Ramon Ruiz Dolz
Salvador Marti Roman

Computabilidad y complejidad: 3CO21

```
(*Autómatas Celulares (1)*)


ReglaFormato[regla_] := Module[{rule, listbin, VNnum, i},
  rule = regla;
  listbin = Reverse[IntegerDigits[rule, 2, 8]];
  (*Número decimal expresado en binario de longitud 8*)
  VNnum = {{0, 0, 0}, {0, 0, 1}, {0, 1, 0},
    {0, 1, 1}, {1, 0, 0}, {1, 0, 1}, {1, 1, 0}, {1, 1, 1}};
  (*Número de Von Neumann*)
  For[i = 1, i ≤ Length[listbin], i++,
   (*Añade cada numero del decimal
    en la posicion i al número de Von Neumann i*)
   AppendTo[VNnum[[i]], listbin[[i]]];
  ];
  Return[VNnum];
 ]
```

```
ReglaFormato[54]
```

```
{{0, 0, 0, 0}, {0, 0, 1, 1}, {0, 1, 0, 1}, {0, 1, 1, 0},
 {1, 0, 0, 1}, {1, 0, 1, 1}, {1, 1, 0, 0}, {1, 1, 1, 0}}
```

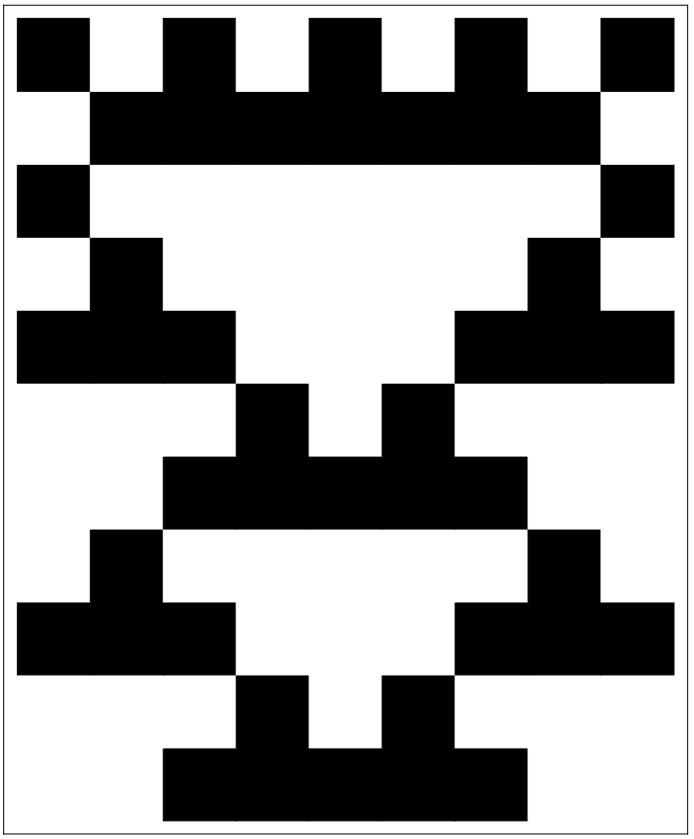
```
Transicion[regla_, estado_] := Module[{ini, fin, i, val, reg, len},
  ini = estado;
  reg = ReglaFormato[regla];
  (*Obtenemos la regla expresada adecuadamente*)
  fin = {};
  len = Length[ini];
  For[i = 0, i < len, i++,
   (*Hace matching para saber a que configuración de celula transitar*)
   val = Cases[reg, {ini[[len]], ini[[1]], ini[[2]], _}];
   val = Flatten[val];
   (*Añade a la solución el nuevo valor de la celula*)
   AppendTo[fin, val[[4]]];
   (*Rota a la izquierda para hacer el matching con la siguiente celula*)
   ini = RotateLeft[ini];
  ];
  Return [fin];
 ]
```

```
Transicion[54, {1, 0, 1, 0, 1, 0, 1, 0, 1}]
```

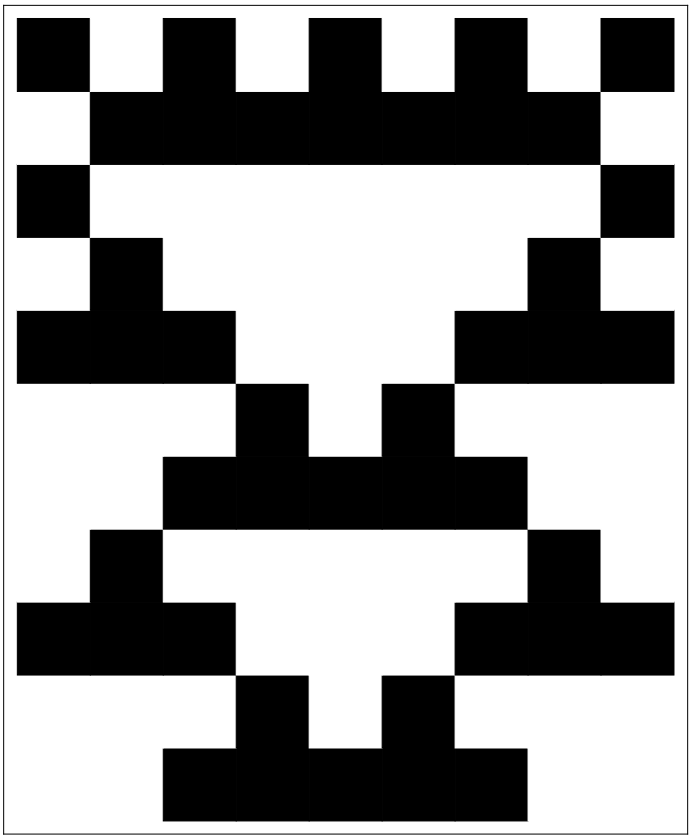
```
{0, 1, 1, 1, 1, 1, 1, 1, 0}
```

```
ACunidim[estado_, regla_, numtrans_] := Module[{t, final, reg, i, aux},
  t = numtrans; (*Numero de configuraciones a calcular*)
  aux = estado; (*Configuracion inicial*)
  final = {estado}; (*Representacion de las transiciones*)
  reg = regla;
  For[i = 0, i < t, i++,
   (*Calcula una configuracion por cada
    numero de configuraciones a calcular y las añade a final*)
   aux = Transicion[reg, aux];
   AppendTo[final, aux];
  ];
  (*Visualizacion de las iteraciones*)
  ArrayPlot[final]
 ]
```

```
ACunidim[{1, 0, 1, 0, 1, 0, 1, 0, 1}, 54, 10]
```



```
ArrayPlot[CellularAutomaton[54, {1, 0, 1, 0, 1, 0, 1, 0, 1}, 10]]
```



```
(*Autómatas Celulares (2)*)
```

```
