Escuela Técnica Superior de Ingeniería Informática
etsinf
DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Computabilidad y Complejidad

Práctica 2: Autómatas Celulares
(II – Reconocedores de Lenguajes)

Autómatas Celulares
Reconocedores/aceptores de lenguajes

Índice:

Bibliografía Básica

- Cellular Automata. A Discrete Universe. A. Ilachinski. World Scientific. 2001.
- Cellular Automata. A Parallel Model. M. Delorme, J. Mazoyer (Eds.) Kluwer. 1999.
- Game of Life Cellular Automata. A. Adamatzky (Ed.) Springer. 2010.
- Handbook of Natural Computing (Vol. 1). G. Rozenberg, T. Bäck, J.N. Kok (Eds.) Springer. 2012.

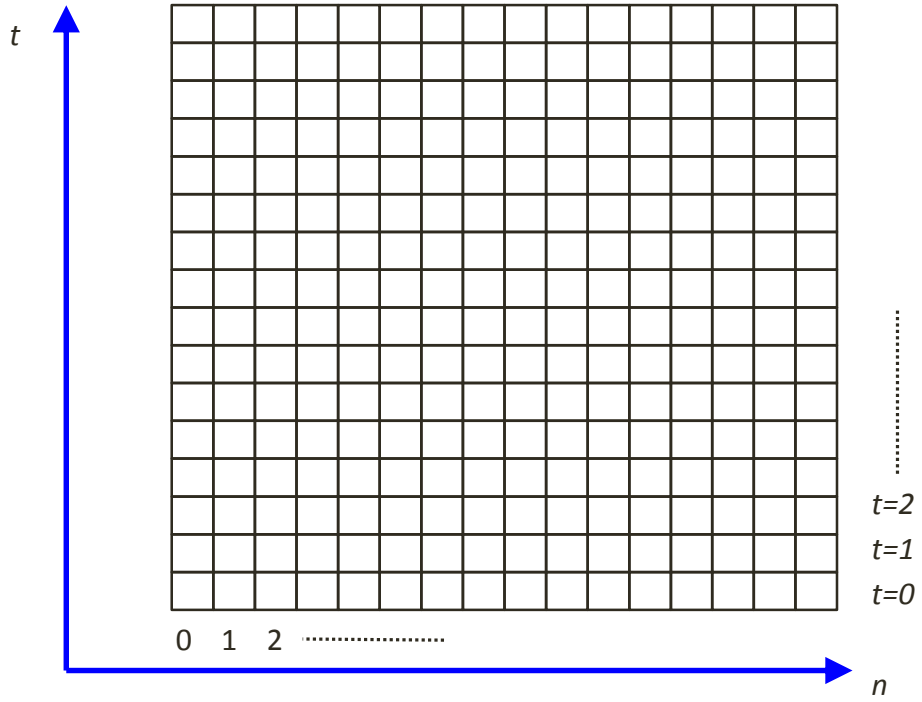
Autómatas Celulares como reconocedores de lenguajes

Algunas puntualizaciones previas

Sea A=(Z,S,N,f) un autómata celular unidimensional

Un estado s ∈ S diremos que es quiescente si cumple que f(s,s,…,s)=s

Utilizaremos diagramas espacio-temporales



Utilizaremos condición de frontera abierta.
Los lenguajes a reconocer/aceptar serán lenguajes formales caracterizados en la jerarquía de Chomsky

Autómatas Celulares como reconocedores/aceptores de lenguajes

Criterios en los componentes

AC unidireccional vs AC bidireccional

- Unidireccional: La información fluye en una sola dirección. El vecindario se define a partir de la propia célula y su célula a la izquierda (o a la derecha).
- Bidireccional: La información fluye en ambas direcciones. El vecindario se define a partir de la propia célula y sus células circundantes.

