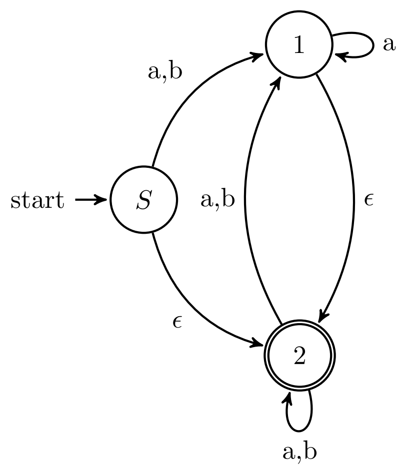
Autómatas y lenguajes formales. Tarea 2

Fabián Romero Jiménez

October 30, 2013

Problema 1 Minimiza el autómata de tu respuesta a los ejercicios 3 y 5 de la tarea 1.

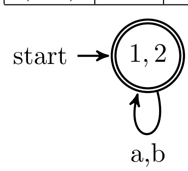


Transfórmalo en un autómata determinista usando los métodos vistos en clase. Minimiza el resultado.

Respuesta

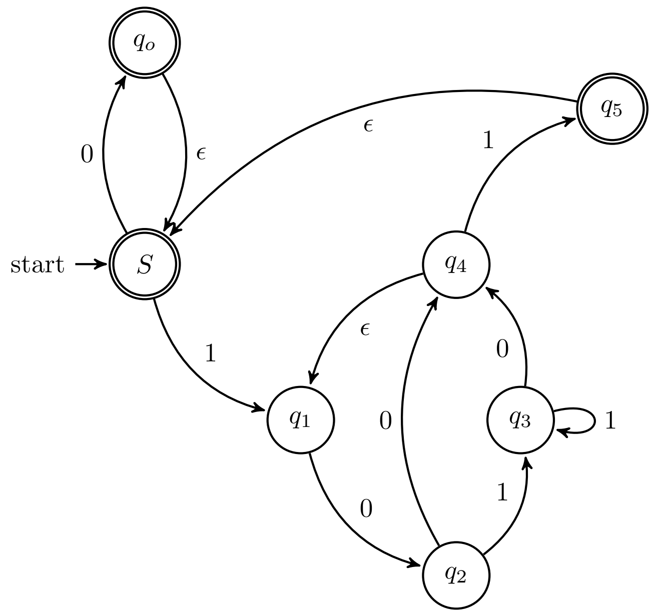
	a	b	$\epsilon^\star$
$> S$	1	1	S,2
1	1	-	1,2
(2)	2	2	2

	$a\epsilon^\star$	$b\epsilon^\star$
$> S$	1,2	1,2
$(1,2)$	1,2	1,2



Problema 5 Construye un autómata que acepte el lenguaje generado por la expresión $(0 + 1(01^\star 0)^\star 1)^\star$. Aplica el algoritmo de minimalización a tu autómata.

Entonces, tenemos los simbolos 010101 por lo que empezamos poniendo los 6 estados $q_0, q_1, .., q_5$ que corresponen a los símbolos en la expresión y el estado inicial S, además por cada simbolo + hacemos una bifurcación y por cada $\star$ una transición ϵ a donde empieza, así tenemos inicialmente el NFA

