



UNIVERSITAT DE GIRONA

PRÀCTICA FINAL

Fonaments de la computació

Francesc Xavier Bullich Parra

Gil Gassó Rovira

Marc Sànchez Pifarré

Tutor de la pràctica
Jaume Rigau

December 27, 2018

Contents

1	Introducció	3
1.1	Definicions	3
2	Autòmata Finit	4
2.1	Definició del context	4
2.2	Dissenyant el nostre autòmata finit	4
2.2.1	Construïm l'autòmata sense RESET	5
2.2.2	Construïm l'Autòmata Complet	6
2.3	Definició formal de $FXBP_{DFA}$	7
2.3.1	Representació taula δ	7
2.4	Operacions regulars amb autòmates	8
2.5	Minimització de $FXBP_{DFA}$	8
2.6	NFA (Nondeterministic finit automata)	13
2.6.1	Disseny de $FXBP_{NFA}$	13
2.7	Definició formal de $FXBP_{NFA}$	13
2.7.1	Representació taula δ	13
2.7.2	Demostrem l'equivalència de $FXBP_{NFA}$ i $FXBP_{DFA}$	14
2.8	Regular expression	20
2.8.1	Equivalència de les expressions regulars i FA	20
2.8.2	Expressió regular equivalent a $FXBP_{DFA}$	20
2.9	Regular Grammar	22
2.9.1	$FXBP_{rg}$ (generada amb jflap).	22
2.9.2	Chomsky Normal Form	23
2.10	Referència al codi	23
2.11	Reconeixibilitat i Decidibilitat	23
3	Push-Down Automata	24
3.1	Definició del context	24
3.2	Anàlisi per el disseny	24
3.3	Definició informal de l'autòmata	24
3.4	Diagrama GGR_{PDA}	25
3.5	Definició formal de GGR_{PDA}	25
3.5.1	Taula δ GGR_{PDA}	26
3.6	Equivalència entre PDA i CFG	26
3.7	Gramàtica lliure de context	26
3.8	Ambigüitat	27
3.9	Chomsky Normal Form	27
3.10	Referència al codi	28
3.11	Reconeixibilitat i Decidibilitat	28

4	Turing Machine	29
4.1	Definició del Context	29
4.2	Disseny descendent i pseudocodi	29
4.2.1	Minimització	30
4.2.2	Minimitzar per files	30
4.2.3	Transposició	30
4.2.4	Algoritme eval	30
4.2.5	Algoritme performTick	31
4.2.6	Algoritme newCellValue	31
4.2.7	Referència al codi	31
4.3	Reconeixibilitat del problema	31
4.4	Decidibilitat del problema	32
4.4.1	Comportaments de l'evolució de patrons	33
4.4.2	Derivació	33
4.4.3	Estabilització	33
4.4.4	Oscil·lació	34
4.4.5	Creixement	36
5	Continguts de l'entrega	37
5.1	Codi	37
5.2	Exemples L^2	37
5.3	Sortides	37
5.4	Documentació	38

Chapter 1: Introducció

1.1 Definicions

Es defineixen una sèrie de noms que seran utilitzats al llarg de la pràctica per ajudar a la simplificació i l'enteniment de la mateixa.

- Definim $FXBP_{DFA}$ com l'autòmata finit determinista capaç de reconèixer el llenguatge $L(FXBP_{DFA})$.
- Definim $L(FXBP_{DFA})$ com el conjunt infinit de mots ω que accepta $FXBP_{DFA}$.
- Definim ω_{DFA} com un mot tal que $\omega_{DFA} \in L(FXBP_{DFA})$.
- Definim Σ^*FXBP_{DFA} com el conjunt de símbols amb el que es construeix el llenguatge $L(FXBP_{DFA})$.
- Definim context Φ com l'escenari ideal en el que l'autòmata a dissenyar es comportarà de manera correcta, $\Phi = \langle \Upsilon, \Psi \rangle$.
- Definim Υ com la precondició que s'ha de complir al utilitzar els nostres autòmats.
- Definim Ψ com la postcondició de l'autòmata.
- Definim GGR_{PDA} com l'autòmata de Pila capaç de reconèixer el llenguatge $L(GGR_{PDA})$.
- Definim $L(GGR_{PDA})$ com el conjunt finit de mots σ que accepta GGR_{PDA} .
- Definim ω_{PDA} com un mot del llenguatge $L(GGR_{PDA})$.
- Definim Σ^*GGR_{PDA} com el conjunt de símbols amb el que es construeix el llenguatge $L(GGR_{PDA})$.

Chapter 2: Autòmata Finit

2.1 Definició del context

Sigui Φ_{DFA} l'espai de funcionament lògic del nostre autòmata com l'espai estipulat a la Secció 3. Patrons[2]. Per tant acabem d'aquirir totes les definicions assumides a l'enunciat del problema, veure [2].

Fem les següents afirmacions sobre Υ_{DFA} :

- $\omega_{DFA} \in P_0 \wedge P_0 \in P$
- γ com a llesca continguda dins d'un P on $\gamma \in \omega_{DFA} \vee \gamma = \epsilon$

Fem les següents afirmacions sobre Ψ_{DFA} :

- retorna True quan $FXBP_{DFA}$ Accepta $\omega_{DFA} \leftrightarrow \exists \gamma \in P$ que compleix PIP.
- retorna False quan $\nexists \gamma \in P$ que compleix PIP.

on : PIP és la propietat dels patrons implícits parcials definida a [2] apartat 3.3.1.


