Arquitectura de Sistemas de Software y Tecnologías Asociadas
Labelled Transition Systems

Alejandro Sánchez
asanchez@unsl.edu.ar

Departamento de Informática
Universidad Nacional de San Luis

Maestría en Informática
Universidad Nacional de San Juan
8-9 Nov 2013

Estructura de la presentación

1. Características principales
   - Motivación
   - ¿Qué es un LTS?

2. Relaciones
   - Motivación
   - Trace equivalence
   - Bisimulation equivalence

Estructura de la presentación

Desarrollo de sistemas reactivos correctos

Sistema reactivo

Un sistema que computa
al reaccionar a estímulos de su ambiente

Modelar, especificar, analizar y verificar sistemas reactivos

Ejemplos de sistemas reactivos I

```
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

Ejemplos de sistemas reactivos I

```
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

¿Es este programa un sistema reactivo?

Ejemplos de sistemas reactivos II

```
public class HelloWorld extends HttpServlet{
  public void doGet(HttpServletRequest request,
   HttpServletResponse response)
  throws ServletException,IOException{
  response.setContentType("text/html");
  PrintWriter pw = response.getWriter();
  pw.println("<html>");
  pw.println("<head><title>Hello World</title></title>");
  pw.println("<body>");
  pw.println("<h1>Hello World</h1>");
  pw.println("</body></html>");
  }
}
```

Ejemplos de sistemas reactivos II

```
public class HelloWorld extends HttpServlet{
  public void doGet(HttpServletRequest request,
   HttpServletResponse response)
  throws ServletException,IOException{
  response.setContentType("text/html");
  PrintWriter pw = response.getWriter();
  pw.println("<html>");
  pw.println("<head><title>Hello World</title></title>");
  pw.println("<body>");
  pw.println("<h1>Hello World</h1>");
  pw.println("</body></html>");
  }
}
```

¿Es este servlet un sistema reactivo?

Ejemplos de sistemas reactivos III

- Sistemas operativos
- Protocolos de comunicación,
  como los usados por un servicio web
- Programas de control,
  como el de un elevador, o en una central nuclear
- Software ejecutando en sistemas embebidos,
  como el de teléfonos móviles

Conceptos clave en sistemas reactivos

Se debe modelar:

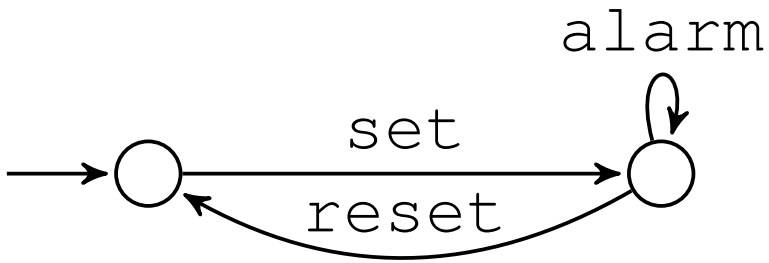
- Observación $\equiv$ interacción
- Comportamiento $\equiv$ registro estructurado de interacciones
- El comportamiento es determinado por la interacción y movilidad de procesos que no terminan, y evolucionan concurrentemente

El Modelo

Modelamos los sitemas reactivos con

labelled transition systems (LTSs)

(sistemas de transición etiquetados)



Action

Las *actions* son el ingrediente fundamental de los LTSs

Características

- Representa un evento
- Observable
- Atómica
- Puede tener parámetros

Ejemplos
Un sistema abstracto: *a, b, c*
Un reloj alarma: *set, reset, alarm*
Un reloj alarma con repeticiones de alarma: *set(n), reset, alarm*

Definición

Labelled Transition System (LTS)

Un LTS es una tupla $A = \langle S, Act, \rightarrow, s, \downarrow \rangle$ donde:

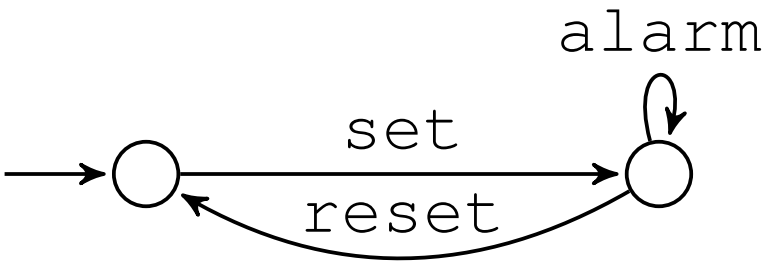
- S es un conjunto de estados
- Act es un conjunto de acciones
- $\rightarrow \subseteq S \times Act \times S$ es la relación de transición
- s es el estado inicial
- $\downarrow \subseteq S$ es el conjunto de estados terminales