automata = {{q, p, r}, {a, b},
   {{q, a, p}, {q, b, r}, {p, a, r}, {p, b, q}, {r, a, r}, {r, b, r}}, q, {r}}
```

```
{{q, p, r}, {a, b},
 {{q, a, p}, {q, b, r}, {p, a, r}, {p, b, q}, {r, a, r}, {r, b, r}}, q, {r}}
```

```
(*Ejercicio 1*)
Act1[automata_] := Module[{i, k, est, sim, tran, regla, fin, AC},
   est = automata[[1]]; (*Estados del automata*)
   sim = automata[[2]]; (*Símbolos del automata*)
   tran = automata[[3]]; (*Transiciones del automata*)
   fin = automata[[5]]; (*Estados finales del automata*)
   regla = {};
   AC = {};
   For[i = 1, i ≤ Length[est], i++,
    For[k = 1, k ≤ Length[est], k++,
      (*Combinaciones de tripletas
       de estados del AFD se añaden a la regla del AC*)
      AppendTo[regla, {est[[i]], est[[k]], est[[k]]}];
     ];
   ];
   For[i = 1, i ≤ Length[sim], i++,
    For[k = 1, k ≤ Length[sim], k++,
      (*Combinaciones de tripletas
       de símbolos del AFD se añaden a la regla del AC*)
      AppendTo[regla, {sim[[i]], sim[[k]], sim[[k]]}];
     ];
   ];
   For[i = 1, i ≤ Length[tran], i++,
    (*Se añaden las transiciones del AFD a las reglas del AC*)
    AppendTo[regla, tran[[i]]];
   ];
   (*Se añade S, la union de estados y simbolos al AC*)
   AppendTo[est, sim];
   AppendTo[AC, Flatten[Union[est]]];
   (*Se añade f, la regla al AC*)
   AppendTo[AC, Union[regla]];
   (*Se añade S+, los finales al AC*)
   AppendTo[AC, Union[fin]];
   Return[AC];
  ];

Act1[automata]

{{p, q, r, a, b}, {{a, a, a}, {a, b, b}, {b, a, a}, {b, b, b}, {p, a, r}, {p, b, q},
  {p, p, p}, {p, q, q}, {p, r, r}, {q, a, p}, {q, b, r}, {q, p, p}, {q, q, q},
  {q, r, r}, {r, a, r}, {r, b, r}, {r, p, p}, {r, q, q}, {r, r, r}}, {r}}
```

```
autocel = {{p, q, r, a, b}, {{a, a, a}, {a, b, b}, {b, a, a}, {b, b, b}, {p, a, r},
    {p, b, q}, {p, p, p}, {p, q, q}, {p, r, r}, {q, a, p}, {q, b, r}, {q, p, p},
    {q, q, q}, {q, r, r}, {r, a, r}, {r, b, r}, {r, p, p}, {r, q, q}, {r, r, r}}, {r}}
```

```
{{p, q, r, a, b}, {{a, a, a}, {a, b, b}, {b, a, a}, {b, b, b}, {p, a, r}, {p, b, q},
  {p, p, p}, {p, q, q}, {p, r, r}, {q, a, p}, {q, b, r}, {q, p, p}, {q, q, q},
  {q, r, r}, {r, a, r}, {r, b, r}, {r, p, p}, {r, q, q}, {r, r, r}}, {r}}
```

```
(*Ejercicio 2*)