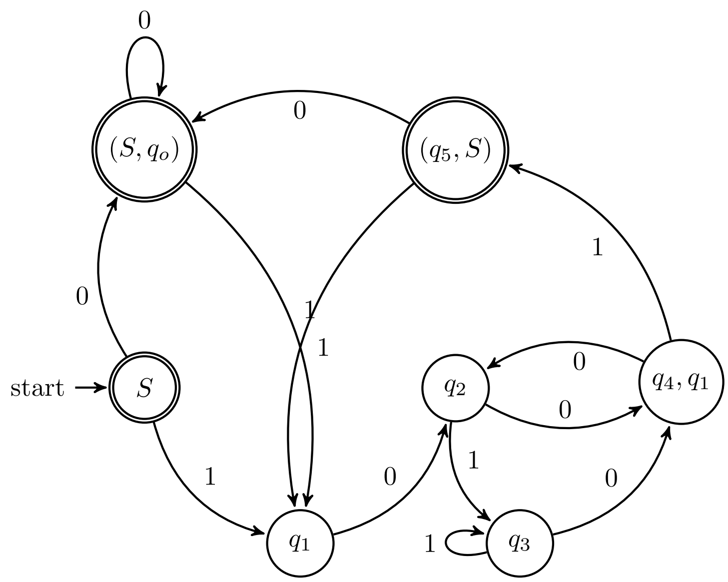


Y lo transformamos en un DFA

	0	1	$\epsilon^\star$
$(> S)$	q_0	q_1	S
(q_0)	–	-	S, q_0
q_1	q_2	-	q_1
q_2	q_4	q_3	q_2
q_3	q_4	q_3	q_3
q_4	–	q_5	q_4, q_1
(q_5)	–	-	q_5, S

	$0\epsilon^\star$	$1\epsilon^\star$
$(> S)$	S, q_0	q_1
(S, q_0)	S, q_0	q_1
q_1	q_2	–
q_2	q_4, q_1	q_3
q_4, q_1	q_2	q_5, S
q_3	q_4, q_1	q_3
(q_5, S)	q_0, S	q_1

Así tenemos el DFA



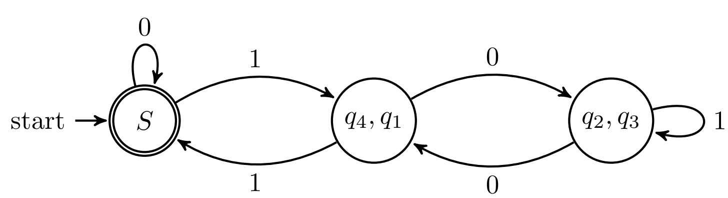
Minimizando tenemos:

$$\{q_1, q_2, q_3, (q_4, q_1)\}, \{S, (q_0, s), (q_5, S)\}$$

Separando (q_4, q_1) pues con entrada 1 va a un estado final

$$\{q_2, q_3\}, \{(q_4, q_1), q1\}, \{S, (q_0, s), (q_5, S)\}$$

y finalmente tenemos el DFA minimizado:



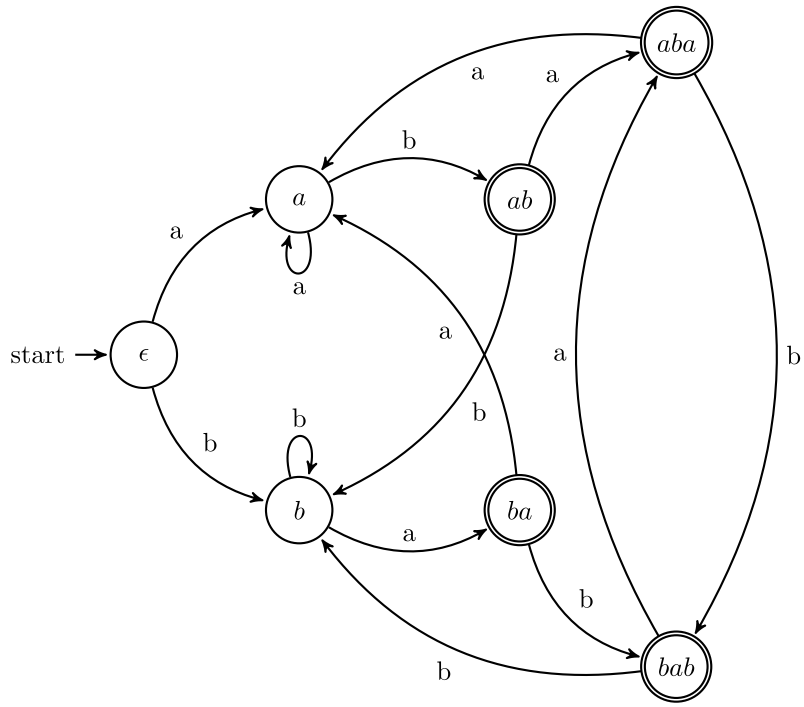
Problema 2 . Considera el siguiente sistema de ecuaciones