2.2 Dissenyant el nostre autòmata finit


Primerament cal observar possibles propietats del problema que ens puguin servir per a la construcció de l'autòmata.

- Definim $4+$ com la seqüència de símbols '++++' d'un mot ω_{DFA} comprés dins del llenguatge $L(FXBP_{DFA})$.

Realitzem les següents observacions :

1. Abans i després de la seqüència $4+$ hi pot haver qualsevol símbol 0 o més vegades comprés dins de Σ^*FXBP_{DFA} .
2. Aïllada és la seqüència de símbols '-+-' dins de ω_{DFA}
3. Entre dues seqüències $4+$ hi trobem 3 aïllades.
4. 2 aïllades poden compartir el símbol inicial o el símbol final o ambdós.
5. S'accepta '-+-+-' com a dues cèl·lules aïllades on $-+$ comparteix - amb $+-$.
6. S'accepta '-+-+-' com a tres cèl·lules aïllades on $-+$ comparteix- amb $+-+-$ i $+-+-$ comparteix - amb $+-$.
7. Entre 2 aïllades hi pot aparèixer qualsevol combinació de símbols pertanyent a Σ^*FXBP_{DFA} .

8. El symbol  ens fa tornar a començar a cercar el PIP (RESET).

Dissenyarem doncs l'autòmata per construcció tenint en compte les anteriors propietats. En la construcció del nostre autòmata reduim el problema al tractament d'un subconjunt dels símbols de l'alfabet, concretament es construeix a partir del subconjunt $+, -$, ja que detectem que el símbol  actúa com a RESET. La funció del RESET en $FXBP_{DFA}$ és exemplificada al capítol 1 del llibre [1], concretament als exemples 1.15 i 1.17.

Incorporarem el RESET al nostre disseny a l'últim pas de la construcció.

Fem les següents definicions en l'escenari sense RESET :

- Definim pa com la seqüència $'-+'$
- Definim fa com la seqüència $'+-'$
- Definim l'operació CONCATENACIÓ amb el symbol $|$
- Definim 3aïllades com :

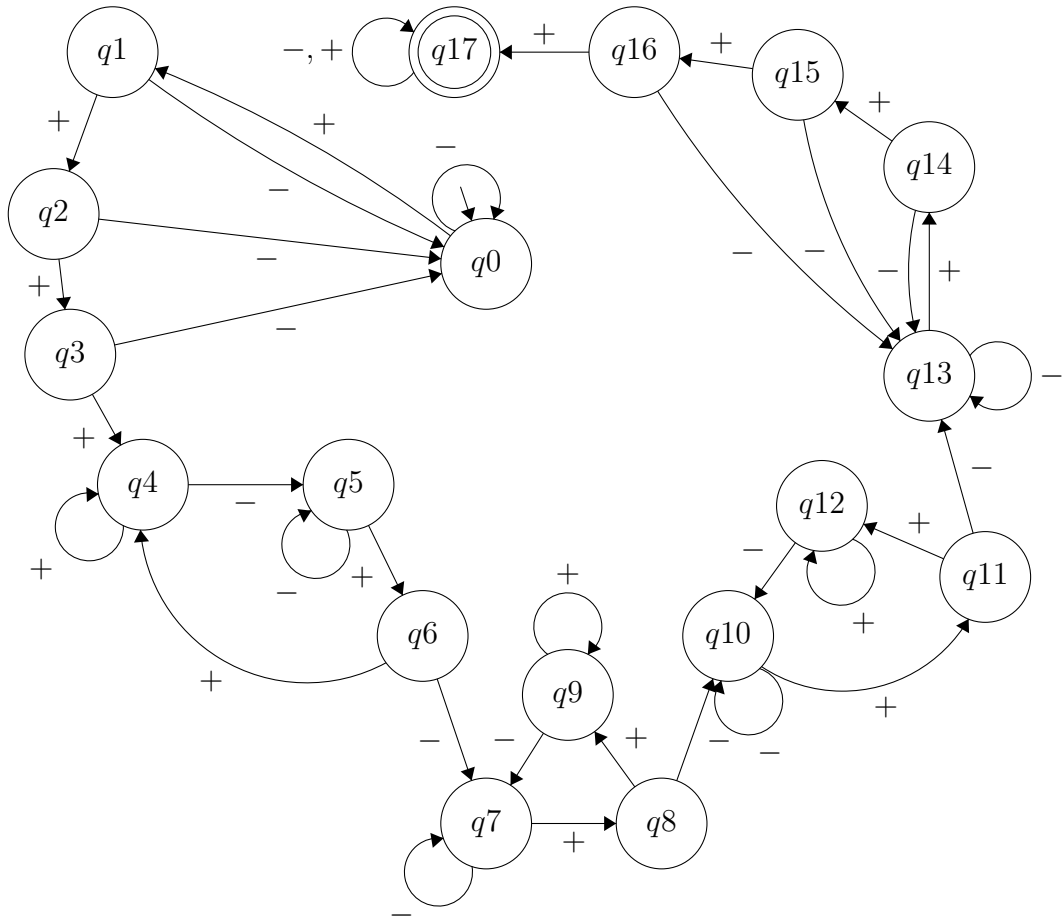
$$- (+-^* | pa | (-+-^*)^* | pa | (-+-^*)^* | pa | - | +-^*) \vee (+-^* | - | fa | (-+-^*)^* | fa | (-+-^*)^* | fa | +-^*)$$