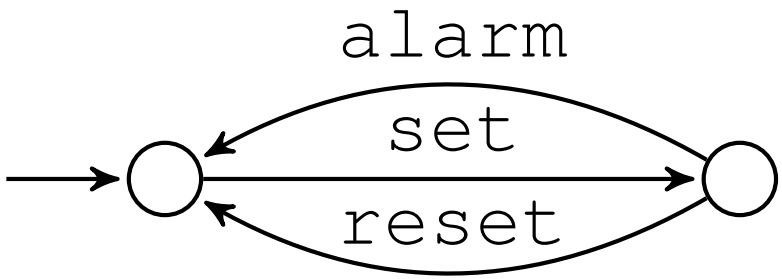
Notación

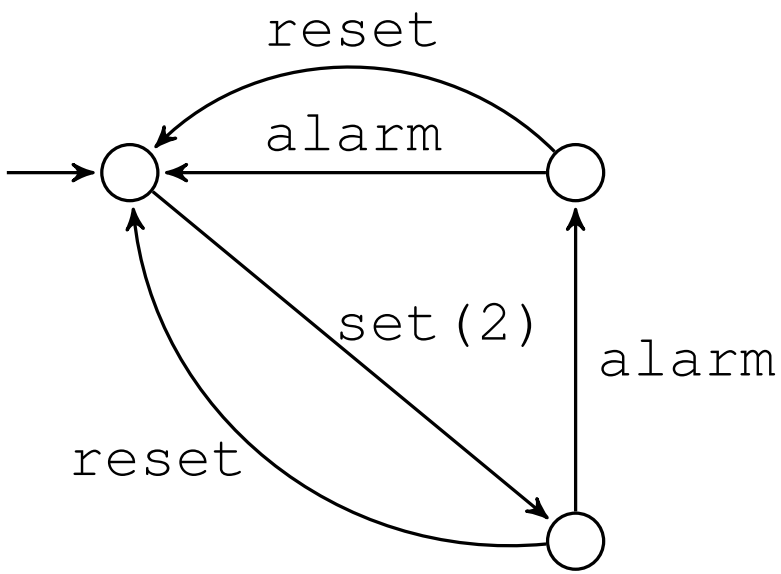
$\downarrow s_1 \equiv s_1 \in \downarrow$