$X_0 = a \cdot (X_1 \wedge X_2) + b \cdot X_o + 0$
$X_1 = b \cdot (\bar{X}_0 \vee X_2) + a \cdot X_o + \epsilon$
$X_2 = a \cdot (X_1 \vee \bar{X}_2) + b \cdot (\bar{X}_1 \wedge X_2) + \epsilon$

Lo que nos indica que hay 3 estados, s que es testado inicial y (1) y (2) que son de aceptación, debido a que sus ecuaciones X_1 y X_2 tienen constante ϵ, como la ecuación de s tiene como conastante 0, no es un estado de aceptación, y la tabla de transiciones es la siguiente:

Estado	a	b
$> s$	$1 \wedge 2$	s
(1)	s	$2 \vee \neg s$
(2)	$1 \vee \neg 2$	$\neg 1 \wedge 2$

Problema 3 . Construye el autómata de diccionario para el conjunto $X = \{ab, ba, aba, bab\}$ con el alfabeto $\Sigma = \{a, b, c\}$.



Problema 4 . Considera la siguiente descripción de un lenguaje de programación simple:

- Localidades de memoria: $X, Y, Z, X_1, ...$;
- Constantes: $0, 1, -1, ...$;
- Expresiones aritméticas: (a) localidades; (b) constantes; (c) si a y b son expresiones aritméticas, también lo son $(a + b)$, $(a \times b)$ y $(a - b)$;
- Constantes boolenas: V y F;
- Comparaciones: $(X = a)$ y $(X < a)$, donde X es una localidad y a una expresión aritmética;
- Expresiones booleanas: (a) constantes booleanas; (b) comparaciones; (c) si b y v son expresiones booleanas, también lo son $\neg b$, $(b \vee v)$ y $(b \wedge v)$;
- Asignaciones: $X := a$, donde X es una localidad y a una expresión aritmética;

- El programa *skip*;
- Programas: (a) skip; (b) asignaciones; (c) si P y Q son programas y b es una expresión booleana, los siguientes también son programas: (P; Q), (if b then P else Q) y (while b do P).

CFG $P \to skip|N|$ if B then P else $P|$ while B do P // Programa
$N \to A|E|N;N$ // Predicado
$B \to v|f|E < E|E = E|\neg B|B \vee B|B \wedge B$ // Expresión Booleana
$A \to L := E$ // Asignación
$E \to C| - C|E + E|E - E|E \times E$ //Expresión
$L \to x|y|z|LC$ // Localidad
$C \to 0|1|2|..|9|CC$ // Constante

Problema 5 Da gramáticas en forma normal de Chomsky y de Greibach del mismo lenguaje

CNF $IF \to if$
$THEN \to then$
$ELSE \to else$
$W \to while$
$DO \to do$
$A_{:=} \to :=$
$N_{;} \to ;$
$E_{+} \to +$
$E_{\times} \to \times$
$E_{-} \to -$
$B_{<} \to <$
$B_{=} \to =$
$B_{\neg} \to \neg$
$B_{\vee} \to \vee$
$B_{\wedge} \to \wedge$
$L \to x|y|z|L\ C$
$C \to 0|1|2|3|4|5|6|7|8|9|CC$