Per tant podem concebre que en un entorn on no hi ha RESET es cerca si el mot conté la següent seqüència de símbols :

$$+-^* | 4+ | 3aïllades | 4+ | +-^*$$

2.2.1 Construïm l'autòmata sense RESET

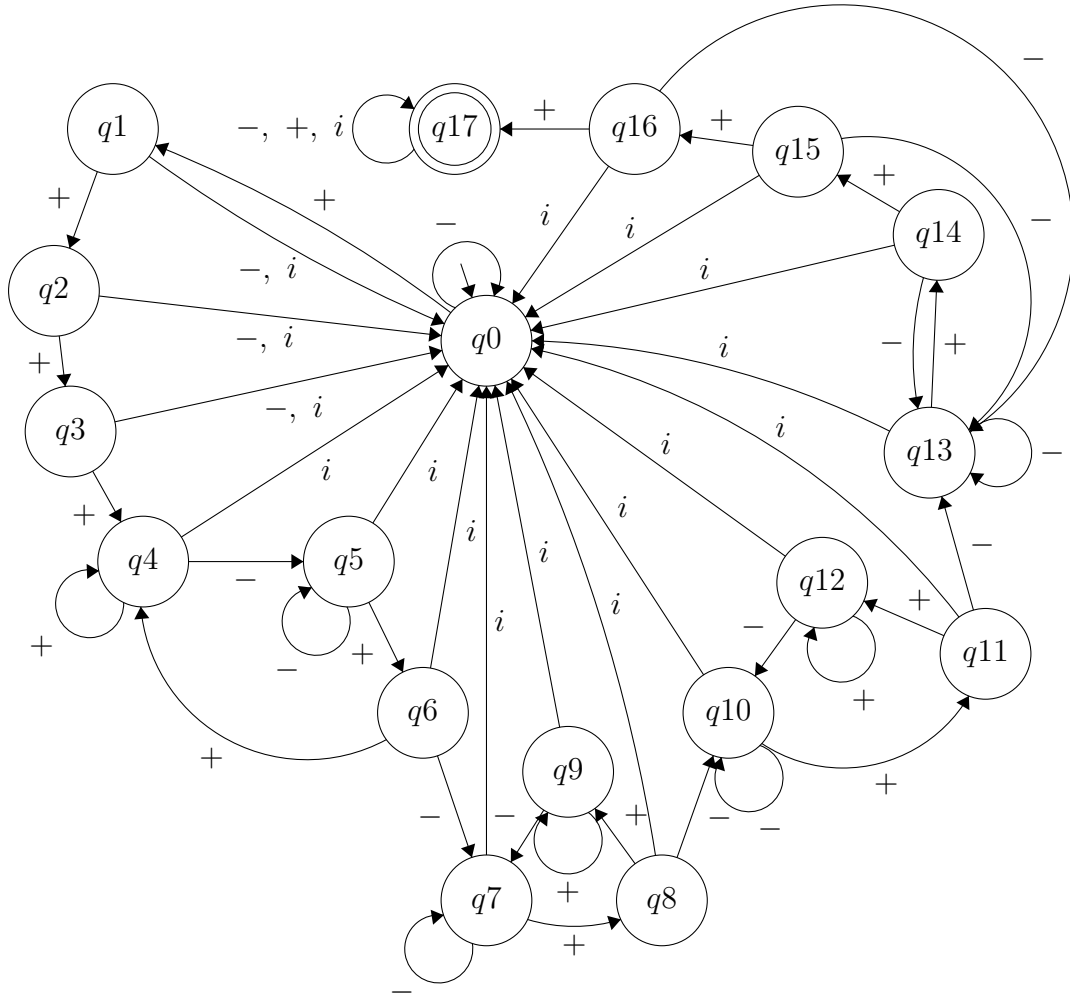
Veiem que els estats de l'autòmata van avançant a mesura que es van trobant el patró. Hi distingim cicles que ens interessin per controlar els casos que on per exemple hi pot haver qualsevol cosa entre cel·les actives o entre cel·les actives i principi o final del patró.



2.2.2 Construïm l'Autòmata Complet

Afegim RESET a l'autòmata, en aquest cas tots els estats menys l'estat final tindran una transició a l'estat inicial amb el símbol $\boxed{\leftarrow}$. L'estat final com que el mot ja està acceptat tindrà una transició a ell mateix amb el símbol $\boxed{\leftarrow}$.

En aquest disseny, per simplificar la codificació del caràcter $\boxed{\leftarrow}$ el codifiquem com una i .