$s_i \xrightarrow{a} s_j \equiv (s_i, a, s_j) \in \rightarrow$

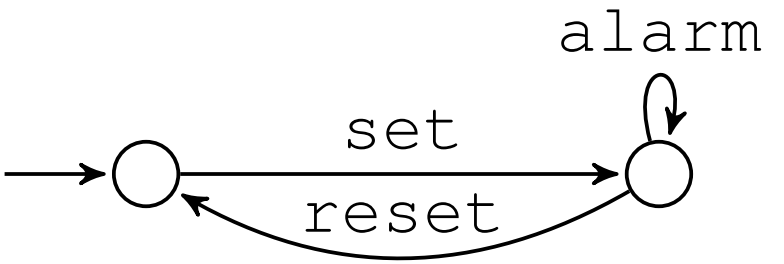
Ejemplos - Relojes alarma

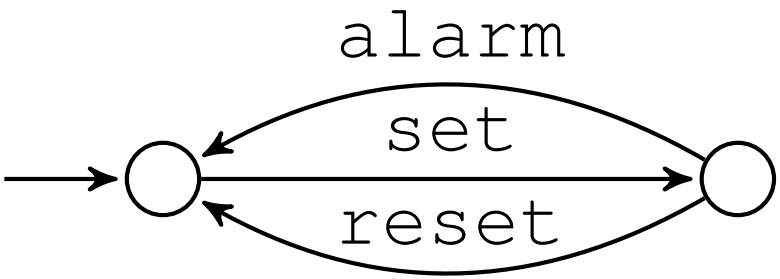


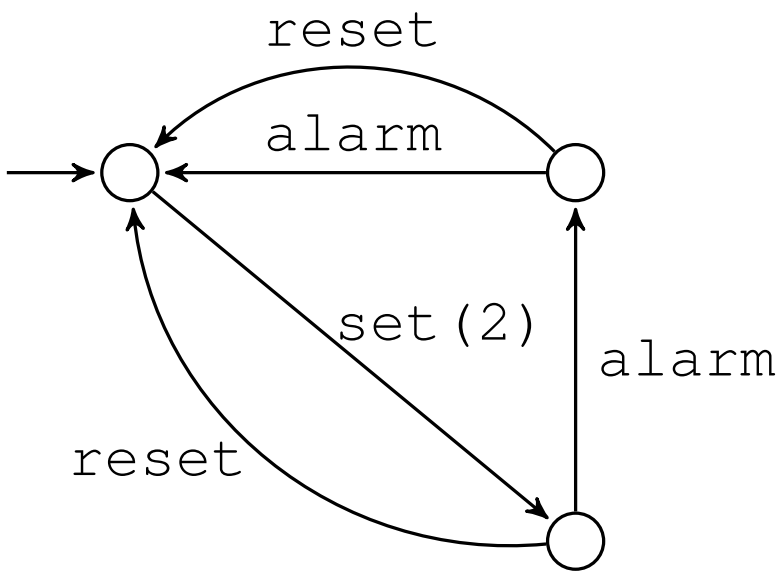




Ejemplos - Relojes alarma







¿En qué se diferencian?

Reachability

(Accesibilidad)

Reachability

La relación de *reachability*, $\rightarrow^* \subseteq S \times Act \times S$, se define inductivamente:

- $s \xrightarrow{\epsilon}{}^* s$, para cada $s \in S$, donde $\epsilon \in Act^*$ denota una palabra vacía
- si $s \xrightarrow{a} s''$ y $s'' \xrightarrow{\sigma}{}^* s'$, entonces $s \xrightarrow{a\sigma}{}^* s'$, para $a \in Act$ y $\sigma \in Act^*$

$t \in S$ es accesible (*reachable*) desde $s \in S$ si hay una palabra $\sigma \in Act^*$ tal que $s \xrightarrow{\sigma}{}^* t$

Sistemas

Sistemas

Dado un LTS $A = \langle S, Act, \rightarrow, s, \downarrow \rangle$,
cada $s_i \in S$ determina un sistema sobre todos los estados accesibles desde s_i
y las restricciones dadas por $\rightarrow$ y $\downarrow$

Clasificación de LTSs

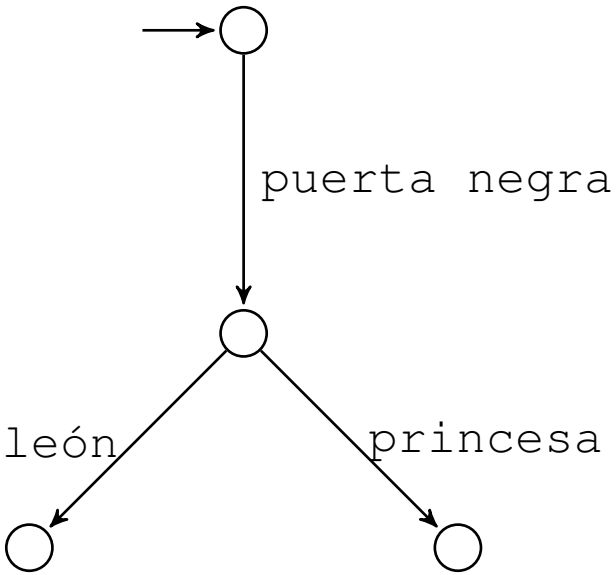
- determinístico
- no determinístico
- finito
- imagen finita (*image finite*)
- ...

