada
  - Não se pode avaliar uma composição mal formada
- Uma composição com retroacção com mais de um ponto fixo nalgum estado atingível, também é uma composição mal formada
- Uma composição com retroacção com um ponto fixo em todos os estados atingíveis na composição, é uma composição bem formada

# 2.6 Composição de máquinas de estado

- Composição de retroacção sem entradas




  - Só é possível se $Saídas_A \subset Entradas_A$
  - Suponha-se que o estado corrente é $s(n) \in Estados_A$
  - Pretende-se encontrar o símbolo $y(n) \in Saídas_A$ que satisfaça

$$(s(n+1),y(n)) = Actualização_A(s(n),y(n))$$

  - Encontrando $y(n)$, a função $Actualização_A$ permite determinar $s(n+1)$
  - A composição anterior é bem formada se para cada estado atingível $s(n) \in Estados_A$ existe um único símbolo de saída $y(n)$ que satisfaz a equação anterior

# 2.6 Composição de máquinas de estado

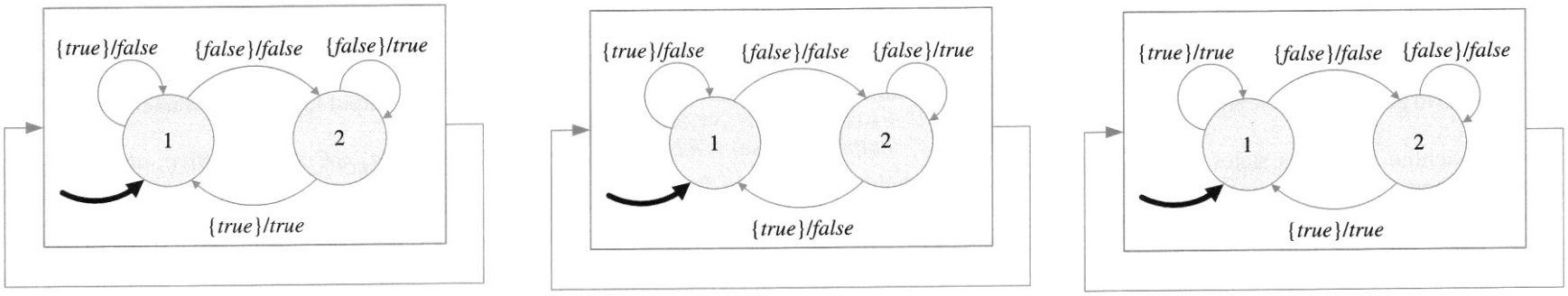


- Se a composição é bem formada a definição da máquina composta é
  - $Estados = Estados_A$
  - $Entradas = \{reage, nulo\}$
  - $Saídas = Saídas_A$
  - $EstadoInicial = EstadoInicial_A$
  - $Actualização(s(n),x(n)) =$

$$= \begin{cases} Actualização_A(s(n), y(n)) & se\ x(n) = reage \\ (s(n), x(n)) & se\ x(n) = nulo \end{cases}$$

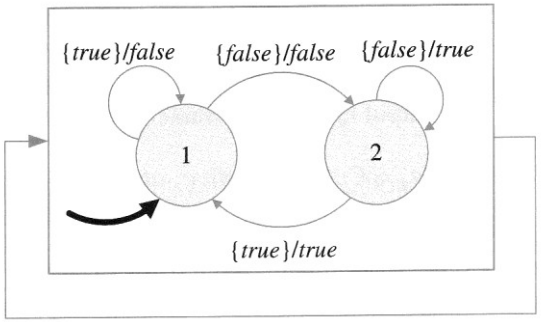
# 2.6 Composição de máquinas de estado

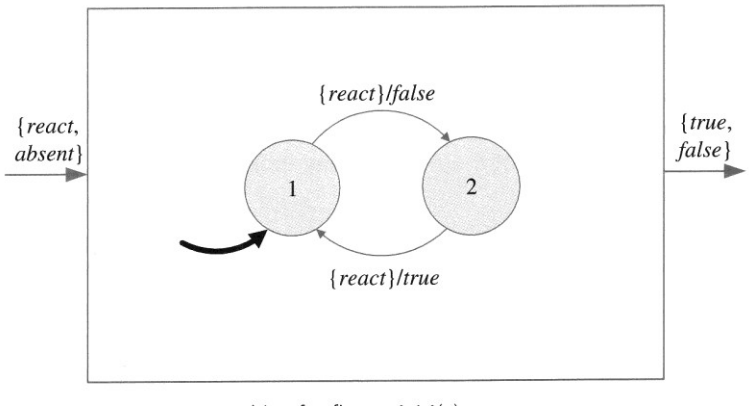
– Exemplo




- $Entradas_A = Saídas_A = \{true, false, nulo\}$
- O alfabeto de entrada da máquina composta é *{reage,nulo}*
- Pretende-se obter uma solução não nula para *y(n)*
- Como o símbolo de saída é também o símbolo de entrada, procuramos um símbolo de entrada diferente do nulo

