Métodos para la Construcción de Software Fiable

Programa de Doctorado: Ingeniería del Software e Inteligencia Artificial

María del Mar Gallardo

Pedro Merino

Introducción

Para aplicar Model Checking necesitamos

Modelar el sistema M que va a ser analizado mediante algún *Lenguaje de Modelado*

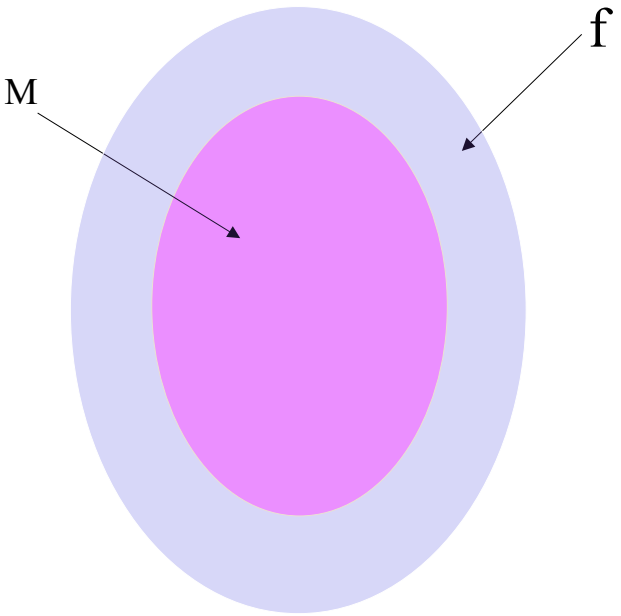
Especificar Establecer las propiedades f que debe satisfacer el sistema. Es común usar alguna variante de la *Lógica Temporal*

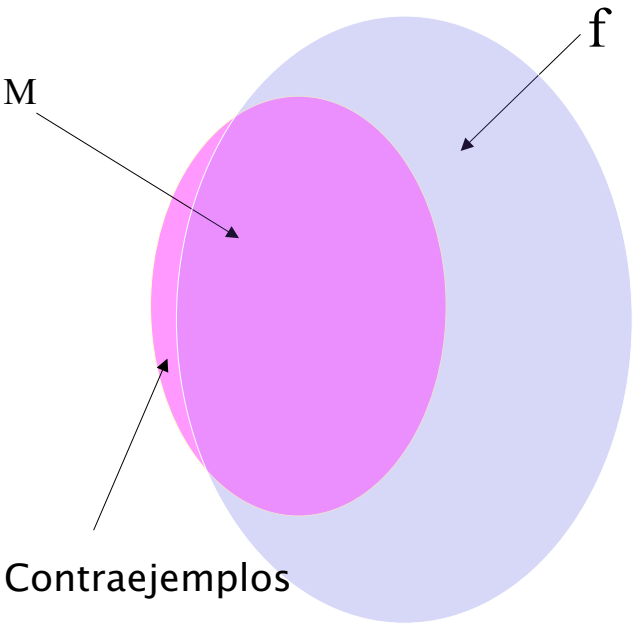
Verificar Utilizar algún algoritmo de Model Checking $\models$ para *comprobar* si el sistema diseñado es un *modelo* de la propiedad.

$$M \models f$$

Model Checking

- Los algoritmos de model checking se basan en una exploración exhaustiva de todo el *espacio de estados* generado por el modelo en busca de *trazas/contraejemplos* que violen la propiedad. $M \models f$???





Introducción

- Sistemas de Transición
- Estructuras de Kripke
- Lógica Temporal LTL
- Autómatas de Büchi
- Verificación Automática (Métodos on-the-fly)
- Complejidad
- Model Checking simbólico mediante BDDs
- Métodos de optimización: Reducción de Orden Parcial, Métodos de Compactación

Sistemas de Transición

La ejecución de un sistema secuencial o concurrente puede describirse mediante un *sistema de transición* como $M = (A, \Sigma, \overline{\longrightarrow}, s_0)$, donde

1. A es el conjunto de *acciones atómicas observables*
2. Σ es el conjunto de *estados*
3. $\overline{\longrightarrow} \subseteq \Sigma \times A \times \Sigma$ es la relación de transición etiquetada .

Denotamos $(s, a, s') \in \overline{\longrightarrow}$ con $s \xrightarrow{a} s'$

4. $s_0 \in \Sigma$ es el estado *inicial*

$\mathcal{O}(M) = \{x : x = s_0 \xrightarrow{a_0} \ldots \ es\ una\ traza\ maximal\}$ define la semántica basada en trazas del sistema de transición M

Sistemas de Transición

Características de la definición $M = (A, \Sigma, \overline{\longrightarrow}, s_0)$,

1. Las instrucciones atómicas se ejecutan secuencialmente, sigue el modelo de concurrencia entrelazada (interleaving)
2. Σ puede ser finito o infinito.
3. Modela el *indeterminismo*, cuando hay más de una transición *habilitada* se puede escoger cualquiera de ellas.
4. La definición puede extenderse fácilmente a un conjunto de estados iniciales $I \subseteq \Sigma$.
5. Una traza maximal no puede extenderse más. Incluye trazas infinitas, trazas que terminan con éxito, o con fallo (por ejemplo, debido a un bloqueo (deadlock)).

Sistemas de Transición: Ejemplos

División Entera

```
m1: coc := 0;
m2: resto := a;
m3: while resto >= coc do
m4:   coc := coc + 1;
m5:   resto := resto - b;
m6: end;
```

1. $\Sigma = Nat \times Nat \times \{m_1, \cdots, m_6\}$
2. $A = BoolExp \times Inst$
3. $s_0 = \langle -, -, m_1 \rangle$

$$\langle coc, resto, m_1 \rangle \xrightarrow{true, coc:=0} \langle 0, resto, m_2 \rangle$$
$$\langle 0, resto, m_2 \rangle \xrightarrow{true, resto:=a} \langle 0, a, m_3 \rangle$$
$$\langle coc, resto, m_3 \rangle \xrightarrow{resto>=coc, -} \langle coc, resto, m_4 \rangle$$
$$\langle coc, resto, m_3 \rangle \xrightarrow{resto<coc, -} \langle coc, resto, m_6 \rangle$$
$$\langle coc, resto, m_4 \rangle \xrightarrow{true, coc:=coc+1} \langle coc + 1, resto, m_5 \rangle$$
$$\langle coc, resto, m_5 \rangle \xrightarrow{true, resto:=resto-b} \langle coc, resto - b, m_3 \rangle$$

Sistemas de Transición: Ejemplos

Exclusión Mutua

```
Proceso1::
m1: while true do
m2:   {sección no crítica 1}
m3:   c1 := 0;
m4:   wait until c2 = 1;
m5:   {sección crítica 1}
m6:   c1 := 1
m7: end;
```

```
Proceso2::
n1: while true do
n2:   {sección no crítica 2}
n3:   c2 := 0;
n4:   wait until c1 = 1;
n5:   {sección crítica 2}
n6:   c2 := 1
n7: end;
```

Sistemas de Transición: Ejemplos

Exclusión Mutua II

1. $\Sigma = Bool \times Bool \times PC1 \times PC2$
2. $A = BoolExp \times Inst$
3. $s_0 = \langle 1, 1, m_1, n_1 \rangle$

```
Proceso1::
m1: while true do
m2:   {sección no crítica 1}
m3:   c1 := 0;
m4:   wait until c2 = 1;
m5:   {sección crítica 1}
m6:   c1 := 1
m7: end;
```

$$\langle c_1, c_2, m_1, n \rangle \xrightarrow{1,-} \langle c_1, c_2, m_2, n \rangle$$
$$\langle c_1, c_2, m_2, n \rangle \xrightarrow{1,snc1} \langle c_1, c_2, m_3, n \rangle$$
$$\langle c_1, c_2, m_3, n \rangle \xrightarrow{1,c1:=0} \langle 0, c_2, m_4, n \rangle$$
$$\langle c_1, c_2, m_4, n \rangle \xrightarrow{c2=1,-} \langle c_1, c_2, m_5, n \rangle$$
$$\langle c_1, c_2, m_5, n \rangle \xrightarrow{1,sc1} \langle c_1, c_2, m_6, n \rangle$$
$$\langle c_1, c_2, m_6, n \rangle \xrightarrow{1,c1:=1} \langle 1, c_2, m_1, n \rangle$$

Sistemas de Transición: Ejemplos

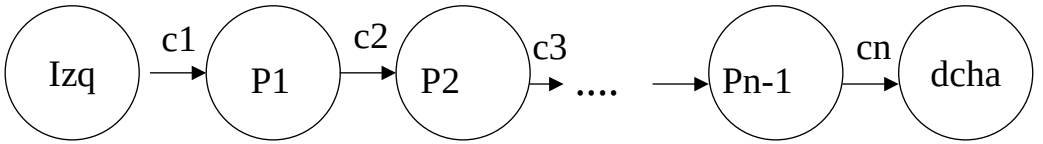
Exclusión Mutua III

```
Proceso2::
m1: while true do
m2:   {sección no crítica 2}
m3:   c2 := 0;
m4:   wait until c1 = 1;
m5:   {sección crítica 2}
m6:   c2 := 1
m7: end;
```

$$\langle c_1, c_2, m, n_1 \rangle \xrightarrow{1,-} \langle c_1, c_2, m, n_2 \rangle$$
$$\langle c_1, c_2, m, n_2 \rangle \xrightarrow{1,snc2} \langle c_1, c_2, m, n_3 \rangle$$
$$\langle c_1, c_2, m, n_3 \rangle \xrightarrow{1,c2:=0} \langle c_1, 0, m, n_4 \rangle$$
$$\langle c_1, c_2, m, n_4 \rangle \xrightarrow{c1=1,-} \langle c_1, c_2, m, n_5 \rangle$$
$$\langle c_1, c_2, m, n_5 \rangle \xrightarrow{1,sc2} \langle c_1, c_2, m, n_6 \rangle$$
$$\langle c_1, c_2, m, n_6 \rangle \xrightarrow{1,c2:=1} \langle c_1, 1, m, n_1 \rangle$$

Sistemas de Transición: Ejemplos

Criba de Eratóstenes



Sistemas de Transición: Ejemplos

```
cont := 2;
Izq::
m1: repeat
m2:   c1!cont;
m3:   cont := cont + 1;
m4: until cont > P;
```

```
Dcha::
n1: cn?mio;
n2: while true do
n3:   cn?otro
n4: end;
```

```
Pi:: (i=1,...,n-1)
mi1: ci-1?mio;
mi2: while true do
mi3:   ci-1?otro;
mi4:   if otro mod mio != 0
mi5:     then ci!otro;
mi6: end;
```

Sistemas de Transición: Ejemplos

Criba de Eratóstenes

$$\Sigma = Global \times EstIzda \times EstP_1 \times \cdots \times EstP_{n-1} \times EstDcha$$

$$Global = CanalNat \times \cdots \times CanalNat$$

$$EstIzda = PCIzda \times Nat$$

$$EstP_i = PCP_i \times Nat \times Nat \; (\forall i = 1, \cdots, n-1)$$

$$EstDcha = PCDcha \times Nat \times Nat$$

Sistemas de Transición: Grafos

Un sistema de transición $M = (A, \Sigma, \overline{\longrightarrow}, s_0)$ puede representarse de forma natural como un grafo dirigido (grafo de alcanzabilidad), donde

1. Los vértices son un subconjunto del conjunto de estados Σ.
2. El estado inicial s_0 se representa mediante un vértice con un flecha de entrada.
3. Existe una flecha del vértice $s \in \Sigma$ al vértice $s' \in \Sigma$, sii existe una transición $s \overline{\longrightarrow} s'$ en M.
4. Los vértices de Σ no alcanzables desde s_0 no están representados en el grafo.