2.3 Definició formal de $FXBP_{DFA}$

Definim formalment el nostre autòmata.

$$FXBP_{DFA} = \{Q, \Sigma^*, \delta, q_0, q_{17}\}$$

- $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9, q_{10}, q_{11}, q_{12}, q_{13}, q_{14}, q_{15}, q_{16}, q_{17}\}$
- $\Sigma^* = \{+, -, \boxed{\leftrightarrow}, \epsilon\}$
- $\delta = Q \times \Sigma^* \rightarrow Q$
- $q_0 \in Q$
- $q_{17} \subseteq Q$

2.3.1 Representació taula δ

Table 2.1: δ table			
State	+	-	
$\rightarrow q0$	q1	q0	q0
q1	q2	q0	q0
q2	q3	q0	q0
q3	q4	q0	q0
q4	q4	q5	q0
q5	q6	q5	q0
q6	q4	q7	q0
q7	q9	q7	q0
q8	q8	q7	q0
q9	q8	q10	q0
q10	q12	q10	q0
q11	q11	q10	q0
q12	q11	q13	q0
q13	q14	q13	q0
q14	q15	q13	q0
q15	q16	q13	q0
q16	q17	q13	q0
q17	q17	q17	q17

2.4 Operacions regulars amb autòmats

La definició 1.23 del llibre parla sobre les operacions que es poden realitzar sobre autòmats finits deterministes. En el nostre cas s'ha emprat una tècnica de construcció per el disseny de l'autòmata però es podria emprar una construcció de subautòmats i mitjançant la concatenació dels mateixos (representada amb el caràcter \bullet) construir el nostre $FXBP_{DFA}$. Hi ha hagut intents de la creació de l'autòmata mitjançant concatenació dels autòmats $FXBP_{DFA1}$ i $FXBP_{DFA2}$ intentant dividir el problema en :

$$FXBP_{DFA1} = 4+$$

$$FXBP_{DFA2} = 3aïllades$$

on:

$$4+ \bullet 3aïllades \bullet 4+$$

No s'ha seguit per la complexitat de tractar la funcionalitat del RESET, en el cas hipotètic que haguéssim presentat aquest tipus de construcció haguéssim derivat a el conjunt $FXBP_{DFA1}$, $FXBP_{DFA2}$ que concatenat mitjançant l'ús d'un autòmata finit indeterminista s'hauria pogut agrupar en $FXBP_{NFA}$.

2.5 Minimització de $FXBP_{DFA}$

L'objectiu d'aquesta secció és cercar el $FXBP_{DFA}$ mínim partint del $FXBP_{DFA}$.

Considerem que dos estats fan el mateix si i només si el resultat de les transicions és el mateix considerant tots els símbols de l'alfabet.

L'idea de la minimització parteix de la fusió d'estats. Podem trobar explicats els processos de minimització al document [3]. Podem utilitzar la manera manual aplicant l'algoritme de Hopcroft, o bé, podem optar per utilitzar UTM.

En aquest cas utilitzem l'algoritme de HopCroit per a la seva minimització implementant una matriu que ens permeti detectar les repeticions de les transicions donat un estat.

El nostre autòmata és mínim.

Table 2.2: Passos de $PI0$ - $PI2$

T		+	-	E	PI0	+	-	E	PI1	+	-	E	PI2
q0	A	B	A	A	1	1	1	1	1	1	1	1	1
q1	B	C	A	A	1	1	1	1	1	1	1	1	1
q2	C	D	A	A	1	1	1	1	1	1	1	1	1
q3	D	E	A	A	1	1	1	1	1	1	1	1	1
q4	E	E	F	A	1	1	1	1	1	1	1	1	1
q5	F	G	F	A	1	1	1	1	1	1	1	1	1
q6	G	E	H	A	1	1	1	1	1	1	1	1	1
q7	H	J	H	A	1	1	1	1	1	1	1	1	1
q8	I	I	H	A	1	1	1	1	1	1	1	1	1
q9	J	I	K	A	1	1	1	1	1	1	1	1	1
q10	K	M	K	A	1	1	1	1	1	1	1	1	1
q11	L	L	K	A	1	1	1	1	1	1	1	1	1
q12	M	L	N	A	1	1	1	1	1	1	1	1	1
q13	N	O	N	A	1	1	1	1	1	1	1	1	1
q14	O	P	N	A	1	1	1	1	1	1	1	1	1
q15	P	Q	N	A	1	1	1	1	1	3	1	1	4
q16	Q	R	N	A	1	2	1	1	3	2	1	1	3
q17	R	R	R	R	2	2	2	2	2	2	2	2	2

Table 2.3: Passos de $PI3$ - $PI5$

T	+	-	E	PI3	+	-	E	PI4	+	-	E	PI5
q0	1	1	1	1	1	1	1	1	1	1	1	1
q1	1	1	1	1	1	1	1	1	1	1	1	1
q2	1	1	1	1	1	1	1	1	1	1	1	1
q3	1	1	1	1	1	1	1	1	1	1	1	1
q4	1	1	1	1	1	1	1	1	1	1	1	1
q5	1	1	1	1	1	1	1	1	1	1	1	1
q6	1	1	1	1	1	1	1	1	1	1	1	1
q7	1	1	1	1	1	1	1	1	1	1	1	1
q8	1	1	1	1	1	1	1	1	1	1	1	1
q9	1	1	1	1	1	1	1	1	1	1	1	1
q10	1	1	1	1	1	1	1	1	1	1	1	1
q11	1	1	1	1	1	1	1	1	1	1	1	1
q12	1	1	1	1	1	1	1	1	1	6	1	7
q13	1	1	1	1	5	1	1	6	5	6	1	6
q14	4	1	1	5	4	1	1	5	4	6	1	5
q15	3	1	1	4	3	1	1	4	3	6	1	4
q16	2	1	1	3	2	1	1	3	2	6	1	3
q17	2	2	2	2	2	2	2	2	2	2	2	2