# 2.6 Composição de máquinas de estado

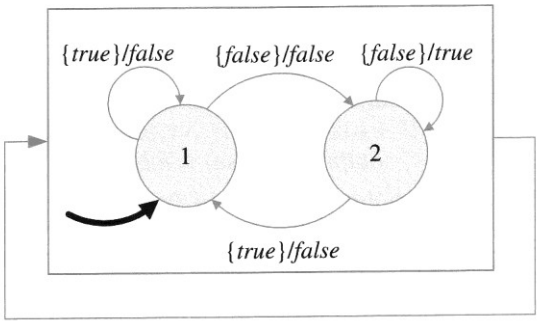



- *Saída$_A$(1,false)=false*
- *Saída$_A$(2,true)=true*
- A composição é bem formada porque em cada estado atingível existe um ponto fixo único
- Sequência de saída *(false,true,false,true,false,true,...)*
  para a sequência de entrada *(reage,reage,reage,...)*

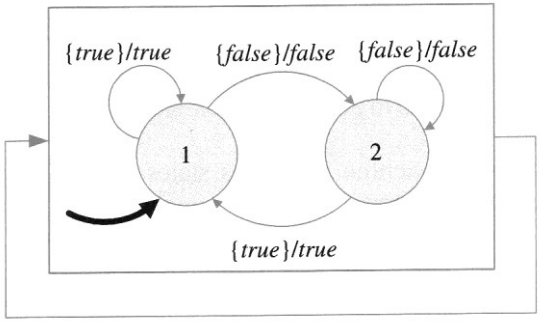
# 2.6 Composição de máquinas de estado






- $Saída_A(1,false)=false$
- Não há solução para $Saída_A(2,y(n))=y(n)$
- A composição é mal formada

# 2.6 Composição de máquinas de estado



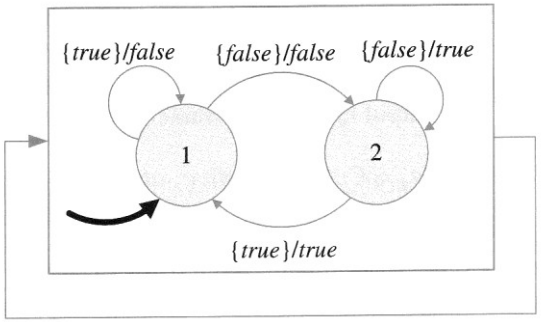

- Mais do que um ponto fixo
- Composição mal formada

– Conclusão

- A 2ª e a 3ª máquina não podem ser usadas numa composição com retroacção
- A 2ª não tem solução, a 3ª tem mais de uma solução
- Aceitamos composições com retroacção se existir exactamente uma solução não nula em cada estado atingível

# 2.6 Composição de máquinas de estado

- Composição de retroacção com saída determinada pelo estado
  - Exemplo anterior
    - Todos os arcos produzem o mesmo símbolo de saída, independentemente da entrada
    - O *y(n)* depende apenas do estado
    - *y(n)=false* se *s(n)=1* e *y(n)=true* se *s(n)=2*
    - Tem apenas um único ponto fixo



# 2.6 Composição de máquinas de estado

- Diz-se que uma máquina **A** tem a **saída determinada pelo estado** se em cada estado atingível $s(n) \in Estados_A$ existe um único símbolo de saída $y(n)=b$ (que depende de $s(n)$), e que é independente do símbolo nulo