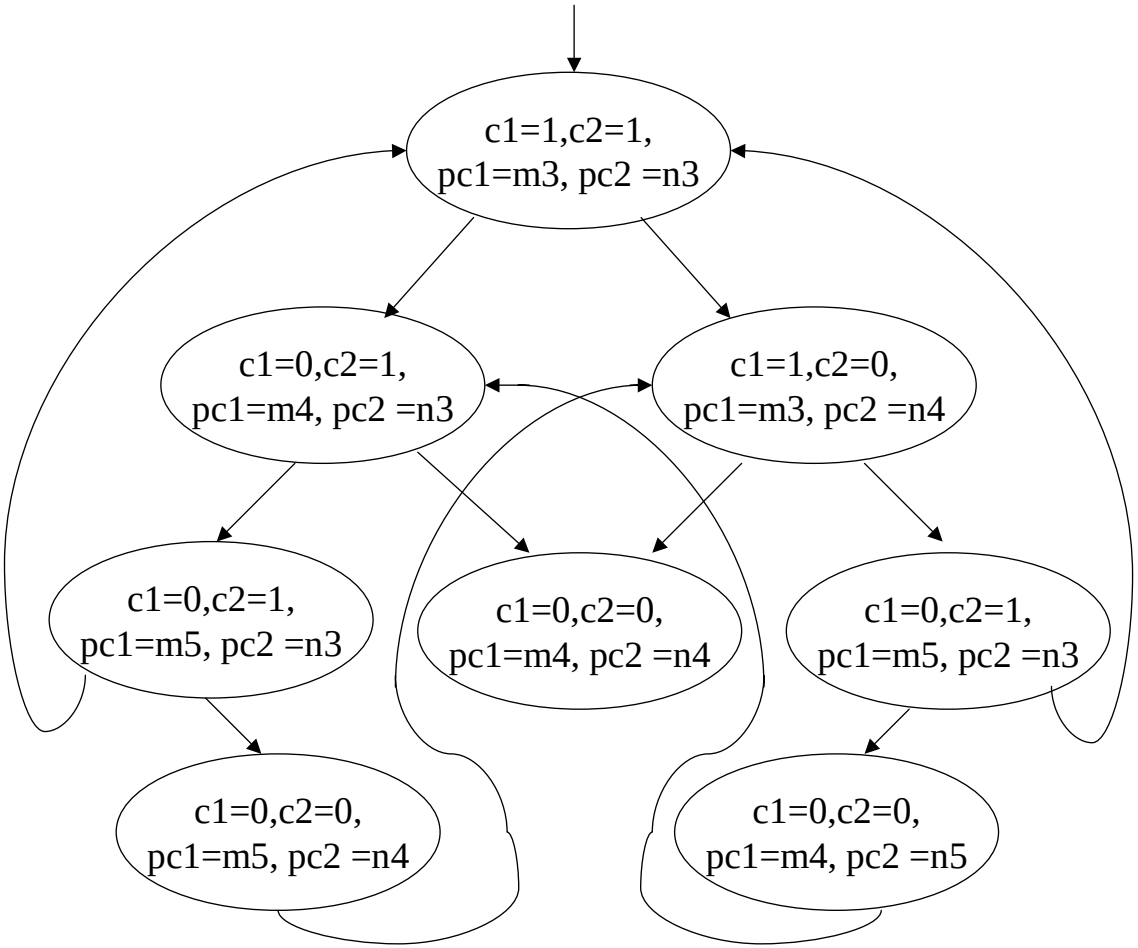
Grafo de alcanzabilidad: Ejemplo

Exclusión Mutua

```
Proceso1::
    while true do
      {sección no crítica 1}
m3:   c1 := 0;
m4:   wait until c2 = 1;
m5:   {sección crítica 1}
      c1 := 1
    end;
```

```
Proceso2::
    while true do
      {sección no crítica 2}
n3:   c2 := 0;
n4:   wait until c1 = 1;
n5:   {sección crítica 2}
      c2 := 1
    end;
```

Grafo de alcanzabilidad: Ejemplo



Estructuras de Kripke

Para evaluar propiedades sobre un sistema $M = (A, \Sigma, \overline{\longrightarrow}, s_0)$ construimos la estructura de Kripke $\mathcal{K} = \langle M, \tau \rangle$ con

1. Un conjunto AP de *proposiciones atómicas* que representan propiedades de los estados Σ de M.
2. Una función de evaluación $\tau : \Sigma \rightarrow 2^{AP}$ que asigna a cada estado el conjunto de propiedades que satisface.

Definiciones alternativas

3. Σ se representa mediante $\tau(\Sigma)$.
4. $\tau : AP \rightarrow 2^{\Sigma}$ asocia cada proposición con el conjunto de estados que la satisfacen.

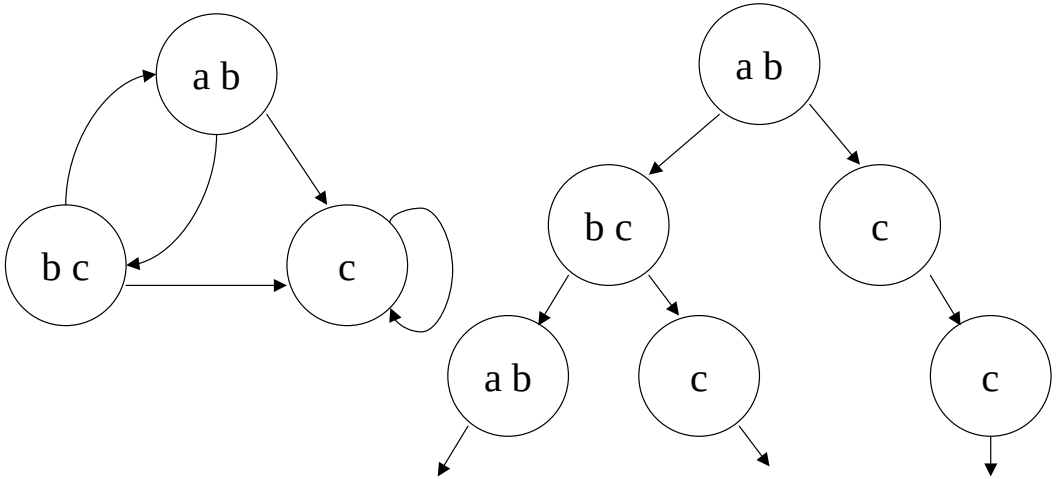
Lógica Temporal

- Es un formalismo para describir "secuencias" de ejecución de un sistema.
- No se hace mención explícita del tiempo.
- Especifican propiedades que utilizan expresiones como *eventualmente* o *nunca*.
- Su semántica hace uso de las estructuras de Kripke.
- Existen distintas lógicas temporales, por ejemplo:
  - CTL* Computational Tree Logic (ramificada),
  - CTL Computational Tree Logic (ramificada)
  - LTL Linear Temporal Logic (lineal)

Lógica Temporal: CTL*

- Describe árboles de computación.
- Un árbol se construye designando un estado inicial $s \in \Sigma$ de una estructura de Kripke y desplegando las transiciones en un árbol que tiene como raíz s.

Lógica Temporal: CTL*



Lógica Temporal: CTL

- Las fórmulas CTL* se construyen con las proposiciones de AP y con cuantificadores de caminos, Operadores Temporales y Operadores booleanos.

cuantificadores de caminos	
$\forall$	*Para todos los caminos*
$\exists$	*Para algún camino*

Operadores booleanos	
$\wedge$	*Conjunción*
$\vee$	*Disyunción*
$\neg$	*Negación*

Operadores Temporales	
$\bigcirc$	*En el siguiente estado*
$\Diamond$	*En algún estado futuro*
$\Box$	*En todos los estados futuros*
U	*Operador "until"*
V	*Operador "release"*

Lógica Temporal: CTL*

Dados $\mathcal{K} = \langle M, \tau \rangle$, $p \in AP$, y g_1, g_2 fórmulas de estado (sin operadores temporales ni cuantificadores) y f_1, f_2 fórmulas CTL*

1. $M, s \models p \quad \Leftrightarrow \quad p \in \tau(s)$
2. $M, s \models \neg g_1 \quad \Leftrightarrow \quad M, s \not\models g_1$
3. $M, s \models f g_1 \vee g_2 \quad \Leftrightarrow \quad M, s \models g_1$ o $M, s \models g_2$
4. $M, s \models g_1 \wedge g_2 \quad \Leftrightarrow \quad M, s \models g_1$ y $M, s \models g_2$
5. $M, s \models \exists f_1 \quad \Leftrightarrow \quad$ hay un camino π que comienza en s tal que $M, \pi \models f_1$
6. $M, s \models \forall f_1 \quad \Leftrightarrow \quad$ para todos los caminos π que comienzan en s $M, \pi \models f_1$
7. $M, s.\pi' \models g_1 \quad \Leftrightarrow \quad M, s \models g_1$
8. $M, \pi \models \neg f_1 \quad \Leftrightarrow \quad M, \pi \not\models f_1$
9. $M, \pi \models f_1 \vee f_2 \quad \Leftrightarrow \quad M, \pi \models f_1$ o $M, \pi \models f_2$