Table 2.4: Passos de $PI9$ - $PI11$

T	+	-	E	PI6	+	-	E	PI7	+	-	E	PI8
q0	1	1	1	1	1	1	1	1	1	1	1	1
q1	1	1	1	1	1	1	1	1	1	1	1	1
q2	1	1	1	1	1	1	1	1	1	1	1	1
q3	1	1	1	1	1	1	1	1	1	1	1	1
q4	1	1	1	1	1	1	1	1	1	1	1	1
q5	1	1	1	1	1	1	1	1	1	1	1	1
q6	1	1	1	1	1	1	1	1	1	1	1	1
q7	1	1	1	1	1	1	1	1	9	1	1	11
q8	1	1	1	1	1	1	1	1	1	1	1	1
q9	1	1	1	1	1	8	1	9	1	8	1	9
q10	7	1	1	8	7	8	1	8	7	8	1	8
q11	1	1	1	1	1	8	1	9	9	8	1	10
q12	1	6	1	7	1	6	1	7	9	6	1	7
q13	5	6	1	6	5	6	1	6	5	6	1	6
q14	4	6	1	5	4	6	1	5	4	6	1	5
q15	3	6	1	4	3	6	1	4	3	6	1	4
q16	2	6	1	3	2	6	1	3	2	6	1	3
q17	2	2	2	2	2	2	2	2	2	2	2	2

Table 2.5: Passos de $PI12$ - $PI14$

T	+	-	E	PI9	+	-	E	PI10	+	-	E	PI11
q0	1	1	1	1	1	1	1	1	1	1	1	1
q1	1	1	1	1	1	1	1	1	1	1	1	1
q2	1	1	1	1	1	1	1	1	1	1	1	1
q3	1	1	1	1	1	1	1	1	1	1	1	1
q4	1	1	1	1	1	1	1	1	1	14	1	15
q5	1	1	1	1	12	1	1	14	12	14	1	14
q6	1	11	1	12	1	11	1	12	1	11	1	12
q7	9	11	1	11	9	11	1	11	9	11	1	11
q8	1	11	1	12	12	11	1	13	13	11	1	13
q9	1	8	1	9	12	8	1	9	13	8	1	9
q10	7	8	1	8	7	8	1	8	7	8	1	8
q11	10	8	1	10	10	8	1	10	10	8	1	10
q12	10	6	1	7	10	6	1	7	10	6	1	7
q13	5	6	1	6	5	6	1	6	5	6	1	6
q14	4	6	1	5	4	6	1	5	4	6	1	5
q15	3	6	1	4	3	6	1	4	3	6	1	4
q16	2	6	1	3	2	6	1	3	2	6	1	3
q17	2	2	2	2	2	2	2	2	2	2	2	2

Table 2.6: Passos de $PI15$ - $PI16$

T	+	-	E	PI12	+	-	E	PI13	+	-	E	PI14
q0	1	1	1	1	1	1	1	1	1	1	1	1
q1	1	1	1	1	1	1	1	1	17	1	1	18
q2	1	1	1	1	16	1	1	17	16	1	1	17
q3	15	1	1	16	15	1	1	16	15	1	1	16
q4	15	14	1	15	15	14	1	15	15	14	1	15
q5	12	14	1	14	12	14	1	14	12	14	1	14
q6	15	11	1	12	15	11	1	12	15	11	1	12
q7	9	11	1	11	9	11	1	11	9	11	1	11
q8	13	11	1	13	13	11	1	13	13	11	1	13
q9	13	8	1	9	13	8	1	9	13	8	1	9
q10	7	8	1	8	7	8	1	8	7	8	1	8
q11	10	8	1	10	10	8	1	10	10	8	1	10
q12	10	6	1	7	10	6	1	7	10	6	1	7
q13	5	6	1	6	5	6	1	6	5	6	1	6
q14	4	6	1	5	4	6	1	5	4	6	1	5
q15	3	6	1	4	3	6	1	4	3	6	1	4
q16	2	6	1	3	2	6	1	3	2	6	1	3
q17	2	2	2	2	2	2	2	2	2	2	2	2

Table 2.7: Passos de $PI0$ - $PI2$

T	+	-	E	PI15	+	-	E	PI16
q0	18	1	1	1	18	1	1	1
q1	17	1	1	18	17	1	1	18
q2	16	1	1	17	16	1	1	17
q3	15	1	1	16	15	1	1	16
q4	15	14	1	15	15	14	1	15
q5	12	14	1	14	12	14	1	14
q6	15	11	1	12	15	11	1	12
q7	9	11	1	11	9	11	1	11
q8	13	11	1	13	13	11	1	13
q9	13	8	1	9	13	8	1	9
q10	7	8	1	8	7	8	1	8
q11	10	8	1	10	10	8	1	10
q12	10	6	1	7	10	6	1	7
q13	5	6	1	6	5	6	1	6
q14	4	6	1	5	4	6	1	5
q15	3	6	1	4	3	6	1	4
q16	2	6	1	3	2	6	1	3
q17	2	2	2	2	2	2	2	2