Clasificación de LTSs: Ejemplo

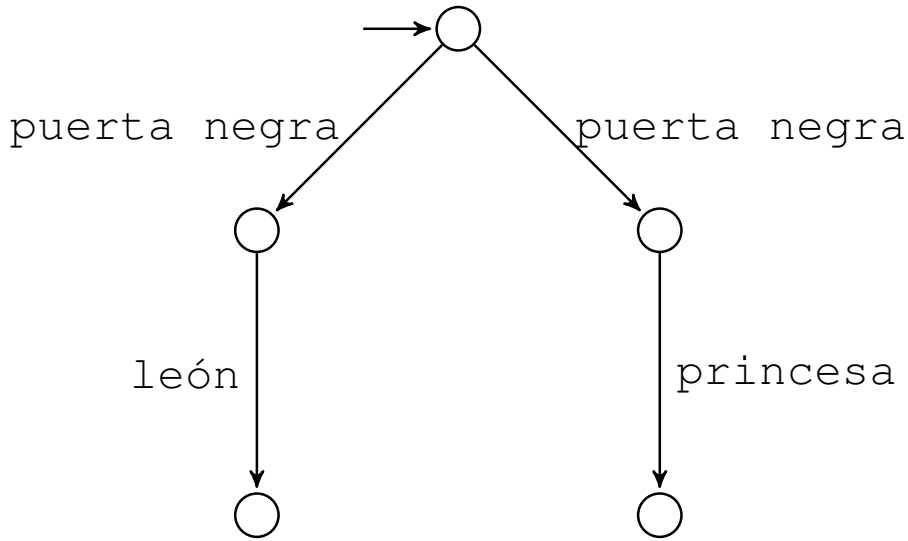
Determinístico vs. no determinístico

(o por que a veces el príncipe encantador fracasa)

Laberinto 1



Laberinto 2



Estructura de la presentación

1. Características principales
   - Motivación
   - ¿Qué es un LTS?

2. Relaciones
   - Motivación
   - Trace equivalence
   - Bisimulation equivalence

Comparar utilizando lenguajes regulares

comportamiento *automaton* finito $\equiv$ lenguaje aceptado

Un autómata finito definido a partir de la sintaxis

$$E ::= \epsilon \mid a \mid E + E \mid EE \mid E^*$$

genera un lenguaje regular

$$\begin{aligned} E_1 + E_2 &\triangleq L_1 \cup L_2 \\ E_1 E_2 &\triangleq \{ab \mid a \in L_1 \wedge b \in L_2\} \\ E^* &\triangleq \{\epsilon\} \cup L \cup LL \cup LLL \cup \ldots \end{aligned}$$

Requerimientos para modelar sistemas reactivos

- Varios puntos de interacción, en contraste con una función
- Distinguir un final normal de uno anómalo (*deadlock*)
- El lenguaje aceptado no distingue al no determinismo (al príncipe encantador le gustaría poder distinguir a la hora de elegir un laberinto)
- Los estados recorridos en una ejecución son relevantes

Equivalencia de comportamientos

¿Cuándo dos sistemas tienen el mismo comportamiento?

Cuando la diferencia de comportamiento
no puede ser observada

¿Cómo se observa el comportamiento?

Hay muchas formas, veremos dos:
trace equivalence y
bisimulation equivalence

Trace equivalence: Definición

Trace equivalence

Dado un LTS $A = \langle S, Act, \rightarrow, s, \downarrow \rangle$, el conjunto de *traces* $Tr(t)$ de un estado $t \in S$ es el conjunto mínimo que:

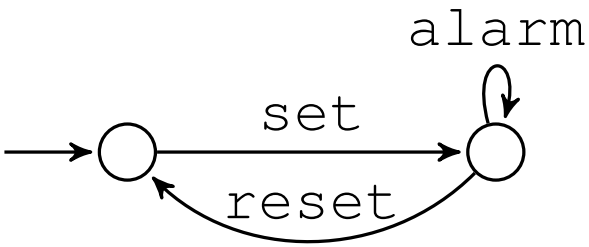
- $\epsilon \in Tr(t)$
- $\surd \in Tr(t)$
- si $\exists\, t' \in S$ tal que $t \xrightarrow{a} t'$ y $\sigma \in Tr(t'), \Rightarrow a\sigma \in Tr(t)$