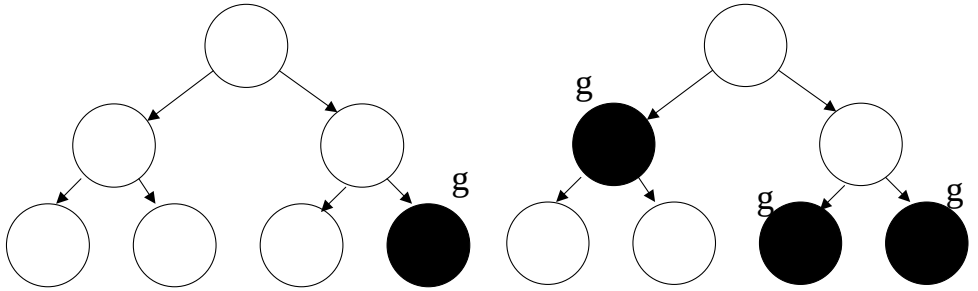
Lógica Temporal: CTL*

Dados $\mathcal{K} = \langle M, \tau \rangle$, $p \in AP$, y f_1, f_2 fórmulas CTL*

10. $M, \pi \models f_1 \wedge f_2 \quad \Leftrightarrow \quad M, \pi \models f_1$ y $M, \pi \models f_2$
11. $M, \pi \models \bigcirc f_1 \quad \Leftrightarrow \quad M, \pi^1 \models f_1$
12. $M, \pi \models \Diamond f_1 \quad \Leftrightarrow \quad \exists k \geq 0. M, \pi^k \models f_1$
13. $M, \pi \models \Box f_1 \quad \Leftrightarrow \quad \forall k \geq 0. M, \pi^k \models f_1$
14. $M, \pi \models f_1 U f_2 \quad \Leftrightarrow \quad \exists k \geq 0. M, \pi^k \models f_2$ y $\forall 0 \leq i < k. M, \pi^i \models f_1$
15. $M, \pi \models f_1 V f_2 \quad \Leftrightarrow \quad \forall i \geq 0. M\pi^i \models g_2$ o
    $\exists j \geq 0. M, \pi^j \models f_1$ y $\forall i < j. M, \pi^i \models f_2$

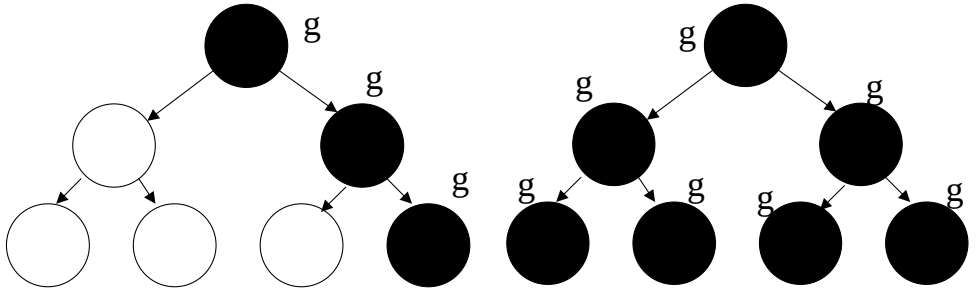
Lógica Temporal: CTL*

$M, s \models \exists \Diamond g$ $\quad$ $M, s \models \forall \Diamond g$



Lógica Temporal: CTL*

$M, s \models \exists \Box g$ $\quad$ $M, s \models \forall \Box g$



Lógica Temporal: CTL

CTL es un subconjunto de CTL* en el que todas las fórmulas se construyen como sigue:

- Los operadores temporales $\bigcirc$, $\Diamond$, $\Box$, U y V siempre están precedidos por un cuantificador de camino: $\forall$, $\exists$.
- Si f y g son fórmulas de estado entonces $\bigcirc f$, $\Diamond f$, $\Box f$, fUg, fVg son fórmulas de camino.

- Sólo se permite un nivel de anidamiento de los operadores temporales

Lógica Temporal: LTL

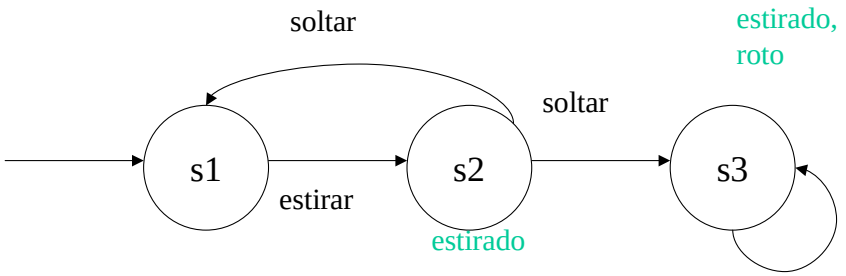
LTL es un subconjunto de CTL* en el que todas las fórmulas se construyen como sigue:

- Todas las fórmulas LTL están cuantificadas universalmente ($\forall$)
- Si $p \in AP$, entonces p es una fórmula de camino
- Si f y g son fórmulas de camino, entonces $\bigcirc f$, $\Diamond f$, $\Box f$, fUg, fVg son fórmulas de camino
- Las fórmulas LTL se evalúan sobre trazas de ejecución

Lógica Temporal LTL: Ejemplo

Un modelo de un elástico

$AP = \{estirado, roto\}$



- $\pi_0 = s_1 s_2 s_1 s_2 s_1 \cdots$
- $\pi_1 = s_1 s_2 s_3 s_3 s_3 \cdots$
- $\pi_2 = s_1 s_2 s_1 s_2 s_3 \cdots$

$\cdots$

- $\pi_2 \not\models estirado$
- $\pi_2 \models \bigcirc estirado$
- $\pi_2 \not\models \bigcirc \bigcirc estirado$
- $\pi_2 \models \Diamond estirado$
- $\pi_2 \not\models \Box estirado$
- $\pi_2 \models \Diamond \Box estirado$

- $M \models \Diamond estirado$
- $M \models \Box(\neg estirado \rightarrow \bigcirc estirado)$
- $M \not\models \Diamond \Box estirado$
- $M \not\models \neg \Diamond \Box estirado$
- $M \not\models \Box(estirado \rightarrow \bigcirc \neg estirado)$
- $\pi_2 \not\models (\neg estirado) U\, roto$

Lógica Temporal LTL: Ejemplo

Un modelo de un semáforo

$$verde \rightarrow amarillo \rightarrow rojo \rightarrow verde$$

$AP = \{gr, re, ye\}$

- Seguridad: $\Box(\neg(gr \wedge ye) \wedge \neg(gr \wedge re) \wedge \neg(ye \wedge re) \wedge (ye \vee gr \vee re))$

  En cada momento el semáforo está en uno y sólo uno de los tres colores

- Viveza: $\Box(gr \rightarrow (grUye))$

  Pasa del color verde al amarillo

- Seguridad: $\Box((grUye) \vee (yeUre) \vee (reUgr))$

  El semáforo cambia de color correctamente

Lógica Temporal LTL: Ejemplo

El semáforo modificado

$$verde \rightarrow amarillo \rightarrow rojo \rightarrow amarillo \rightarrow verde$$

- Seguridad: $\Box(((gr \vee re)Uye) \vee (yeU(gr \vee re))$

NO modela el cambio de color del semáforo

$$verde \rightarrow rojo \rightarrow verde$$

- Seguridad:

$\Box((gr \rightarrow (gr\, U(ye \wedge (ye\, Ure))))$
$\wedge(re \rightarrow (re\, U(ye \wedge (yeU\, gr))))$
$\wedge(ye \rightarrow (ye\, U(gr \vee re))))$

Autómatas de Büchi

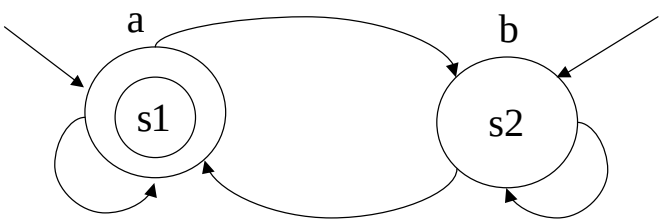
Un autómata de Büchi $\mathcal{A}$ es una 6-upla $\langle \Sigma, S, \triangle, I, L, F \rangle$ donde

- Σ es un *alfabeto* finito.
- S es un conjunto finito de *estados*.
- $\triangle \subseteq S \times S$ es la *relación de transición*.
- $I \subseteq S$ son los *estados iniciales*.
- $L : S \rightarrow \Sigma$ *etiqueta* a los estados.
- $F \subseteq S$ es el conjunto de *estados de aceptación*.

Autómatas de Büchi como grafos

- Marcamos los estados iniciales que una flecha de entrada,
- Marcamos los estados de aceptación con un doble círculo.
- Ejemplo

$\Sigma = \{a, b\}$, $S = \{s_1, s_2\}$, $I = \{s_1, s_2\}$, $L(s_1) = a$ y $L(s_2) = b$, $F = \{s_1\}$



Lenguajes aceptados por Autómatas de Büchi

Una *ejecución* ρ de $\mathcal{A}$ en v es

- un camino infinito en el grafo del autómata a partir de un estado inicial,
- las etiquetas de los nodos por los que pasa se corresponden con las letras de v.
- decimos que v es una *entrada* del autómata o que $\mathcal{A}$ *lee* v.

Formalmente, dada $v \in \Sigma^\omega$ representada como

$v : Nat \rightarrow \Sigma \; (\; v = v(0)v(1)v(2)\ldots)$

Una ejecución de $\mathcal{A}$ sobre v es una aplicación $\rho : Nat \rightarrow S$ tal que

- $\rho(0) \in I$. El primer estado es un estado inicial.
- Dado $i \geq 0$, $(\rho(i), \rho(i+1)) \in \triangle$.
- $v(i) = L(\rho(i))$.

Lenguajes aceptados por Autómatas de Büchi

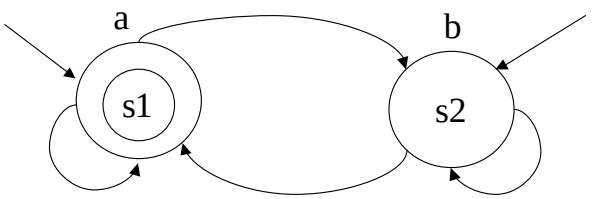
- Sea $inf(\rho)$ el conjunto de los estados que aparecen con infinita frecuencia en ρ ($inf(\rho)$ es un conjunto finito).
- Una ejecución ρ de un autómata de Büchi $\mathcal{A}$ es de *aceptación* cuando

$$inf(\rho) \cap F \neq \emptyset$$

es decir, cuando algún estado de aceptación aparece un número infinito de veces en ρ.