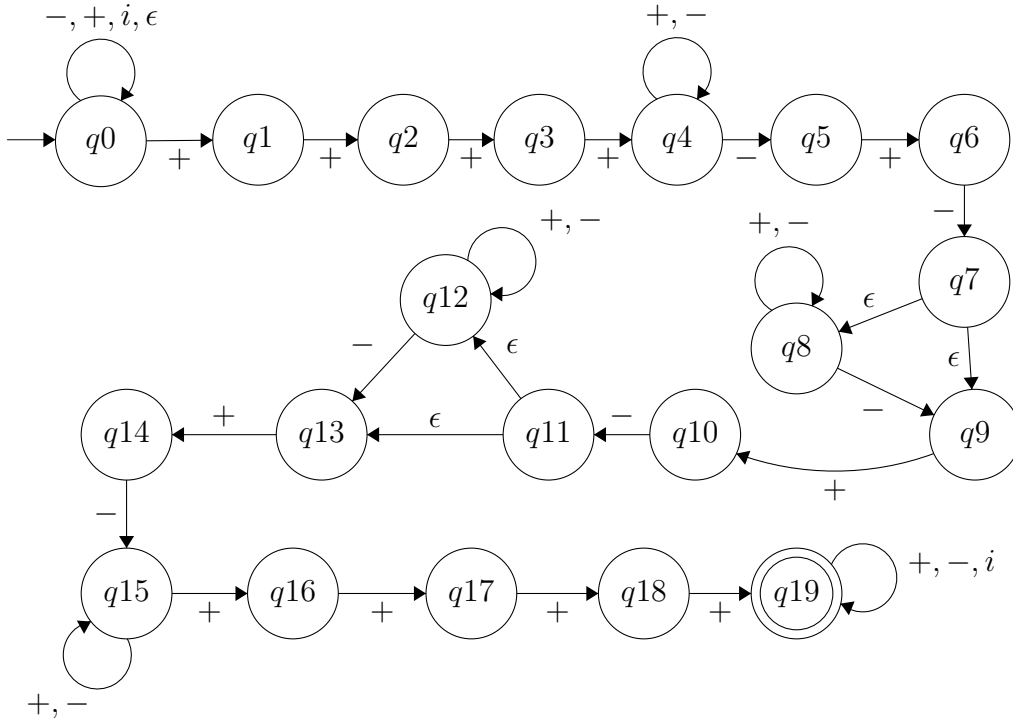
2.6 NFA (Nondeterministic finit automata)

L'equivalència de NFA i DFA està demostrada al capítol 1.2 del llibre [1], donat un autòmata NFA existeix un DFA equivalent. Per equivalència entenem que ambdós reconeixen el mateix llenguatge, en aquest cas el llenguatge $L(FXBP_{DFA})$.

Definim doncs $FXBP_{NFA}$ com l'autòmata finit indeterminista equivalent a $FXBP_{DFA}$.

2.6.1 Disseny de $FXBP_{NFA}$

Dissenyem ANFA, en aquest cas ens convé només cercar el camí que ens porta directes a Q_{Accept} . Directament tracem un camí acceptador i el modifiquem per que es comporti de manera correcte afegint les branques indeterministes oportunes.



2.7 Definició formal de $FXBP_{NFA}$


Definim formalment el nostre autòmata.

$$FXBP_{DFA} = \{Q, \Sigma^*, \delta, q_0, q_{19}\}$$

- $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9, q_{10}, q_{11}, q_{12}, q_{13}, q_{14}, q_{15}, q_{16}, q_{17}, q_{18}, q_{19}\}$
- $\Sigma^* = \{ +, -, \boxed{\leftarrow}, \epsilon \}$
- $\delta = Q \times \Sigma^* \rightarrow P(Q)$ on $P(Q)$ son possibles subconjunts de Q
- $q_0 \in Q$
- $q_{19} \subseteq Q$

2.7.1 Representació taula δ

Table 2.8: δ table

State	+	-		ϵ
q0	{q0,q1}	{q0}	{q0}	{q0}
q1	{q2}	\emptyset	\emptyset	\emptyset
q2	{q3}	\emptyset	\emptyset	\emptyset
q3	{q4}	\emptyset	\emptyset	\emptyset
q4	{q4}	{q4,q5}	\emptyset	\emptyset
q5	{q6}	\emptyset	\emptyset	\emptyset
q6	\emptyset	{q7}	\emptyset	\emptyset
q7	\emptyset	\emptyset	\emptyset	{q8,q9}
q8	{q8}	{q8,q9}	\emptyset	\emptyset
q9	{q10}	\emptyset	\emptyset	\emptyset
q10	\emptyset	{q11}	\emptyset	\emptyset
q11	\emptyset	\emptyset	\emptyset	{q12,q13}
q12	{q12}	{q12,q13}	\emptyset	\emptyset
q13	{q14}	\emptyset	\emptyset	\emptyset
q14	\emptyset	{q15}	\emptyset	\emptyset
q15	{q15,q16}	{q15}	\emptyset	\emptyset
q16	{q17}	\emptyset	\emptyset	\emptyset
q17	{q18}	\emptyset	\emptyset	\emptyset
q18	{q19}	\emptyset	\emptyset	\emptyset
q19	{q19}	{q19}	{q19}	\emptyset

2.7.2 Demostrem l'equivalència de $FXBP_{NFA}$ i $FXBP_{DFA}$

Per demostrar l'equivalència dels autòmats apliquem un procés de determinització sobre ANFA obtenint ADFA'. Ens ajudem de [UTM](referència al programa).