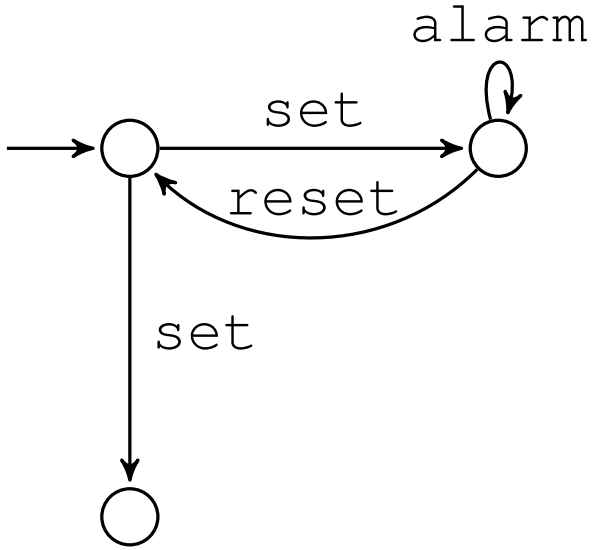
con ϵ una palabra vacía y $\surd$ indicando terminación.
Dos estados $s, t \in S$ son *trace equivalent* si $Tr(s) = Tr(t)$.
Dos LTSs son *trace equivalent* si sus estados iniciales lo son.

Dos sistemas son equivalentes si las mismas secuencias de acciones pueden ejecutarse desde sus estados iniciales

Utilidad

Apropiado cuando uno no puede interactuar con el sistema ni distinguir un sistema lento de uno que se encuentra detenido





No distingue entre estos dos sistemas.
Tampoco entre los laberintos del príncipe encantador.

Simulation

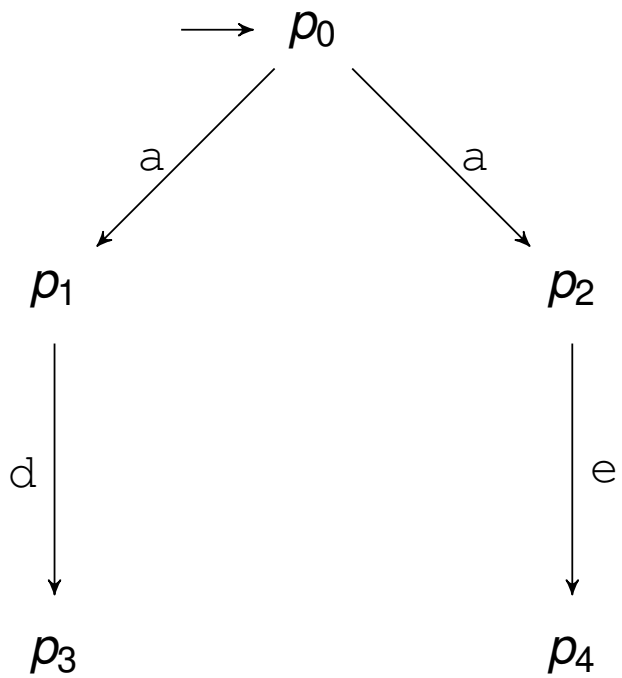
Simulation

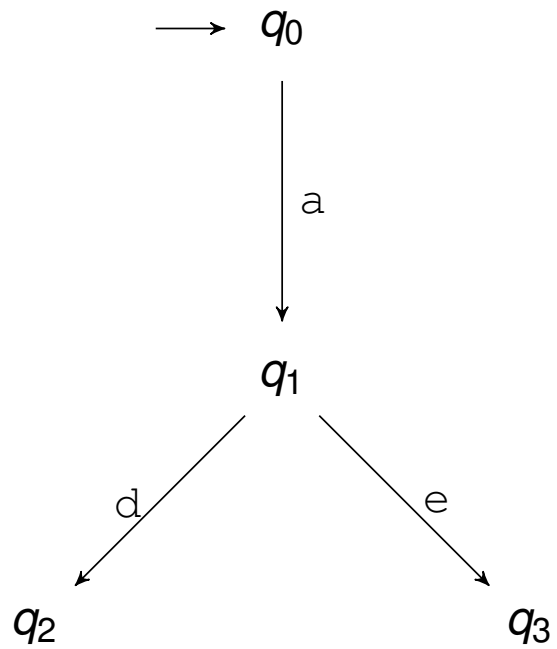
Dados $\langle S_1, Act, \rightarrow_1, s_1, \downarrow_1 \rangle$ y $\langle S_2, Act, \rightarrow_2, s_2, \downarrow_2 \rangle$, la relación $R \subseteq S_1 \times S_2$ es una *simulation* si y solo si $\forall \langle p, q \rangle \in R$ y $a \in Act$

- $p \downarrow_1 \Rightarrow q \downarrow_2$
- $p \xrightarrow{a}_1 p' \Rightarrow \langle \exists q' : q' \in S_2 : q \xrightarrow{a}_2 q' \wedge \langle p', q' \rangle \in R \rangle$

Un estado p simula un estado q si cada transición desde p es correspondida por una transición desde q y esta capacidad se mantiene en todo estado accesible desde p.