El *lenguaje* $\mathcal{L}(A) \subseteq \Sigma^\omega$ de un autómata de Büchi $\mathcal{A}$ es el conjunto de todas las ejecuciones ρ *aceptadas* por $\mathcal{A}$

Ejemplo



- $a^\omega = aaaa\cdots \in \mathcal{L}(\mathcal{A})$

$\rho = s_1 s_1 s_1 \cdots$

- $b^\omega \notin \mathcal{L}(\mathcal{A})$
- $(ab)^\omega \in \mathcal{L}(\mathcal{A})$
- $\mathcal{L}(\mathcal{A}) = \{(b^*a)^\omega\}$

Especificación

Un autómata de Büchi

$$\mathcal{A} = \langle 2^{AP}, S, \triangle, I, L, F \rangle$$

representa un sistema del modo siguiente

- S es el conjunto de estados del sistema;
- la *relación sucesor* entre estados es la relación de transición $\triangle \subseteq S \times S$
- el conjunto de estados iniciales es $I \subseteq S$
- Dado AP *un conjunto finito de proposiciones atómicas*, la función

etiquetado

$$L : S \to 2^{AP}$$

asocia cada estado s con el subconjunto $L(s) \subseteq AP$ de proposiciones que se *satisfacen* en s

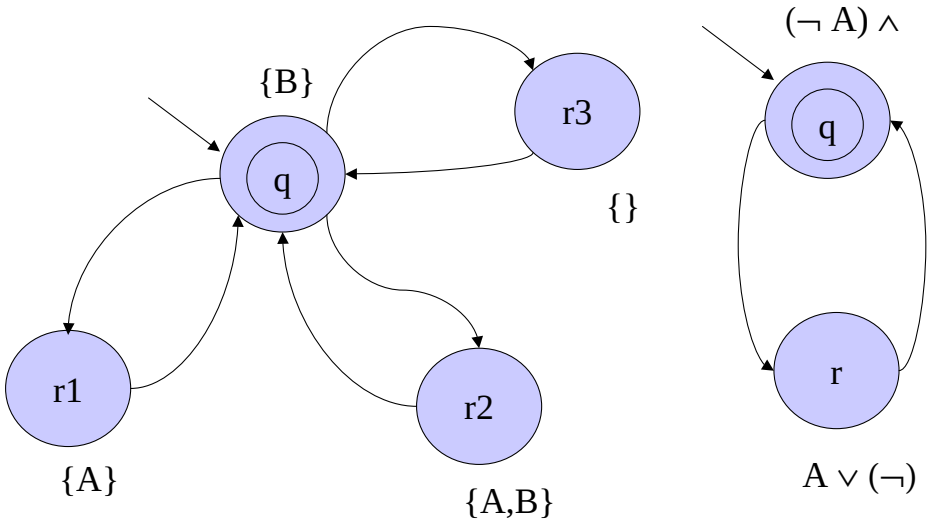
- Cuando el autómata es el modelo, normalmente *todos* los estados son estados de aceptación

Autómatas de Büchi no deterministas

- Un autómata es *no-determinista*, si *es posible transitar a más de un estado* para leer la siguiente letra de entrada
- si $\exists r_1, r_2 \in S$ tales que $L(r_1) = L(r_2)$ y
  (1) $r_1, r_2 \in I$, o
  (2) $\exists (s, r_1), (s, r_2) \in \triangle$

- Los autómatas de Büchi no deterministas pueden tener más de una ejecución para una entrada dada
(es suficiente con que una de las ejecuciones de v sea de aceptación, para v esté en $\mathcal{L}(A)$

Expresiones lógicas como etiquetas

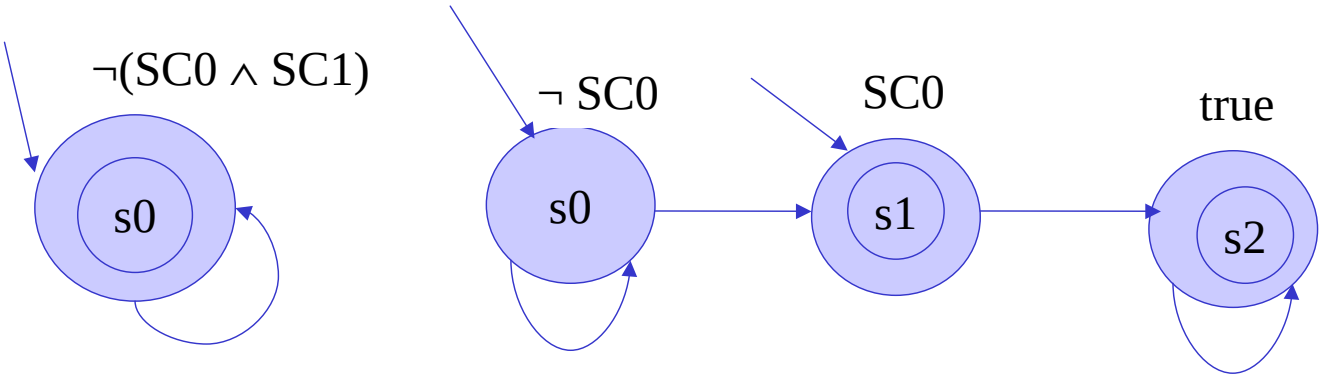


- Los estados cuyos estados predecesores y sucesores coinciden pueden combinarse en un mismo estado produciendo una representación más eficiente
- La función etiqueta del nuevo autómata es $L : S \rightarrow 2^{2^{AP}}$
- Cada $L(s) \subseteq AP$ se corresponde de forma unívoca una fórmula lógica φ_s

- $\Sigma = 2^{AP}$, y $v(i) = L(\rho(i))$ se escribe como $v(i) \models L(\rho(i))$

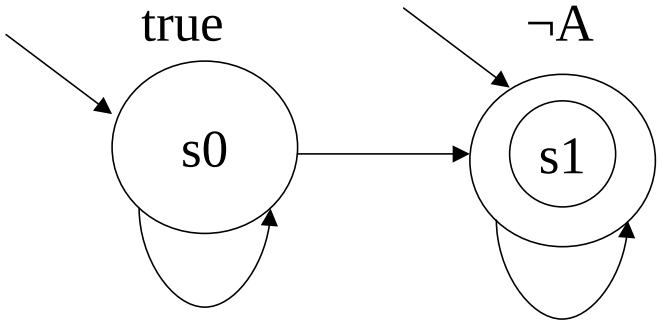
Ejemplos

- Exclusión Mutua : $\Box\neg(SC_1 \wedge SC_2)$
- Propiedad de viveza : $\Diamond SC_1$



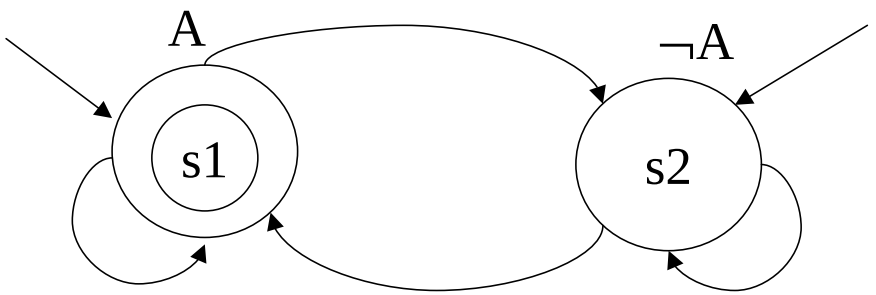
Ejemplos

- Un autómata que acepta un número finito de A (Autómata no determinista que no puede expresarse de forma determinista)



Ejemplos

- Un autómata determinista que acepta un número infinito de A



Verificación Automática

- Objetivo: Analizar la corrección del software
- Restricción: Problemas indecidibles
- Propuestas
  - Restringirse a un conjunto de programas

    Por ejemplo, el model checking se aplica a sistemas

    que generan un número finito de estados.
  - Analizar sólo parte del sistema
  - Aplicar técnicas de abstracción: automática o manualmente
  - Combinar técnicas automáticas con otras que utilizan asistencia humana

    Demostradores de Teoremas

Búsqueda en el Espacio de Estados

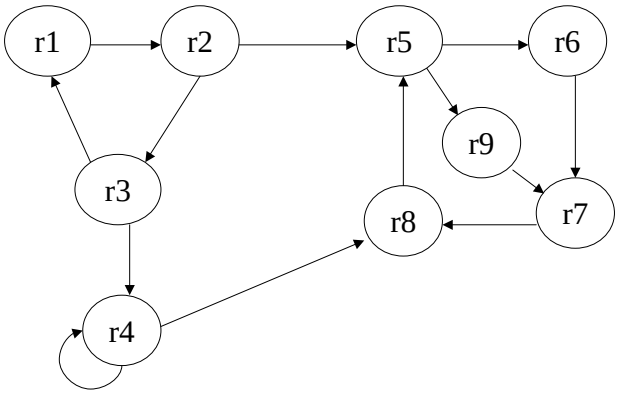
Suponemos que S es finito

Algoritmo de Recorrido del Espacio de Búsqueda

```
1 nuevos = I
2 visitados = {}
3 mientras nuevos != {} hacer
4   escoger algún estado s de nuevos
5   nuevos = nuevos - {s}
6   visitados = visitados + {s}
7   para cada transición t habilitada para s hacer
8     aplicar t a s, y obtener s'
9     si s' no está ni en nuevos ni en visitados entonces
10       nuevos = nuevos + {s'}
```

Búsqueda en Amplitud

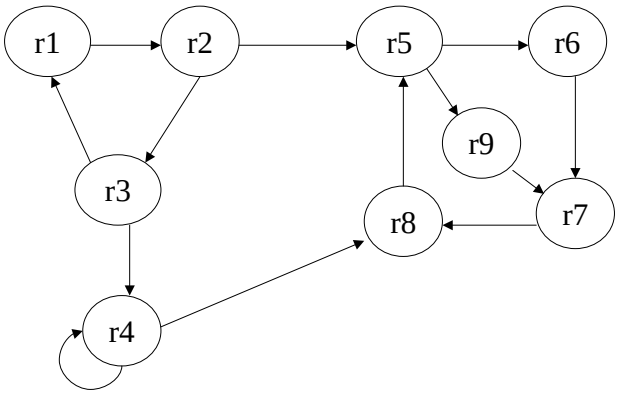
Usando una cola fifo



nuevos	s	visitados
$\langle r_1 \rangle$	r_1	$\langle\rangle$
$\langle r_2 \rangle$	r_2	$\langle r_1 \rangle$
$\langle r_3, r_5 \rangle$	r_3	$\langle r_1, r_2 \rangle$
$\langle r_5, r_4 \rangle$	r_5	$\langle r_1, r_2, r_3 \rangle$
$\langle r_4, r_9, r_6 \rangle$	r_4	$\langle r_1, r_2, r_3, r_5 \rangle$
$\langle r_9, r_6, r_8 \rangle$	r_9	$\langle r_1, r_2, r_3, r_5, r_4 \rangle$
$\langle r_6, r_8, r_7 \rangle$	r_6	$\langle r_1, r_2, r_3, r_5, r_4, r_9 \rangle$
$\langle r_8, r_7 \rangle$	r_8	$\langle r_1, r_2, r_3, r_5, r_4, r_9, r_6 \rangle$
$\langle r_7 \rangle$	r_7	$\langle r_1, r_2, r_3, r_5, r_4, r_9, r_6, r_8 \rangle$
$\langle\rangle$		$\langle r_1, r_2, r_3, r_5, r_4, r_9, r_6, r_8, r_7$