Modo de entrada

- Paralelo: la cadena de entrada se introduce en la configuración inicial (cada símbolo en una célula definiendo su estado)
- Secuencial: Se designa una célula de entrada predeterminada. Todas las células (excepto la de entrada) están en un estado quiescente y la cadena de entrada se va suministrando símbolo a símbolo finalizando con un símbolo especial (p.ej. $)

Modo de salida

Se establece una célula que almacene el estado de aceptación/rechazo de la cadena de entrada. En autómatas unidireccionales la célula debe de procesar toda la información de entrada. En el caso bidireccional la célula de salida se elige de forma arbitraria

Autómatas Celulares como reconocedores/aceptores de lenguajes

Reconocimiento vs aceptación

(a) En el caso de reconocimiento de lenguajes, en el autómata se definen dos conjuntos disjuntos de estados: S^+ (estados de aceptación) y S^- (estados de rechazo). El autómata reconoce el lenguaje L si, tomando como entrada una cadena w, la célula de salida entra en un estado de aceptación si $w \in L$, o de rechazo si $w \notin L$ en el momento de tiempo t_e. En todo momento anterior a t_e, la célula no está en ningún estado de S^+ ni de S^-.

(b) En el caso de aceptación de lenguajes, en el autómata se define únicamente el conjunto de estados de aceptación. El autómata acepta el lenguaje L si, tomando como entrada una cadena w, la célula de salida entra en un estado de aceptación sii $w \in L$.

Observación: Cuando el autómata celular tiene una complejidad temporal definida, entonces los conceptos de reconocimiento y de aceptación son equivalentes y no se hace distinción entre ellos. En lo sucesivo, tomaremos ambos conceptos como equivalentes.

Autómatas Celulares como reconocedores/aceptores de lenguajes

Tipos de autómatas celulares como aceptores de lenguajes en dimensión uno

Arrays iterativos

PCA: Entrada paralela Célula de salida: n Vecindario: {-1,0,1}	SCA: Entrada secuencial Célula de salida: n Vecindario: {-1,0,1}	Bidireccionales
POCA: Entrada paralela Célula de salida: n Vecindario: {-1,0}	SOCA: Entrada secuencial Célula de salida: n Vecindario: {-1,0}	Unidireccionales

Además, si el autómata trabaja en tiempo real, el número de configuraciones para aceptar ó rechazar una cadena de longitud n es exactamente n en los casos PCA, SCA y POCA y 2n en el caso SOCA. Para identificar estos caso introduciremos el prefijo R: RPCA, RSCA, RPOCA y RSOCA.

En el caso de complejidad lineal, el número de configuraciones para aceptar o rechazar una cadena de longitud n es una constante multiplicativa con respecto al caso de tiempo real. Para identificar este caso introduciremos el prefijo L: LPCA, LSCA, LPOCA y LSOCA.

Autómatas Celulares como reconocedores/aceptores de lenguajes

Cada tipo de autómata define la clase de lenguajes que acepta. Podemos establecer las siguientes relaciones entre esas clases de lenguajes.

$$\text{RPOCA} \neq \text{RSCA} \subset \text{RPCA=RSOCA=LPOCA} \subseteq \text{LPCA=LSCA=LSOCA} \subseteq \text{POCA=SOCA} \subseteq \text{PCA=SCA}$$

Además se cumplen las siguientes relaciones con clases de lenguajes de la Jerarquía de Chomsky

$$\text{REG} \subset \text{LIN} \subseteq \text{RPOCA} \not\subset \text{CF} \subset \text{POCA=SOCA}$$

$$\text{REG} \subseteq \text{RSCA} \not\subset \text{CF}$$

$$\text{PCA=SCA} \subseteq \text{CS}$$

Además si no se acota el número de células que utiliza el autómata tenemos las clases uPOCA, uSOCA, uPCA y uSCA y se cumplen las siguientes relaciones

$$\text{uPOCA} = \text{uSOCA} = \text{uPCA} = \text{uSCA} = \text{RE}$$

Autómatas Celulares y Lenguajes Regulares

Para todo lenguaje regular L existe un AC unidireccional unidimensional que lo reconoce en tiempo real.

Demostración

Sea L = L(A) donde A es un AFD definido como $A = (Q, \Sigma, \delta, q_0, F)$

Podemos definir un autómata celular unidimensional unidireccional y con entrada paralela con las siguientes características:

$S = Q \cup \Sigma$ con $S^+ = F$

Definiremos el vecindario {-1,0}

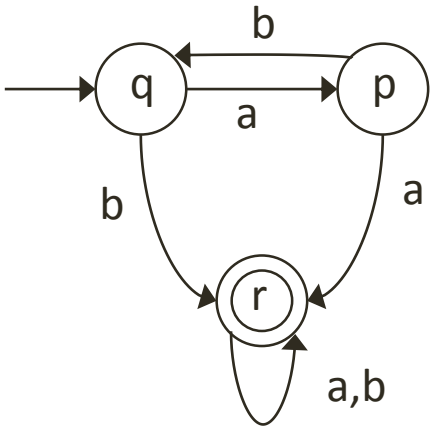
Si $\delta(q, a) = p$ entonces f(q,a)=p

Además $(\forall a, b \in \Sigma)$ f(a,b)=b y $(\forall q, p \in Q)$ f(q,p)=p

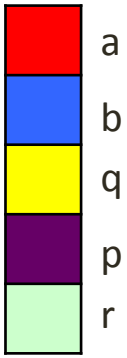
Consideramos condición de frontera abierta con el estado quiescente q_0

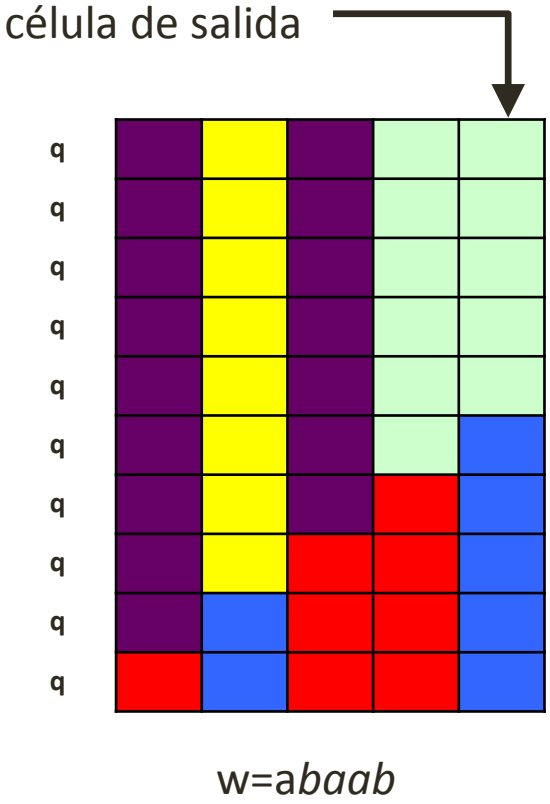
Autómatas Celulares y Lenguajes Regulares

Un ejemplo



f	a	b	q	p	r
a	a	b			
b	a	b			
q	p	r	q	p	r
p	r	q	q	p	r
r	r	r	q	p	r





w=a*baaab*

Implementación en *Mathematica*

1. Para las actividades que se plantean en esta práctica, se deben implementar AC con funciones de transición definidas de la siguiente forma:

   - En el caso unidireccional (vecindario {-1,0}) implementaremos la función como una lista de listas de tres elementos:
     {…, {vecinoizquierda,celula,nuevoestado},…}
   - En el caso bidireccional (vecindario {-1,0,1}) implementaremos la función como una lista de listas de cuatro elementos:
     {…, {vecinoizquierda,celula,vecinoderecha, nuevoestado}, …}

2. Dado que hay que definir estados de aceptación, el AC $A=(Z,S,N,f,S^+)$ se implementará como la tupla (S,f,S^+) donde S^+ será una lista de estados incluida en S. Obsérvese que Z no se representará explícitamente. Además, la definición de f implicará la definición de N, por lo que tampoco hace falta definir N explícitamente.

3. Para la implementación de AFDs nos remitiremos a las estructuras de listas que se vieron en la asignatura de TAL.

Actividades propuestas

1. Implemente un módulo en *Mathematica* que, tomando como entrada un AFD proporcione como salida un AC unidireccional unidimensional que acepte el mismo lenguaje que el AFD mediante entrada paralela.

2. Implemente un módulo *Mathematica* que, tomando como entrada una cadena arbitraria, un AC que reconoce un lenguaje mediante entrada paralela en tiempo real y un estado de frontera $q \in S$, proporcione como salida True si el AC acepta la cadena y False en caso contrario.

3. Proporcione una construcción formal para obtener a partir de un AFD, un AC equivalente unidimensional bidireccional (vecindario {-1,0,1}) con entrada paralela. Implemente un módulo en *Mathematica* que lleve a cabo la construcción propuesta.

4. Implemente un módulo *Mathematica* que muestre la secuencia de computación que se lleva a cabo en el módulo propuesto en la actividad (2) mediante un diagrama espacio-temporal. (Nota: Utilice las funciones ArrayPlot y ListAnimate explicados en la práctica de Fundamentos Básicos)