Act2[automata_, palabra_, q_] := Module[{s, f, S, i, cel, len},
   s = automata[[1]];
   f = automata[[2]];
   S = automata[[3]];
   len = Length[palabra];
   cel = Prepend[palabra, q];
   For[i = 1, i ≤ len, i++,
    (*Calcula una nueva configuracion por cada caracter de la palabra*)
    cel = Transicion2[f, cel]
   ];
   (*Comprueba que el último elemento de la
     configuracion final pertenezca a S+, es decir, que sea final*)
   Return[MemberQ[S, cel[[len + 1]]]];
  ];

Transicion2[regla_, estado_] := Module[{ini, fin, i, val, reg, len},
  ini = estado;
  reg = regla;
  fin = {};
  len = Length[ini];
  For[i = 2, i ≤  len, i++,
   (*Hace matching para saber a que configuración de celula transitar*)
   val = Cases[reg, {ini[[i - 1]], ini[[i]], _}];
   val = Flatten[val];
   (*Añade el nuevo valor de la celula*)
   AppendTo[fin, val[[3]]];
  ];
  fin = Prepend[fin, ini[[1]]];
  Return [fin];
 ]

Act2[autocel, {a, b, a, a, b}, q]
```

```
True
```

```
(*Ejercicio 3*)
Act3[automata_] := Module[{aux, i, k, j, est, sim, tran, regla, fin, AC},
  est = automata[[1]];
  sim = automata[[2]];
  tran = automata[[3]];
```

```
 fin = automata[[5]];
 aux = {};
 regla = {};
 AC = {};
 For[i = 1, i ≤ Length[est], i++,
  For[k = 1, k ≤ Length[est], k++,
    For[j = 1, j ≤ Length[est], j++,
      (*Combinaciones de quatripletas
       de estados del AFD se añaden a la regla del AC*)
      AppendTo[regla, {est[[i]], est[[k]], est[[j]], est[[k]]}];
     ];
   ];
 ];
 For[i = 1, i ≤ Length[sim], i++,
  For[k = 1, k ≤ Length[sim], k++,
    For[j = 1, j ≤ Length[sim], j++,
      (*Combinaciones de quatripletas
       de símbolos del AFD se añaden a la regla del AC*)
      AppendTo[regla, {sim[[i]], sim[[k]], sim[[j]], sim[[k]]}];
     ];
   ];
 ];
 For[i = 1, i ≤ Length[tran], i++,
  For[k = 1, k ≤ Length[est], k++,
   (*Se añaden las transiciones del AFD
    combinadas con todos los estados a las reglas del AC*)
   aux = Insert[tran[[i]], est[[k]], 3];
   AppendTo[regla, aux];
  ];
  For[k = 1, k ≤ Length[sim], k++,
   (*Se añaden las transiciones del AFD
    combinadas con todos los símbolos a las reglas del AC*)
   aux = Insert[tran[[i]], sim[[k]], 3];
   AppendTo[regla, aux];