Búsqueda en Profundidad

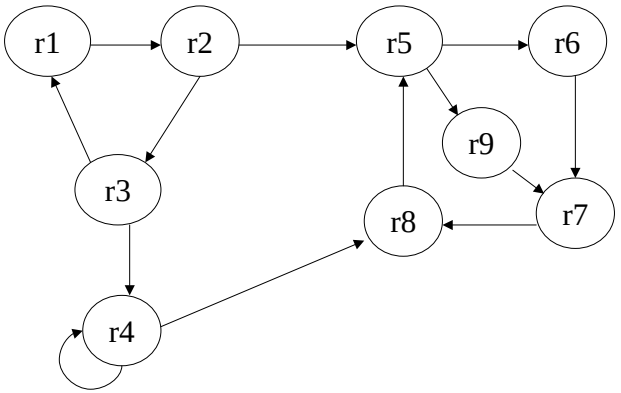
Usando una pila lifo



nuevos	s	visitados
$\langle r_1 \rangle$	r_1	$\langle\rangle$
$\langle r_2 \rangle$	r_2	$\langle r_1 \rangle$
$\langle r_3, r_5 \rangle$	r_3	$\langle r_1, r_2 \rangle$
$\langle r_4, r_5 \rangle$	r_4	$\langle r_1, r_2, r_3 \rangle$
$\langle r_8, r_5 \rangle$	r_8	$\langle r_1, r_2, r_3, r_4 \rangle$
$\langle r_5 \rangle$	r_5	$\langle r_1, r_2, r_3, r_4, r_8 \rangle$
$\langle r_9, r_6 \rangle$	r_9	$\langle r_1, r_2, r_3, r_4, r_8, r_5 \rangle$
$\langle r_7, r_6 \rangle$	r_7	$\langle r_1, r_2, r_3, r_4, r_8, r_5, r_9 \rangle$
$\langle r_6 \rangle$	r_6	$\langle r_1, r_2, r_3, r_4, r_8, r_5, r_9, r_7 \rangle$
$\langle\rangle$		$\langle r_1, r_2, r_3, r_4, r_8, r_5, r_9, r_7, r_6 \rangle$

Contraejemplos

Invariante: $\Box\varphi$



Los estados iniciales satisfacen φ

- Cada estado añadido a `nuevos` satisface φ
- Cuando un estado alcanzable no satisface φ, es interesante conocer cómo se ha generado
- Para recuperar el camino basta con añadir a cada estado un enlace que apunta a su único predecesor
- Por ejemplo, si $r_9 \not\models \varphi$ entonces la traza de error es $\mathtt{r}_1, \mathtt{r}_2, \mathtt{r}_5, \mathtt{r}_9$

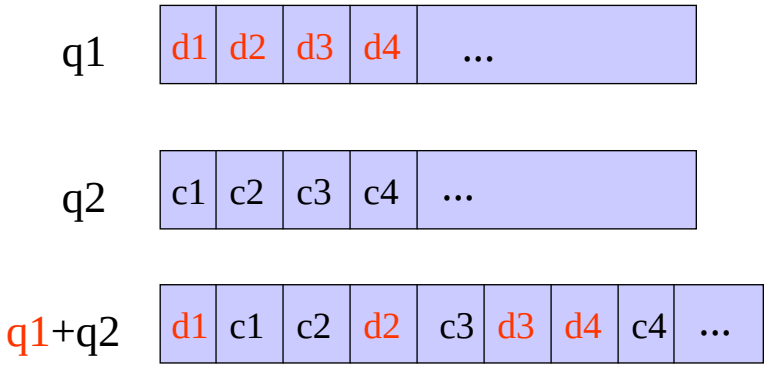
Representación de los Estados

- La representación de los estados es crucial en los algoritmos de verificación automática.
- Necesitamos registrar información sobre
  - los estados que suceden a uno dado siguiendo la relación de transición
  - el valor de las variables del programa
  - el contador del programa
  - el contenido de las colas de mensajes

- Necesitamos poder decidir si un estado alcanzado es igual o no a otro alcanzado previamente (sino los algoritmos de model checking no terminarían).
- Los programas que utilizan pilas, árboles, ..., producen un espacio de estados demasiado grande para poder automatizar el análisis.
- Se utilizan tablas hash para acelerar el acceso a los mismos y técnicas de comprensión para minimizar el espacio utilizado.

Representación de los Estados

- No debemos distinguir entre estado *esencialmente* iguales



Si |q1| = n, |q2| = m entonces hay (m+n)!/(m!*n!) configuraciones de q1 + q2 diferentes, pero esencialmente iguales

Por ejemplo, si n = 5 y m = 5, hay 252 estados equivalentes

Model Checking basado en Autómatas

- Sean $\mathcal{A}$ y $\mathcal{B}$ dos autómatas sobre el mismo alfabeto que representan, respectivamente, un modelo y una propiedad
- Para comprobar si el modelo satisface la propiedad debemos analizar si

$$\mathcal{L}(A) \subseteq \mathcal{L}(B)$$

- Sea $\overline{\mathcal{L}(B)} = \Sigma^w \setminus \mathcal{L}(B)$, las palabras *no aceptadas* por $\mathcal{B}$,
- La inclusión puede reescribirse como

$$\mathcal{L}(A) \cap \overline{\mathcal{L}(B)} = \emptyset$$

Interpretación de los resultados

- Los resultados del proceso de verificación automática tienen que evaluarse cuidadosamente
- Si $\mathcal{A}$ *sub-aproxima* el sistema modelado puede ser que

$$\mathcal{L}(A) \subseteq \mathcal{L}(B)$$

aunque el sistema original no satisfaga $\mathcal{B}$.

- Si $\mathcal{A}$ *sobre-aproxima* el sistema puede ser que

$$\mathcal{L}(A) \not\subseteq \mathcal{L}(B)$$

aunque el sistema original sí satisface $\mathcal{B}$.

- Como comparar un contraejemplo con el sistema modelado es sencillo, normalmente se considera de más interés sobre-aproximar el modelo que sub-aproximarlo

Model Checking

Estrategia de model checking

$$\mathcal{L}(A) \cap \overline{\mathcal{L}(B)} = \emptyset$$

1. Complementar $\mathcal{B}$, es decir, construir un autómata $\overline{\mathcal{B}}$ que reconoce el lenguaje $\overline{\mathcal{L}(B)}$.
2. Intersecar $\mathcal{A}$ y $\overline{\mathcal{B}}$
   - Si la intersección es vacía $\mathcal{A}$ satisface $\mathcal{B}$.
   - En otro caso, utilizar la palabra aceptada por la intersección como *contraejemplo*.

Intersección de Autómatas

- Dados $\mathcal{A}_1 = \langle \Sigma, S_1, \triangle_1, I_1, L_1, F_1 \rangle$ y $\mathcal{A}_2 = \langle \Sigma, S_2, \triangle_2, I_2, L_2, F_2 \rangle$ el autómata intersección

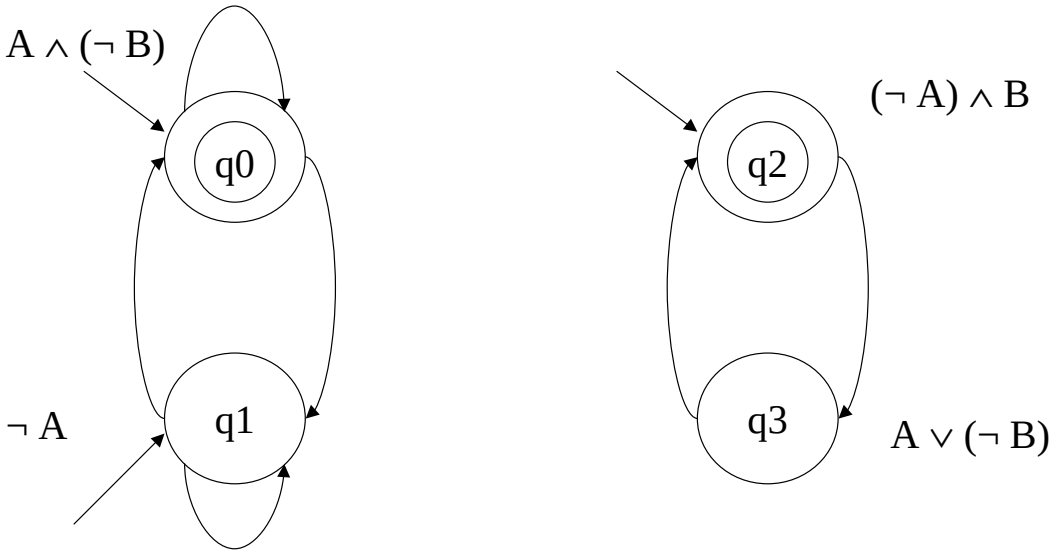
$$\boxed{\mathcal{A}_1 \cap \mathcal{A}_2}$$

debe aceptar aquellas palabras aceptadas por ambos autómatas.