  $x(n) \neq nulo \qquad Saída_A(s(n),x(n))=b$

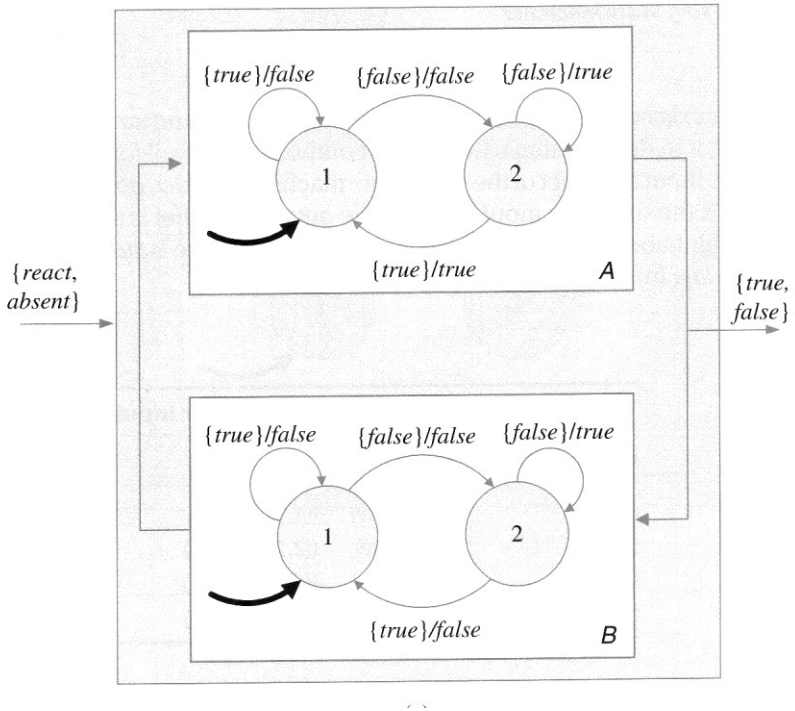
- Nesta situação a máquina composta é definida como
  - $Estados = Estados_A$
  - $Entradas = \{reage,nulo\}$
  - $Saídas = Saídas_A$
  - $EstadoInicial = EstadoInicial_A$
  - $Actualização(s(n),x(n)) =$

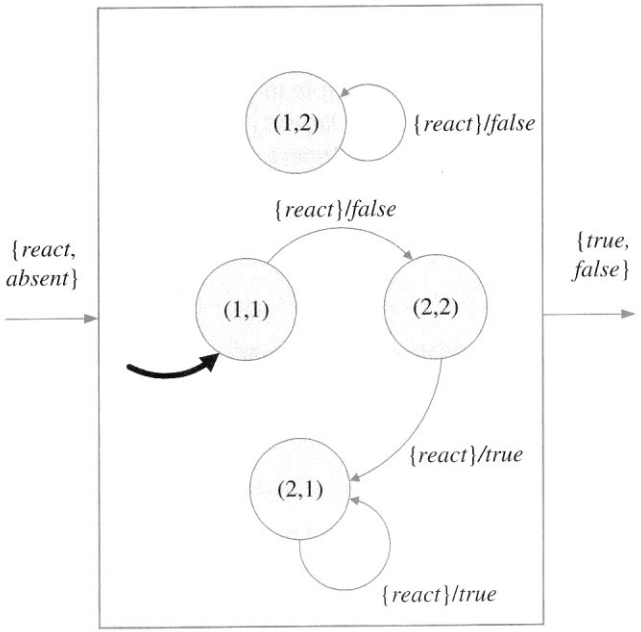
$$= \begin{cases} Actualização_A(s(n),b) & se\ x(n) = reage \\ (s(n),x(n)) & se\ x(n) = nulo \end{cases}$$

*onde b é o único símbolo de saída no estado s(n)*

# 2.6 Composição de máquinas de estado

– Exemplo



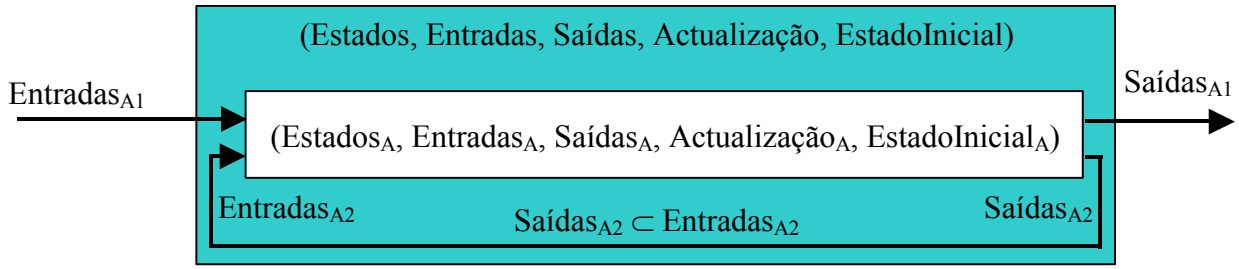




- **A** tem a saída determinada pelo estado, **B** não é bem formada, mas a composição é bem formada