Simulation: Ejemplo





$$R = \langle (p_0, q_0), (p_1, q_1), (p_2, q_1), (p_3, q_2), (p_4, q_3) \rangle$$

Similarity

Similarity

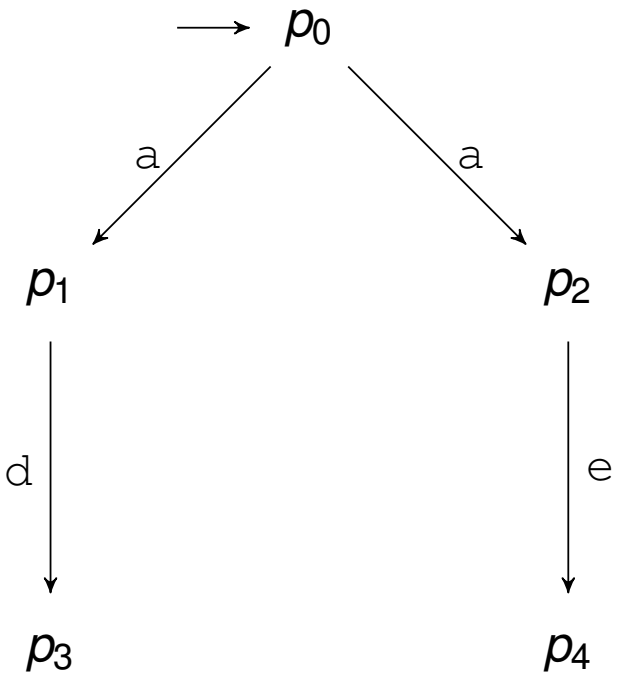
$p \lesssim q$ si y solo si $\langle \exists R : R$ es una *simulation* $\wedge \langle p, q \rangle \in R \rangle$

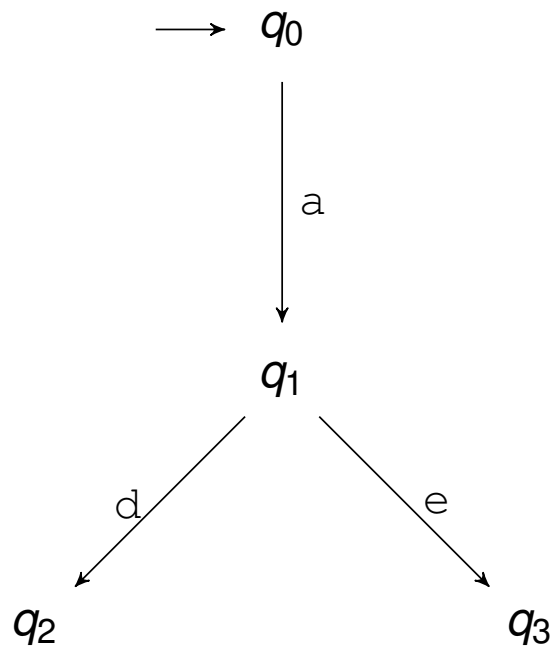
Es una relación de preorden (reflexiva y transitiva)

Similarity como la *simulation* más grande.

$$\lesssim \triangleq \bigcup \{R \mid R \text{ es simulation}\}$$

Similarity: Ejemplo





$$p_0 \lesssim q_0$$

Bisimulation

Bisimulation

Dados $\langle S_1, Act, \rightarrow_1, s_1, \downarrow_1 \rangle$ y $\langle S_2, Act, \rightarrow_2, s_2, \downarrow_2 \rangle$, la relación $R \subseteq S_1 \times S_2$ es una *bisimulation* si y solo si R y R° son *simulations*, *i.e.*,

- $p \downarrow_1 \Leftrightarrow q \downarrow_2$
- $p \xrightarrow{a}_1 p' \Rightarrow \langle \exists q' : q' \in S_2 : q \xrightarrow{a}_2 q' \wedge \langle p', q' \rangle \in R \rangle$
- $q \xrightarrow{a}_2 q' \Rightarrow \langle \exists p' : p' \in S_1 : p \xrightarrow{a}_1 p' \wedge \langle p', q' \rangle \in R \rangle$