- Cada ejecución del autómata intersección tiene que simular simultáneamente dos ejecuciones de la entrada, una del autómata $\mathcal{A}_1$ y otra de $\mathcal{A}_2$

(Por el momento ignoramos las condiciones de aceptación)

Intersección de Autómatas: Ejemplo



Intersección de Autómatas: Ejemplo

Los estados y las etiquetas

$$L(\langle q_0, q_2\rangle) = L_1(q_0) \wedge L_2(q_2) = A \wedge (\neg B) \wedge (\neg A) \wedge B \quad = false$$

$$L(\langle q_0, q_3\rangle) = L_1(q_0) \wedge L_2(q_3) = A \wedge (\neg B) \wedge (A \vee (\neg B)) \quad = A \wedge (\neg B)$$

$$L(\langle q_1, q_2\rangle) = L_1(q_1) \wedge L_2(q_2) = (\neg A) \wedge (\neg A) \wedge B \quad = (\neg A) \wedge B$$

$$L(\langle q_1, q_3\rangle) = L_1(q_1) \wedge L_2(q_3) = (\neg A) \wedge (A \vee (\neg B) \quad = (\neg A) \wedge (\neg B)$$

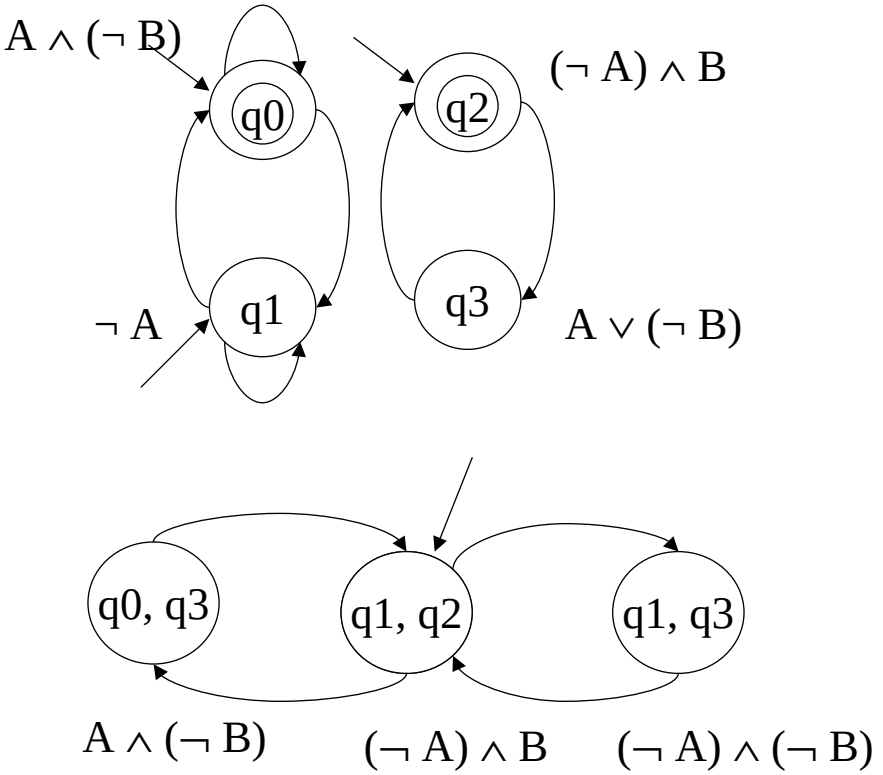
Intersección de Autómatas: Ejemplo

Las flechas

- Hay una flecha entre dos estados $\langle q, r \rangle$ y $\langle q', r' \rangle$ en el autómata intersección sii hay una flecha de q a q' en $\mathcal{A}_1$ y una flecha de r a r' en $\mathcal{A}_2$

Intersección de Autómatas: Ejemplo

Las flechas



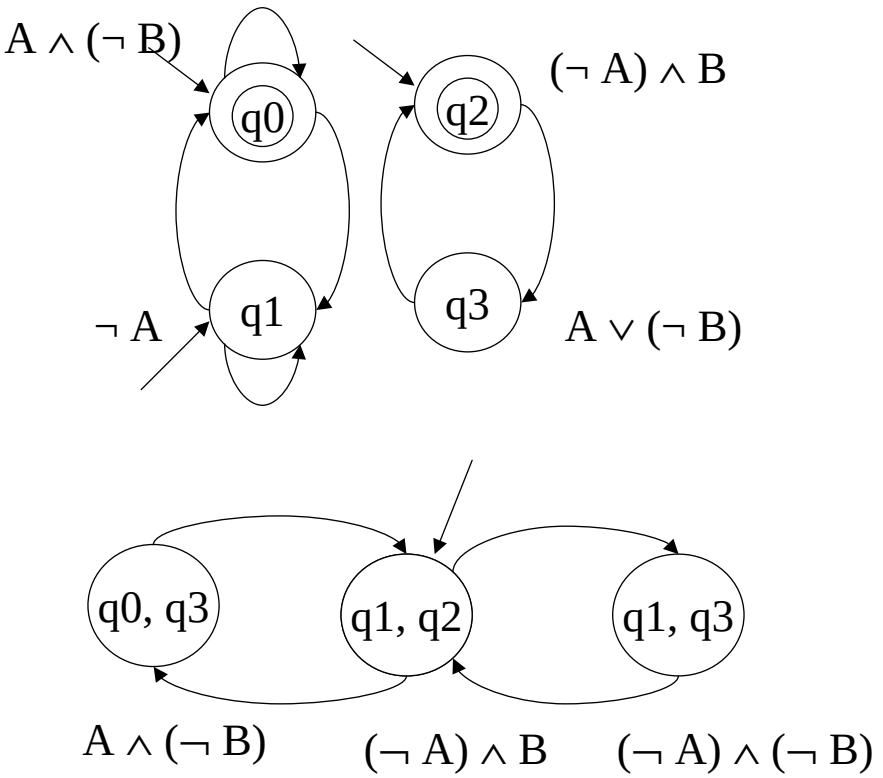
Intersección de Autómatas: Ejemplo

Los estados iniciales

- Los estados iniciales son aquellos pares cuyas componentes son estados iniciales en ambos autómatas.

Intersección de Autómatas: Ejemplo

Los estados iniciales



Intersección de Autómatas

Estados de Aceptación

- Propuesta 1:Un estado $\langle q, r \rangle$ es de aceptación si $q \in F_1$ o $r \in F_2$.

  No es aceptable: Una ejecución válida podría no pasar infinitas veces por ningún estado de aceptación de alguno de los autómatas intersecados.
- Propuesta 2:Un estado $\langle q, r \rangle$ es de aceptación si $q \in F_1$ y $r \in F_2$.

  No es aceptable: Una ejecución válida tiene que alcanzar lo estados de aceptación simultáneamente.

Intersección de Autómatas

$\langle \Sigma, S, \triangle, I, L, ?? \rangle$

$\mathcal{A}_1 \cap \mathcal{A}_2 = \langle \Sigma, S', \triangle', I, L, F' \rangle$

(1) Definimos $S' = (S \times \{1\} \cup S \times \{2\})$

(2) Definimos $\triangle'$ como

$(\langle q, i \rangle, \langle q', i \rangle) \in \triangle'$ sii $(q, q') \in \triangle$ y $q \notin F_i$.

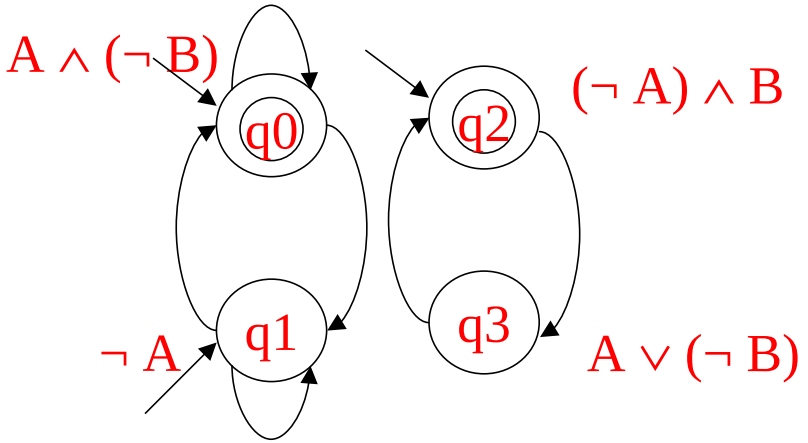
$(\langle q, i \rangle, \langle q', (i mod 2) + 1 \rangle) \in \triangle'$ sii $(q, q') \in \triangle$ y $q \in F_i$.

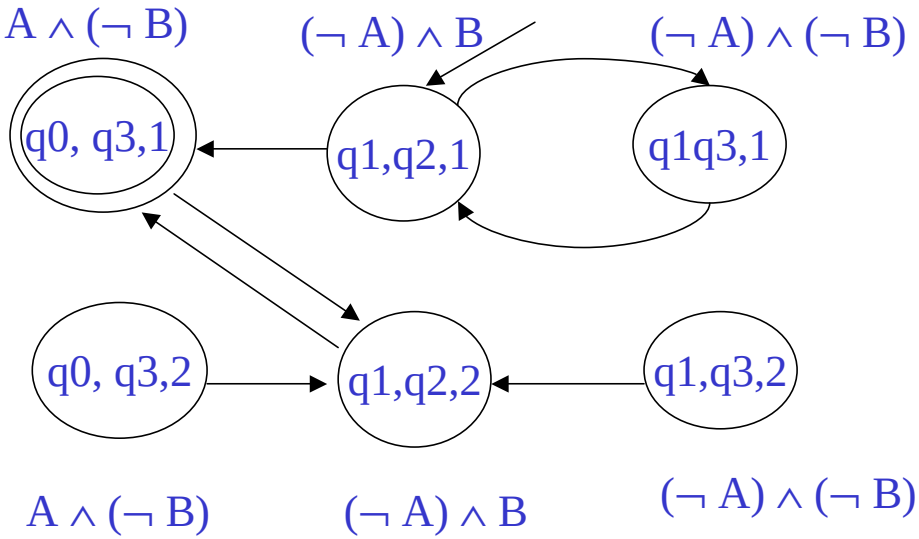
(3) Definimos $F' = F_1 \times S_2$

- Cuando $F_1 = S_1$ entonces el proceso se simplifica y

$$\mathcal{A}_1 \cap \mathcal{A}_2 = \langle \Sigma, S_1 \times S_2, \triangle, I, L, S_1 \times F_2 \rangle$$

Intersección de Autómatas: Ejemplo





Autómatas Complemento

$$\mathcal{L}(\mathcal{A} \cap \overline{\mathcal{B}}) = \emptyset$$

- Dado $\mathcal{B}$ construir $\overline{\mathcal{B}}$ es muy complicado y costoso.
- Una alternativa es