# 2.6 Composição de máquinas de estado



- Composição de retroacção com entradas




  - Pretende-se construir uma máquina de estados que esconda a retroacção, ficando apenas com uma entrada e uma saída

# 2.6 Composição de máquinas de estado

– Máquina **A**:

- $Entradas_A = Entradas_{A1} \times Entradas_{A2}$
- $Saídas_A = Saídas_{A1} \times Saídas_{A2}$
- $Saídas_{A2} \subset Entradas_{A1}$
- $saída_A = (saída_{A1}, saída_{A2}) : Estados_A \times Entradas_A \rightarrow Saídas_A$
  onde $saída_{A1} : Estados_A \times Entradas_A \rightarrow Saídas_{A1}$
  **e** $saída_{A2} : Estados_A \times Entradas_A \rightarrow Saídas_{A2}$
- Pretende-se encontrar o símbolo de saída $(y_1(n), y_2(n)) \in Saídas_A$ tal que

$$(y_1(n), y_2(n)) = saída_A(s(n), (x_1(n), y_2(n)))$$

onde

$$y_1(n) = saída_{A1}(s(n), (x_1(n), y_2(n)))$$

**e**

$$y_2(n) = saída_{A2}(s(n), (x_1(n), y_2(n)))$$

# 2.6 Composição de máquinas de estado

– Composição:

- $Estados = Estados_A$
- $Entradas = Entradas_{A1}$
- $Saídas = Saídas_{A1}$
- $EstadoInicial = EstadoInitial_A$
- $(s(n+1),y(n)) = Actualização(s(n),x(n))$

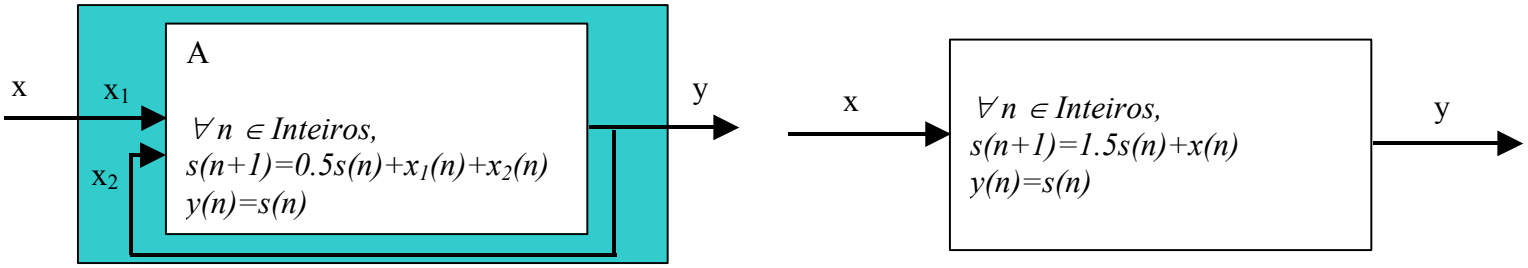
  $= ( EstadoSeguinte(s(n),x(n)) , saída(s(n),x(n)) )$

- $EstadoSeguinte(s(n),x(n)) = EstadoSeguinte_A(s(n),(x(n), y_2(n)))$
- $saída(s(n),x(n)) = saída_{A1}(s(n),(x(n), y_2(n)))$

  onde $y_2(n)$ é solução única

# 2.6 Composição de máquinas de estado

– Exemplo:




- $Entradas_A = Reais \times Reais$
- $Saídas_A = Reais$
- $Estados_A = Reais$
- $(s(n+1),y(n)) = Actualização_A(s(n),(x_1(n),x_2(n))) = (0.5s(n)+x_1(n)+x_2(n),s(n))$