Si dos procesos son *bisimulation equivalent*
no se pueden distinguir por ninguna forma
de observación de comportamiento realística.
Pueden considerarse iguales

Bisimilarity

Bisimilarity

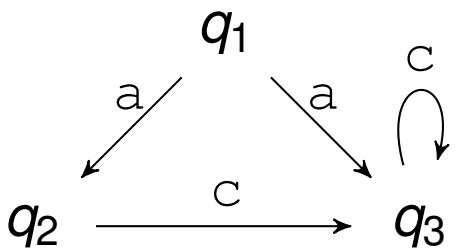
$p \sim q$ si y solo si $\langle \exists R : R$ es una *bisimulation* $\wedge \langle p, q \rangle \in R \rangle$

Es una relación de equivalencia (reflexiva, simétrica, transitiva).

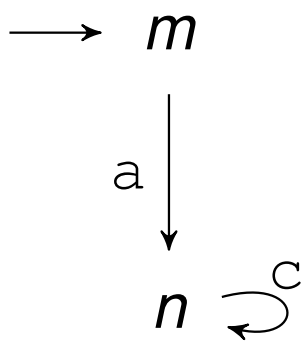
Bisimilarity como la bisimulación más grande.

$$\sim \triangleq \bigcup \{R \mid R \text{ es bisimulation}\}$$

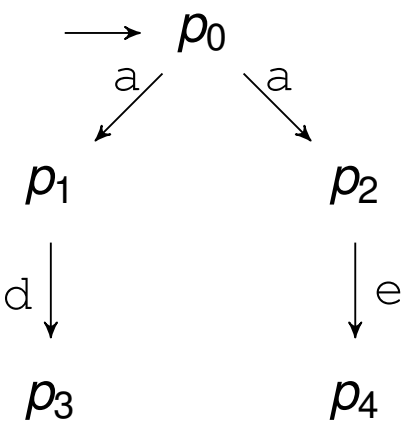
Bisimulation: Ejemplo



$q_1 \sim m$

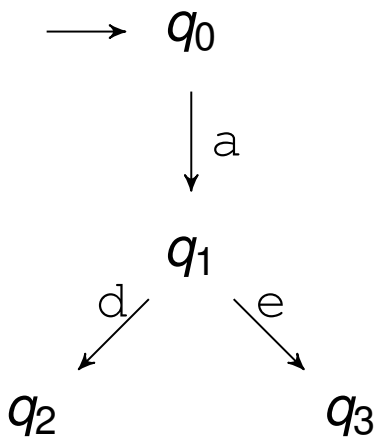


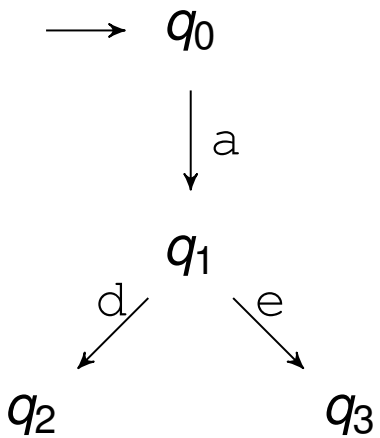
Bisimulation: Ejemplo laberinto



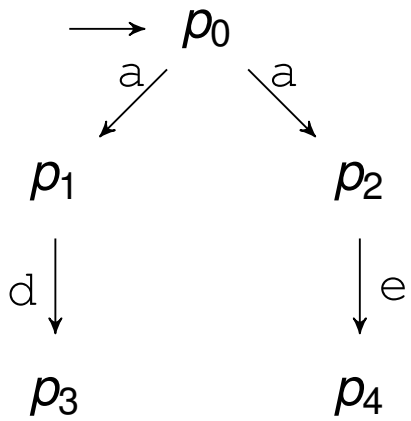
$q_0 \not\sim p_0$

$p_0 \lesssim q_0$





$q_0 \not\lesssim p_0$



Ejercicio

Suponga un LTS dado por la relación

$$\{ (1,a,2),(1,a,3),(2,a,3),(2,b,1),$$
$$(3,a,3),(3,b,1),(4,a,5),(5,a,5),$$
$$(5,b,6),(6,a,5),(7,a,8),(8,a,8),(8,b,7)\}$$

Probar o refutar que $1 \sim 4 \sim 6 \sim 7$.