Expressem l'autòmata en format UTM mitjançant un fitxer .def tenint en compte que UTM contempla alfabet Σ^*

```

1
2 Sigma=[-+i]
3 Final=[q19]
4 -----
5 | Description: ANFA
6 |   Sigma = {-,+,i}
7 |   L = "exists the sequence '-+-' 3 times flanked both sides by the sequence '++++'"
8 | Function: f : Sigma^* -> Boolean
9 |   f(w) = 1, if word in L
10 |          = 0, otherwise
11 | Input: word w (e.g., -----+-+--+--+--+--+-----+i++++-----+-+--+--+-----+-+--+i
12 |          -----+-+--+--+--+--+--+-----+i)
13 |   where i = \Return
14 | Output: 0 or 1
15 | Date: November, 2018
16 | Version: 1.0
17 | UTM-version: 0.2357111317192329313741434753
18 | Author: Copyright 2011 Gil Gasso Rovira, Marc Sanchez Pifarre, Francesc Xavier Bullich Parra
19 -----
19 [q0|-]=[q0]
20 [q0|+]=[q0]
21 [q0|i]=[q0]
22 [q0|.]=[q0]
23 [q0|+]=[q1]
24 [q1|+]=[q2]

```

$$\begin{array}{l}
25 \quad [q2+] = [q3] \\
26 \quad [q3+] = [q4] \\
27 \quad [q4+] = [q4] \\
28 \quad [q4-] = [q4] \\
29 \quad [q4-] = [q5] \\
30 \quad [q5+] = [q6] \\
31 \quad [q6-] = [q7] \\
32 \quad [q7.] = [q8] \\
33 \quad [q7.] = [q9] \\
34 \quad [q8+] = [q8] \\
35 \quad [q8-] = [q8] \\
36 \quad [q8-] = [q9] \\
37 \quad [q9+] = [q10] \\
38 \quad [q10-] = [q11] \\
39 \quad [q11.] = [q12] \\
40 \quad [q11.] = [q13] \\
41 \quad [q12+] = [q12] \\
42 \quad [q12-] = [q12] \\
43 \quad [q12-] = [q13] \\
44 \quad [q13+] = [q14] \\
45 \quad [q14-] = [q15] \\
46 \quad [q15-] = [q15] \\
47 \quad [q15+] = [q15] \\
48 \quad [q15+] = [q16] \\
49 \quad [q16+] = [q17] \\
50 \quad [q17+] = [q18] \\
51 \quad [q18+] = [q19] \\
52 \quad [q19+] = [q19] \\
53 \quad [q19-] = [q19] \\
54 \quad [q19i] = [q19]
\end{array}$$

Aplicant el procés de determinització que permet UTM obtenim $FXBP_{DFA}$ on $FXBP_{DFA}$ Comprés dins del DFA. Tenir en compte que tal com comenta la *PROOF IDEA de la pàgina 55 de la referència [1]*, NFA de k estats es pot determinitzar en un DFA de n^k estats, és per aquest motiu que la següent representació pot semblar monstruosa.

```

1 | =====
2 | Description: ADFA
3 |   Sigma = {-,+,i}
4 |   L = "exists the sequence '-+-' 3 times flanked both sides by the sequence '++++'"
5 | Function: f : Sigma^* -> Boolean
6 |   f(w) = 1, if word in L
7 |   = 0, otherwise
8 | Input: word w (e.g., -----+-+--+--+--+--+-----+i++++-+-+--+--+-----+-+++++i
   | -----+-+--+--+--+--+--+-----+i)
9 |   where i = \Return
10 | Output: 0 or 1
11 | Date: November, 2018
12 | Version: 1.0
13 | UTM-version: 0.2357111317192329313741434753
14 | Author: Copyright 2011 Gil Gasso Rovira, Marc Sanchez Pifarre, Francesc Xavier Bullich Parra
15 | =====
16 Q = {{q1, q8, q6, q12, q10, q14, q15, q4, q16, q19, q0}, {q1, q8, q6, q12, q10, q14, q15, q4, q16, q0}, {q1
   | , q8, q6, q12, q10, q14, q4, q19, q0}, {q1, q8, q6, q12, q10, q14, q4, q0}, {q1, q8, q6, q10, q4, q19,
   | q0}, {q1, q8, q6, q10, q4, q0}, {q1, q8, q2, q12, q3, q15, q4, q16, q17, q18, q19, q0}, {q1, q8, q2,
   | q12, q3, q15, q4, q16, q17, q18, q0}, {q1, q8, q2, q12, q3, q4, q19, q0}, {q1, q8, q2, q12, q3, q4, q0
   | }, {q1, q8, q2, q12, q15, q4, q16, q17, q19, q0}, {q1, q8, q2, q12, q15, q4, q16, q17, q0}, {q1, q8, q2
   | , q12, q4, q19, q0}, {q1, q8, q2, q12, q4, q0}, {q1, q8, q2, q3, q4, q19, q0}, {q1, q8, q2, q3, q4, q0
   | }, {q1, q8, q2, q4, q19, q0}, {q1, q8, q2, q4, q0}, {q1, q6, q4, q19, q0}, {q1, q6, q4, q0}, {q1, q2,
   | q3, q4, q19, q0}, {q1, q2, q3, q4, q0}, {q1, q2, q3, q19, q0}, {q1, q2, q3, q0}, {q1, q2, q4, q19, q0},
   | {q1, q2, q4, q0}, {q1, q2, q19, q0}, {q1, q2, q0}, {q1, q19, q0}, {q1, q0}, {q8, q5, q7, q12, q9, q11}