- Como $x_2(n)=y(n)=s(n)$ então

  $Actualização(s(n),x(n)) = (0.5s(n)+x(n)+s(n),s(n)) = (1.5s(n)+x(n),s(n))$

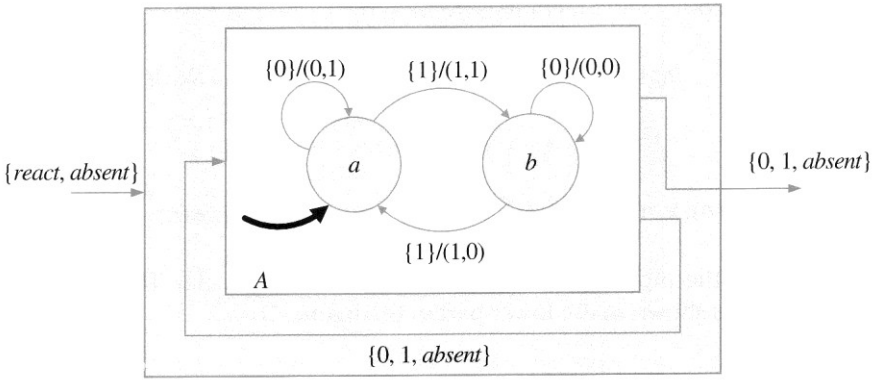
# 2.6 Composição de máquinas de estado



- Procedimento construtivo para uma composição com retroacção
  - Num número finito de passos identifica um ponto fixo ou desiste
  - Começa em cada reacção, para todos os sinais não especificados, com o valor "desconhecido"
  - Com o que é conhecido acerca dos símbolos de entrada experimenta determinar para cada máquina de estados o máximo possível acerca dos símbolos de saída
  - As máquinas de estado podem ser executadas em qualquer ordem
  - Com base no aprendido sobre os símbolos de saída actualiza o conhecimento sobre os símbolos de entrada e repete o processo
  - Repete o processo até que todos os valores dos sinais estejam especificados ou então não ser possível aprender nada mais