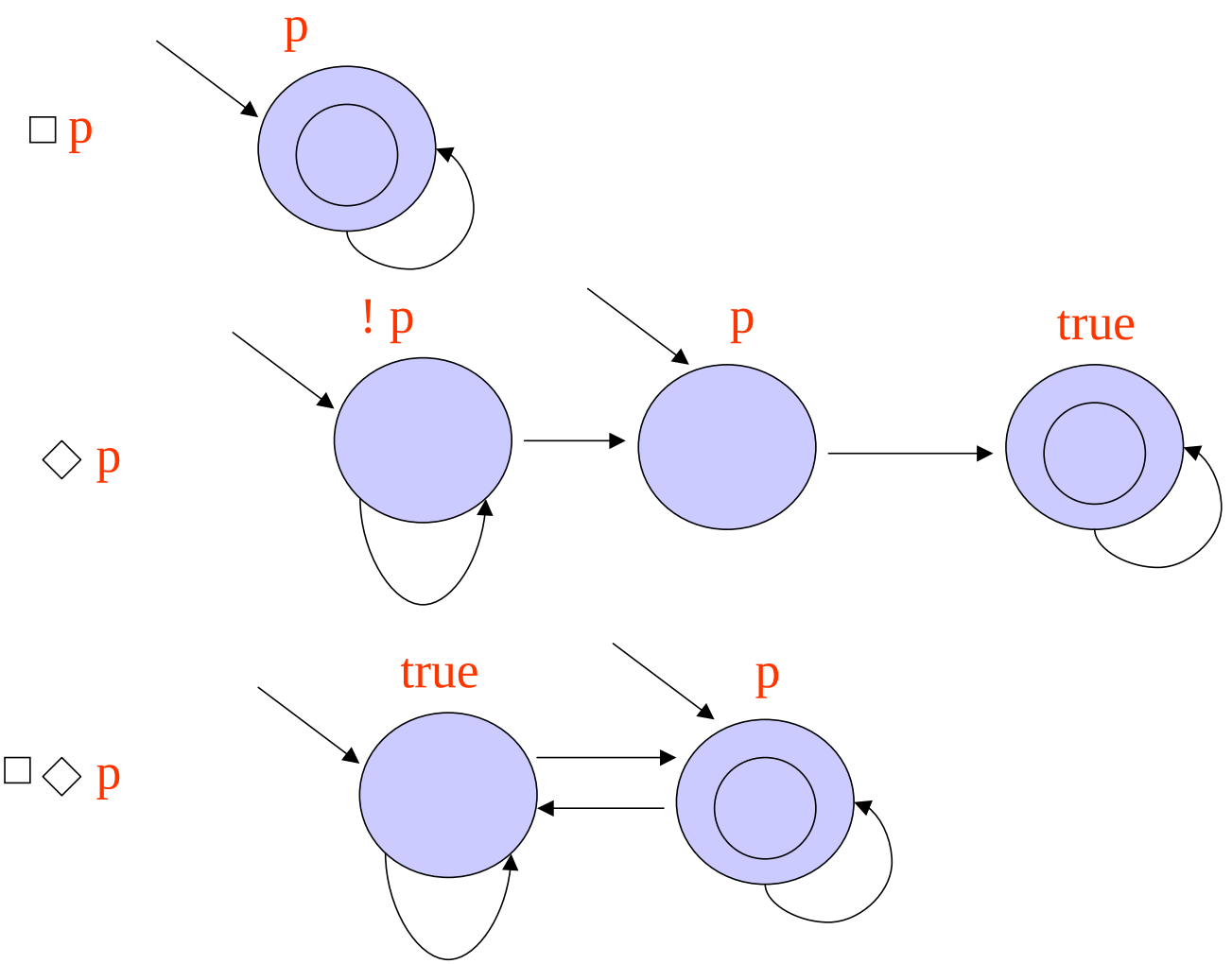
  (1) especificar la propiedad mediante una fórmula LTL φ

  (2) negar la fórmula y obtener $\neg\varphi$

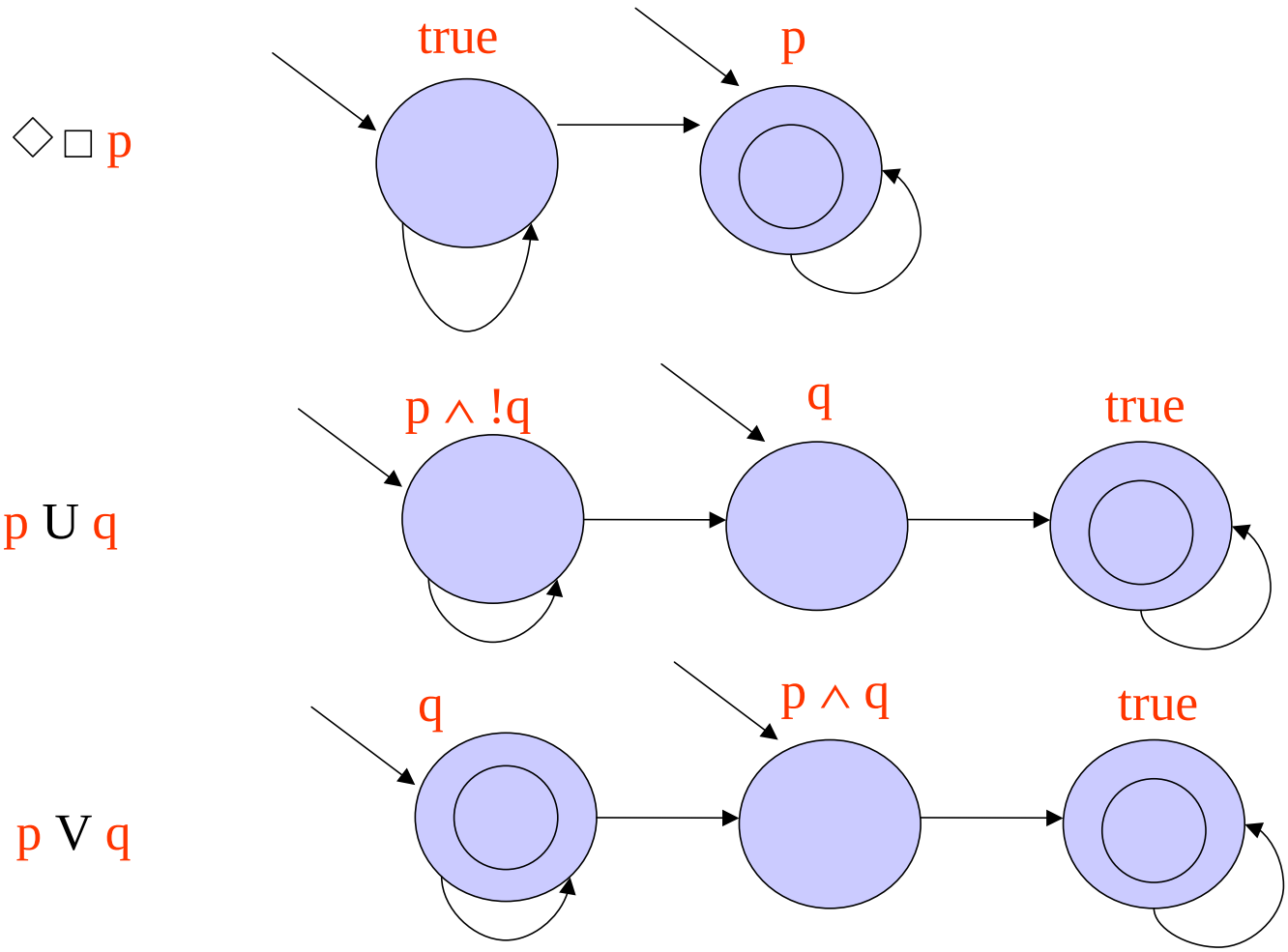
  (3) construir el autómata $\mathcal{A}_{\neg\varphi}$ que representa $\neg\varphi$

  (4) $\mathcal{L}(A_{\neg\varphi}) = \overline{\mathcal{L}(A_{\varphi})}$

De LTL a Autómatas

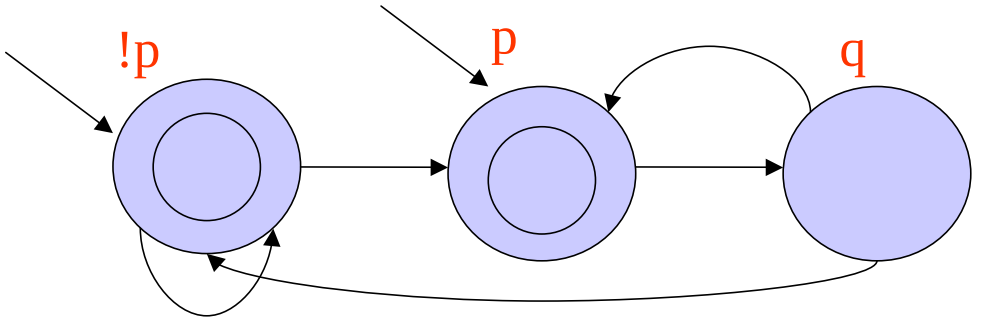


De LTL a Autómatas

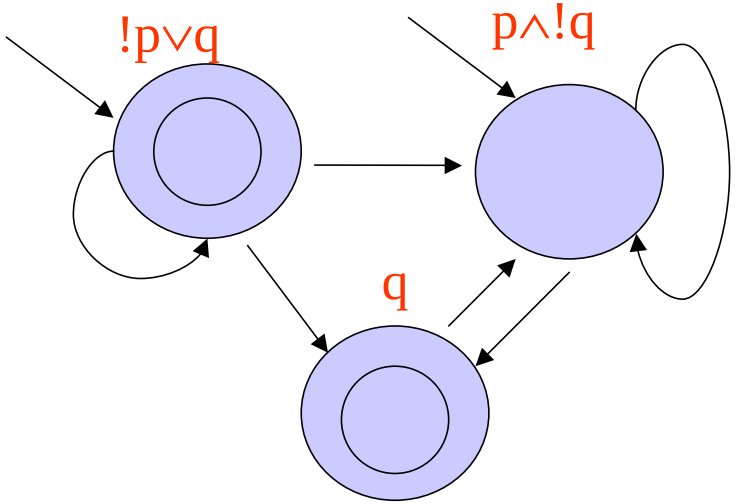


De LTL a Autómatas

$\Box (p \rightarrow \bigcirc q)$



$\Box (p \rightarrow \Diamond q)$



Autómatas de Büchi vacíos

$\mathcal{L}(\mathcal{A} \cap \overline{\mathcal{B}}) = \emptyset$??

Para comprobar que $\mathcal{A} = \langle \Sigma, S, \triangle, I, F \rangle$ es vacío razonamos como sigue:

- si $\rho \in \mathcal{L}(A)$, entonces contiene algunos elementos de F infinitas veces.
- Entonces $\exists \rho'$ tal que $\rho = \sigma \rho'$ cuyos estados aparecen infinitas veces.
- Es decir, cada estado de ρ' es alcanzable a partir de cualquier estado de ρ'.
- Por lo tanto, ρ' es una componente fuertemente conectada de $\mathcal{A}$.
- Además ρ' es alcanzable desde el estado inicial y contiene algún estado de F.

Autómatas de Büchi vacíos

$\mathcal{L}(\mathcal{A} \cap \overline{\mathcal{B}}) = \emptyset$??