$P_0 \to skip|L\ N_{asign_1}|E\ E_{+1}|E\ E_{\times 1}|E\ E_{-1}|E_{-}\ C|NN_{sep_1}|IF\ P_{if_1}|W\ P_{w_1}$
$P \to skip|L\ N_{asign_1}|E\ E_{+1}|E\ E_{\times 1}|E\ E_{-1}|E_{-}\ C|NN_{sep_1}|IF\ P_{if_1}|W\ P_{w_1}$
$P_{if_1} \to B\ P_{if_2}$
$P_{if_2} \to THEN\ P_{if_3}$
$P_{if_3} \to P\ P_{if_4}$
$P_{if_4} \to ELSE\ P$
$P_{w_1} \to B\ P_{w_2}$
$P_{w_2} \to DO\ P$
$N_{sep_1} \to N_{;}\ N$
$N_{asign_1} \to A_{:=}\ N$
$B \to v|f|E\ B_{<1}|E\ B_{=1}|B\ B_{\wedge 1}|B\ B_{\vee 1}|B_{\neg}\ B$
$B_{=1} \to B_{=}\ E$

$B_{<1} \to B_< \ E$
$E \to 0|1|2|3|4|5|6|7|8|9|E\ E|E_- E|E\ E_{+1}|E\ E_{-1}|E\ E_{\times 1}$
$E_{+1} \to E_+ \ E$
$E_{-1} \to E_- \ E$
$E_{\times 1} \to E_\times \ E$

GNF $P \to skip|N|$ if B then P else $P|$ while B do P // Programa
$N \to A|E|N;N$ // Predicado
$B \to v|f|E < E|E = E|\neg B|B \vee B|B \wedge B$ // Expresión Booleana
$A \to L := E$ // Asignación
$E \to C| - C|E + E|E - E|E \times E$ //Expresión
$L \to x|y|z|LC$ // Localidad
$C \to 0|1|2|..|9|CC$ // Constante

Problema 6 Describe un NPDA que acepte este lenguaje de programación.

En general, podemos convertir cualquier CFG dado en GNF a un NPDA de la siguiente forma:

sea $G = (\Sigma_g, \Gamma_g, S_g, \to_g)$ la Gramática en GNF, costruimos una NPDA $M = (Q_m, \Sigma_m, \Gamma_m, \gamma_m, s_m)$ con aceptación con pila vacía siguiendo las reglas:
$Q_m = q$ Un único estado
$\Sigma_m = \Gamma_g$ Los símbolos de entrada de M, son los simbolos terminales de G $\Gamma_m = \Sigma_g$ Los símbolos de pila de M, son todos los simbolos de de G
$s_m = S_g$ El simbolo inicial de M es el simbolo inicial de G.

Las función de transición será la siguiente:
para toda $\alpha \in \Sigma_g$ es decir, para todos los simbolos terminales, hay una transición:
$\delta(q, a, a) = (q, \epsilon)$ como esta en GNF, las otras producciones del tipo

$$\beta \to \alpha\beta_1\beta_2...\beta_n$$

se agrega a la función de transición: $\delta(q, \alpha, \beta) = (q, \beta_1\beta_2...\beta_n)$ en caso de que sea n sea 0 es decir una producción de la forma $\beta \to \alpha$ se agrega $\delta(q, \alpha, \beta) = (q, \epsilon)$

y este NPDA acepta el mismo lenguage que la CFG en GNF dada.

Problema 7 El teorema de Chomsky-Schützenberger nos dice que hay existen $n \in \mathbb{N}, R \in Reg$ tales que existe un homomorfismo entre $D_n^\star \cap R$ y el lenguaje de programación anterior. Da un valor de n y justifica tu respuesta.

Problema 8 Da CFG para los lenguages:

a) $\{a^n b^{2n} c^k | 1 \leq k, n\}$
$S \to Ac|Sc$
$A \to aAbb|abb$

b) $\{a^k b^m c^n | 1 \leq k, m, n, n \leq 2k\}$

$S \rightarrow aaAc|aaSc$
$A \rightarrow aA|Ab|b$

c) $\{a,b\}^\star - \{palindromas\}$

$S \rightarrow aSa|bSb|aSb|bSa|AB|BA$
$A \rightarrow a|aA$
$B \rightarrow b|bB$

Problema 9 Demuestra que los siguientes conjuntos no son CFL:

a) $\{a^n b^m c^k d^n | 2n = 3m \wedge 5k = 7m\}$
si $2n = 3m$ y $5k = 7m$ tenemos que $2|m$ y $5|m$ asi que $10|m$ por lo que diremos que $m = 10m'$ como $2n = 3m$ tenemos que $2n = 30m'$ y entonces $n = 15m'$, análogamente por $5k = 7m$ tenemos que $5k = 70m'$ y $k = 14m'$ así el lenguage lo podemos expresar como $\{a^{15m'} b^{10m'} c^{14m'} d^{15m'} | m \in \mathbb{N}\}$ como a aparece siempre en potencias de 15, sea $a_1 = a^{15}$ y análogamente $b_1 = b^{10}$,$c_1 = b^{14}$ y $d_1 = d^{15}$ por lo que tenemos que encontrar si el lenguage descrito como $\{a_1^{m'} b_1^{m'} c_1^{m'} d_1^{m'}\}$ es un CFL. pero por el lema del bombeo sea $m' > k$ donde k es la constante para el CFL.
donde tenemos que $z = a_1^{m'} b_1^{m'} c_1^{m'} d_1^{m'} = \beta\gamma\eta\theta\psi$ donde $\gamma\theta \neq \epsilon$ y $|\gamma\eta\theta| \leq k$ así $\gamma\eta\theta$ puede contener cuando más dos tipos diferentes de simbolos a_1, b_1, c_1, d_1 pues cada simbolo se repite k veces consecutivas. por lo que $\gamma^i\eta\theta^i$ inserta al menos un tipo de simbolo y cuando más dos, pero el lenguage requiere que los cuatro simbolos se agregen en las mismas cantidades, por lo que el lenguage no es CFG ■

b) $\{a^i b^j c^k d^l | i = k, j = l\}$
elijamos $i, j > k$ donde k es la constante para el CFL.
tenemos que $z = a^i b^j c^i d^j = \beta\gamma\eta\theta\psi$ donde $\gamma\theta \neq \epsilon$ y $|\gamma\eta\theta| \leq k$ así $\gamma\eta\theta$ puede contener cuando más dos tipos diferentes de simbolos a, b, c, d pues cada simbolo se repite más de k veces consecutivas. por lo que $\gamma^i\eta\theta^i$ inserta al menos un tipo de simbolo y cuando más dos, pero el lenguage requiere que los cuatro simbolos se agregen por pares en las mismas cantidades, por lo que el lenguage no es CFG ■

Problema 10 Describe detalladamente la ejecución del algoritmo CKY para decidir si la cadena $((x = 0) \vee f)$ es una expresión booleana:

Primero el subconjunto requerido de la grámatica de evaluación booleana en CNF.

$S \rightarrow P_a\ S_c|S\ S_{\vee 1}|V\ E_{=1}|v|f$
$P_a \rightarrow ($

$P_c \rightarrow)$
$S_= \rightarrow =$
$E_+ \rightarrow +$
$S_c \rightarrow S\ P_c$
$S_{\vee 1} \rightarrow S_\vee\ S$
$E_{=1} \rightarrow S_=\ S$
$E \rightarrow 0|1|2|..|9|E\ E_{+1}$
$E_{+1} \rightarrow E_+\ E$
$V \rightarrow x|y|z$

En este caso, como esta puesta con parentesis elimina rápidamente las opciones. Así

(	(	x	=	0	)	$\vee$	f	)	Cadena Reconocida
P_a	P_a	V	$S_=$	E	P_c	$S_\vee$	S	P_C	
-	-	-	-	-	-	-	-		
-	-	S	-	-	-	-	-		$x = 0$
-	S	-	-	-	-	-			$(x = 0$
-	S	-	-	-	-				$(x = 0)$
-	-	-	-	-					
-	S	-	-						$(x = 0) \vee f$
S	-								$((x = 0) \vee f$
S									$((x = 0) \vee f)$

Solo tiene una producción que genera la cadena deseada y empieza en S, por lo que la parsea de forma no ambigua