  ];
 ];
 (*Añadimos S al AC*)
 AppendTo[est, sim];
 AppendTo[AC, Flatten[Union[est]]];
 (*Añadimos f al AC*)
 AppendTo[AC, Union[regla]];
 (*Añadimos S+ al AC*)
 AppendTo[AC, Union[fin]];
 Return[AC];
]
```

Act3[automata]

```
{{p, q, r, a, b},
 {{a, a, a, a}, {a, a, b, a}, {a, b, a, b}, {a, b, b, b}, {b, a, a, a}, {b, a, b, a},
  {b, b, a, b}, {b, b, b, b}, {p, a, a, r}, {p, a, b, r}, {p, a, p, r}, {p, a, q, r},
  {p, a, r, r}, {p, b, a, q}, {p, b, b, q}, {p, b, p, q}, {p, b, q, q}, {p, b, r, q},
  {p, p, p, p}, {p, p, q, p}, {p, p, r, p}, {p, q, p, q}, {p, q, q, q}, {p, q, r, q},
  {p, r, p, r}, {p, r, q, r}, {p, r, r, r}, {q, a, a, p}, {q, a, b, p}, {q, a, p, p},
  {q, a, q, p}, {q, a, r, p}, {q, b, a, r}, {q, b, b, r}, {q, b, p, r}, {q, b, q, r},
  {q, b, r, r}, {q, p, p, p}, {q, p, q, p}, {q, p, r, p}, {q, q, p, q}, {q, q, q, q},
  {q, q, r, q}, {q, r, p, r}, {q, r, q, r}, {q, r, r, r}, {r, a, a, r}, {r, a, b, r},
  {r, a, p, r}, {r, a, q, r}, {r, a, r, r}, {r, b, a, r}, {r, b, b, r}, {r, b, p, r},
  {r, b, q, r}, {r, b, r, r}, {r, p, p, p}, {r, p, q, p}, {r, p, r, p}, {r, q, p, q},
  {r, q, q, q}, {r, q, r, q}, {r, r, p, r}, {r, r, q, r}, {r, r, r, r}}, {r}}
```

```
(*Ejercicio 4*)
Act4[automata_, palabra_, q_] :=
 Module[{aux, aux2, s, f, S, i, k, cel, len, final, frame},
  s = automata[[1]]; (*S*)
  f = automata[[2]]; (*f*)
  S = automata[[3]]; (*S+*)
  len = Length[palabra];
  cel = Prepend[palabra, q];
  aux = {};
  aux2 = {};
  frame = {};
  final = {cel};
  (*Generamos un frame vacio para la animación de len x len*)
  For[k = 0, k ≤ len, k++,
   AppendTo[aux2, 0];
  ];
  For[i = 0, i ≤ len, i++,
   AppendTo[aux, aux2];
  ];
  AppendTo[frame, ArrayPlot[aux,
    ColorRules → {a → Red, b → Blue, q → Yellow, p → Purple, r → Cyan}]];
  (*Por cada trasicion vamos añadiendo un nuevo frame a la animacion*)
  For[i = 0, i ≤ len, i++,
   aux = Rest[aux];
   AppendTo[aux, cel];
   (*Transicion a una nueva configuracion*)
   cel = Transicion2[f, cel];
   AppendTo[final, cel];
   AppendTo[frame, ArrayPlot[aux,
     ColorRules → {a → Red, b → Blue, q → Yellow, p → Purple, r → Cyan}]];
  ];
  Return[ListAnimate[frame]]
 ]
```

```
Act4[autocel, {a, b, a, a, b}, q]
```