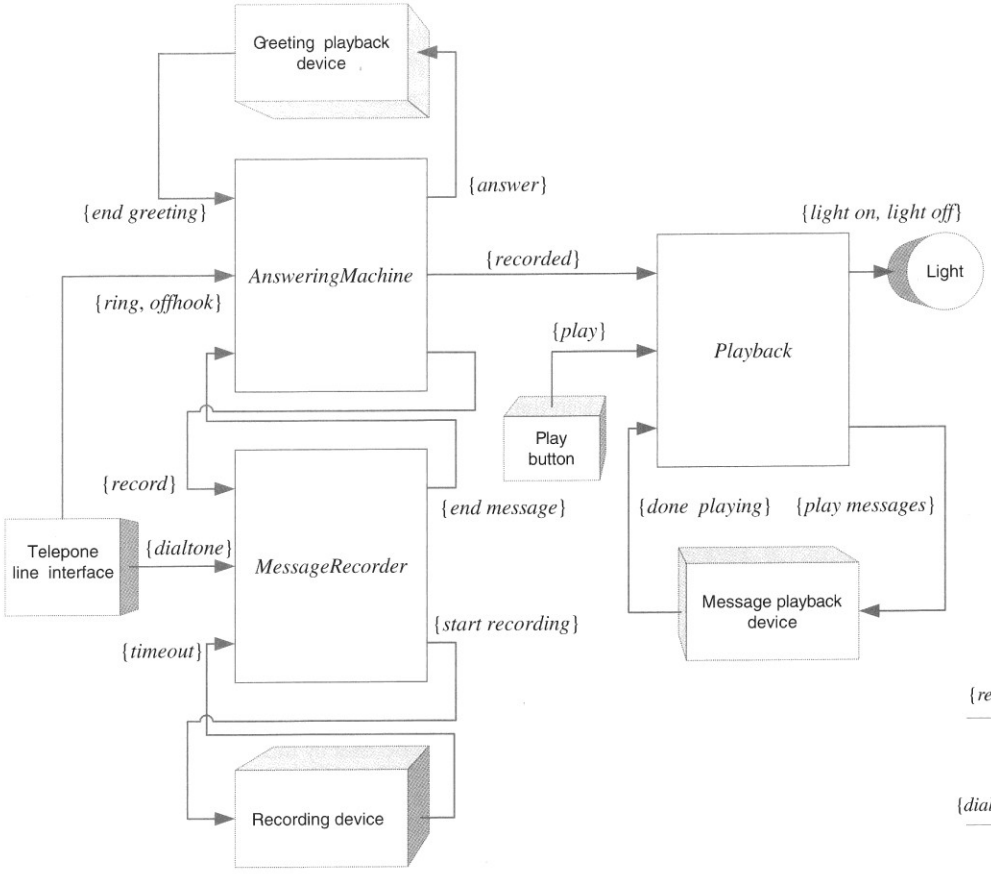
# 2.6 Composição de máquinas de estado

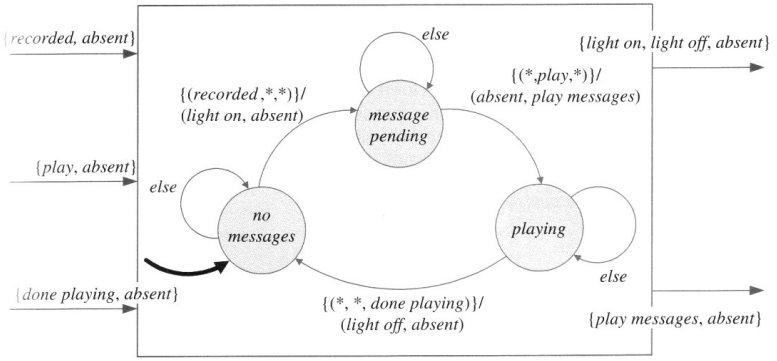
– Exemplo

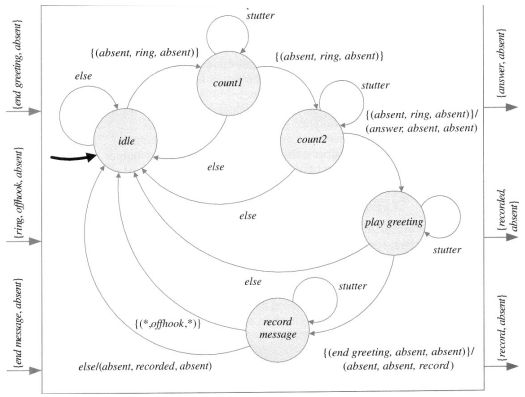


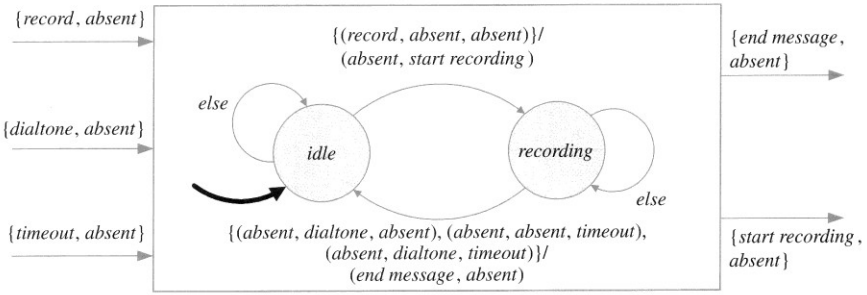

- Comportamento={( (reage,reage,reage,...) , (1,0,0,...) )}

# 2.6 Composição de máquinas de estado



# 2.6 Composição de máquinas de estado



- Busca exaustiva
  - Se na composição com retroacção existem uma ou mais máquinas cuja saída é determinada pelo estado, é fácil determinar um único ponto fixo
  - Se a máquina não tiver a saída determinada pelo estado aplicamos o procedimento anterior
    - Se falhar não podemos concluir que a composição é mal formada
  - Podemos determinar se a máquina tem um ponto fixo único através de uma busca exaustiva
    - Para cada estado atingível e para cada possível entrada, tentamos todas as possíveis transições
    - Rejeitamos aquelas que levam a contradições
    - Se após rejeitarmos todas as contradições restar apenas uma possibilidade em cada estado atingível a composição é bem formada
    - É necessário que o número de estados e o número de transições seja finito

# 2.6 Composição de máquinas de estado



- Máquinas não-determinísticas
  - As máquinas não-determinísticas podem ser compostas da mesma forma que as determinísticas
  - Como as máquinas determinísticas são um caso especial, os dois tipos de máquinas podem ser misturadas numa composição
  - Composições sem retroacção operam quase como as anteriores
  - Composições com retroacção necessitam de modificações:
    - Em cada reacção a máquina pode ter vários possíveis símbolos de saída e vários possíveis estados seguintes
    - Para cada máquina, para cada reacção definem-se os conjuntos
      - $PossíveisEntradas \subset Entradas$, $PossíveisEstadosSeguintes \subset Estados$, $PossíveisSaídasSeguintes \subset Saídas$
    - Se as entradas para uma dada máquina na composição é conhecida completamente então *PossíveisEntradas* tem apenas um elemento, se são desconhecidas então *PossíveisEntradas* é vazio

# Conclusões

- Muitos sistemas são desenhados como máquinas de estados
- A estrutura resulta da composição de várias máquinas de estado
- Considerámos composição síncrona
- Composição com retroacção torna-se particularmente difícil porque o símbolo de entrada pode depender do símbolo de saída na mesma reacção
- Denominámos uma composição com retroacção bem formada se qualquer sinal tem um único símbolo nulo em cada reacção