Por lo tanto

$\mathcal{L}(A) \neq \emptyset \Leftrightarrow$

el autómata $\mathcal{A}$ tiene una componente

(1) fuertemente conectada,

(2) alcanzable desde algún estado inicial,

(3) conteniendo algún estado de aceptación

Además, podemos encontrar un contraejemplo del tipo $\sigma_1\sigma_2^\omega$, σ_1 y σ_2 secuencias de estados finitas.

Componentes fuertemente conectadas

```
procedure emptiness;
  forall s in I do
    dfs1(s);
  terminate(false);
end es_vacio;

procedure dfs1(s);
  var s'

  hash(s);
  forall successors s' of s do
    if s' not in the hash table
         then dfs1(s');
  if accept(s) then dfs2(s);
end dfs1;
```

```
procedure dfs2(s);
  var s'

  flag(s);
  forall successors s' of s do
    if s' is on dfs1 stack
       then terminate(true)
    else if s' no flagged
       then dfs2(s');
    end if;
end dfs2;
```

Aplicación al Model Checking

- Construimos $\mathcal{A}$ que representa el sistema.
- Construimos $\overline{\mathcal{B}}$ que representa el complemento de la especificación de la propiedad.

Habitualmente la propiedad $\neg\varphi$ se especifica en lógica temporal LTL y se aplica algún algoritmo que transforma fórmulas LTL a autómatas.

- Construimos $\mathcal{C} = \mathcal{A} \cap \overline{\mathcal{B}}$.
- Aplicamos algún algoritmo (por ejemplo,"emptiness") para encontrar alguna componente fuertemente conectada en el autómata, alcanzable desde algún estado inicial y con un estado de aceptación.
- Si no existe tal componente entonces $\mathcal{A}$ satisface la especificación $\mathcal{B}$, o φ.
- En otro caso, construimos un contraejemplo de la forma $\sigma_1\sigma_2^\omega$.

Model Checking on-the-fly

- Normalmente, no hace falta construir $\mathcal{C} = \mathcal{A} \cap \overline{\mathcal{B}}$ completamente.

La técnica on-the-fly consiste en

Construir sólo $\overline{\mathcal{B}}$

Dado $s = \langle r, q \rangle$, calculamos los sucesores de s uno por uno

Los sucesores de q, $q_1, \ldots, q_n$, ya han sido computados.

Dado r' un sucesor de r, $s_i = \langle r', q_i \rangle$ existe en la intersección sii

$$L(r') \wedge L(q_i) \neq false$$

Model Checking on-the-fly

Podemos reducir el espacio de búsqueda de dos formas:

1. Si $L(r') \wedge L(q_i) = false$
2. Si se detecta un ciclo antes de que algoritmo haga backtracking sobre s, la búsqueda termina y no se construyen los demás sucesores de s.

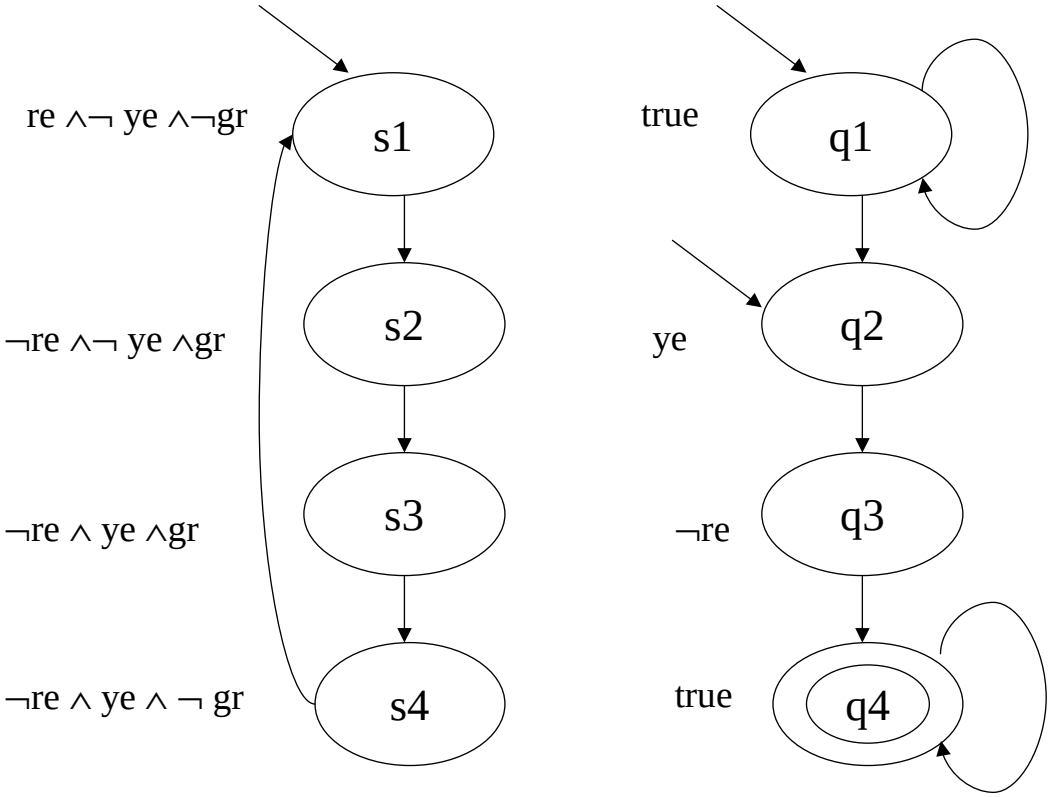
En ambos casos la reducción del número de estados está dirigida por el autómata de la especificación $\overline{\mathcal{B}}$.

Verificación automática: Un Ejemplo

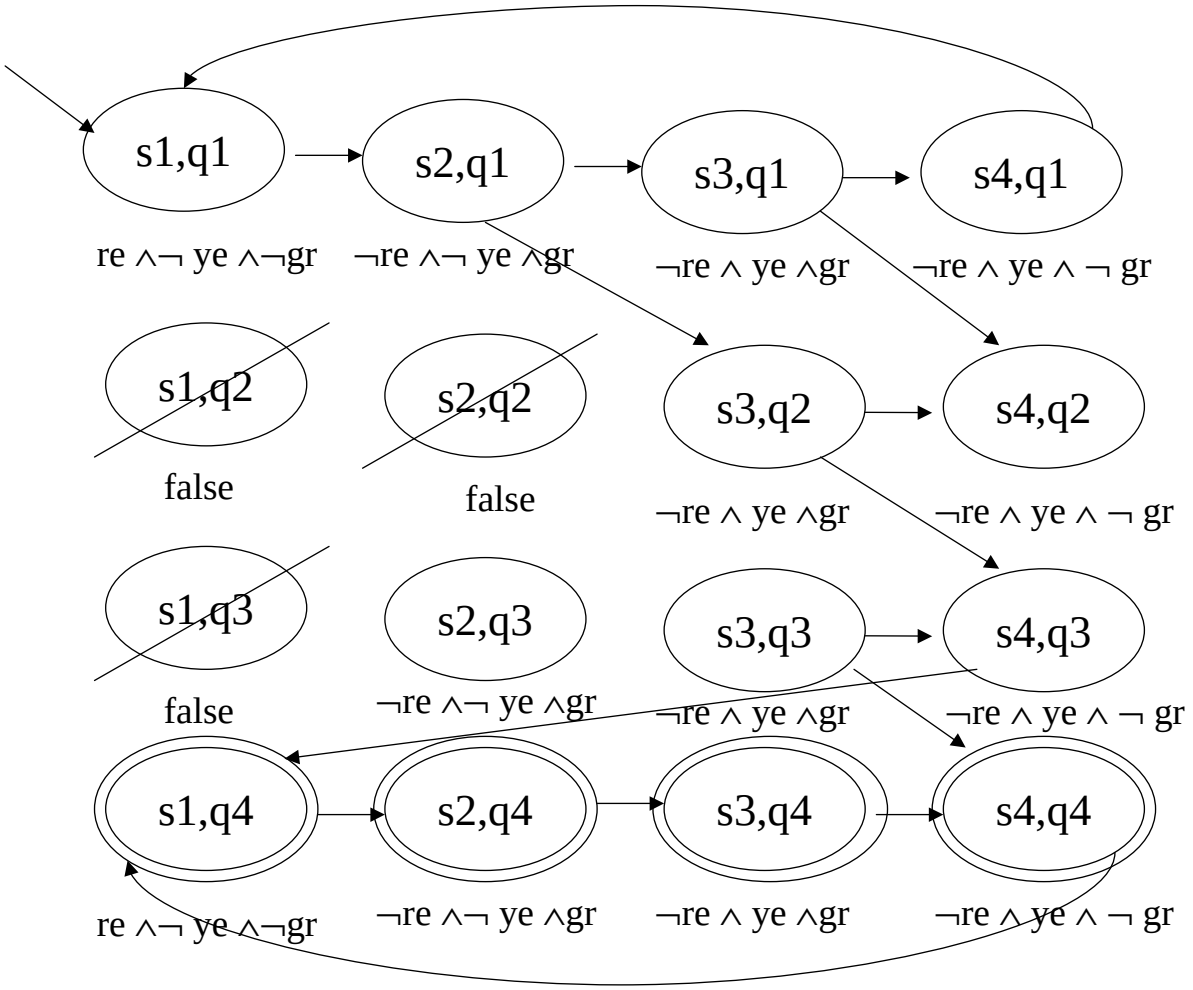
Modelo : un semáforo

Propiedad:“El semáforo pasa siempre de amarillo a rojo”

$\neg\Box(\texttt{ye} \rightarrow \bigcirc\texttt{re}) = \Diamond(\texttt{ye} \wedge \bigcirc\neg\texttt{re})$



Verificación automática: Un Ejemplo



Verificación automática: Un Ejemplo

Contraejemplo

$$\sigma_1 = \langle s_1, q_1 \rangle, \langle s_2, q_1 \rangle, \langle s_3, q_2 \rangle, \langle s_4, q_3 \rangle, \langle s_1, q_4 \rangle$$

$$\sigma_2 = \langle s_2, q_4 \rangle, \langle s_3, q_4 \rangle, \langle s_4, q_4 \rangle, \langle s_1, q_4 \rangle$$

El error se produce en $\langle s_3, q_2 \rangle \rightarrow \langle s_4, q_3 \rangle$ y no es verdadero. Debemos especificar la propiedad como

$$\Box(\texttt{ye} \rightarrow \texttt{ye} \cup \texttt{re})$$

Bibliografía

- D. A. Peled, Software Reliability Methods, 2001, Springer
- E. M. Clarke and O. Grumberg and D. A. Peled, Model Checking, 2000, The MIT Press
- B. Bérard et. al, Systems and Software Verification. Model Checking Techniques and Tools, 1999, Springer