```



```

    q13, q15, q4, q19, q0}, {q8, q5, q7, q12, q9, q11, q13, q15, q4, q0}, {q8, q5, q7, q12, q9, q11, q13,
    q4, q19, q0}, {q8, q5, q7, q12, q9, q11, q13, q4, q0}, {q8, q5, q7, q9, q4, q19, q0}, {q8, q5, q7, q9,
    q4, q0}, {q8, q5, q12, q9, q13, q15, q4, q19, q0}, {q8, q5, q12, q9, q13, q15, q4, q0}, {q8, q5, q12,
    q9, q13, q4, q19, q0}, {q8, q5, q12, q9, q13, q4, q0}, {q8, q5, q9, q4, q19, q0}, {q8, q5, q9, q4, q0},
    {q5, q4, q19, q0}, {q5, q4, q0}, {q19, q0}, {q0}}
17 Sigma = {+, -, i}
18 Initial state = {q0}
19 Final = {{q1, q8, q6, q12, q10, q14, q15, q4, q16, q19, q0}, {q1, q8, q6, q12, q10, q14, q4, q19, q0}, {q1,
    q8, q6, q10, q4, q19, q0}, {q1, q8, q2, q12, q3, q15, q4, q16, q17, q18, q19, q0}, {q1, q8, q2, q12,
    q3, q4, q19, q0}, {q1, q8, q2, q12, q15, q4, q16, q17, q19, q0}, {q1, q8, q2, q12, q4, q19, q0}, {q1,
    q8, q2, q3, q4, q19, q0}, {q1, q8, q2, q4, q19, q0}, {q1, q6, q4, q19, q0}, {q1, q2, q3, q4, q19, q0},
    {q1, q2, q3, q19, q0}, {q1, q2, q4, q19, q0}, {q1, q2, q19, q0}, {q1, q19, q0}, {q8, q5, q7, q12, q9,
    q11, q13, q15, q4, q19, q0}, {q8, q5, q7, q12, q9, q11, q13, q4, q19, q0}, {q8, q5, q7, q9, q4, q19,
    q0}, {q8, q5, q12, q9, q13, q15, q4, q19, q0}, {q8, q5, q12, q9, q13, q4, q19, q0}, {q8, q5, q9, q4,
    q19, q0}, {q5, q4, q19, q0}, {q19, q0}}
20 =====
21 delta({q1, q8, q6, q12, q10, q14, q15, q4, q16, q19, q0}, +) = {{q1, q8, q2, q12, q15, q4, q16, q17, q19,
    q0}}
22 delta({q1, q8, q6, q12, q10, q14, q15, q4, q16, q19, q0}, -) = {{q8, q5, q7, q12, q9, q11, q13, q15, q4,
    q19, q0}}
23 delta({q1, q8, q6, q12, q10, q14, q15, q4, q16, q19, q0}, i) = {{q19, q0}}
24 delta({q1, q8, q6, q12, q10, q14, q15, q4, q16, q0}, +) = {{q1, q8, q2, q12, q15, q4, q16, q17, q0}}
25 delta({q1, q8, q6, q12, q10, q14, q15, q4, q16, q0}, -) = {{q8, q5, q7, q12, q9, q11, q13, q15, q4, q0}}
26 delta({q1, q8, q6, q12, q10, q14, q15, q4, q16, q0}, i) = {{q0}}
27 delta({q1, q8, q6, q12, q10, q14, q4, q19, q0}, +) = {{q1, q8, q2, q12, q4, q19, q0}}
28 delta({q1, q8, q6, q12, q10, q14, q4, q19, q0}, -) = {{q8, q5, q7, q12, q9, q11, q13, q15, q4, q19, q0}}
29 delta({q1, q8, q6, q12, q10, q14, q4, q19, q0}, i) = {{q19, q0}}
30 delta({q1, q8, q6, q12, q10, q14, q4, q0}, +) = {{q1, q8, q2, q12, q4, q0}}
31 delta({q1, q8, q6, q12, q10, q14, q4, q0}, -) = {{q8, q5, q7, q12, q9, q11, q13, q15, q4, q0}}
32 delta({q1, q8, q6, q12, q10, q14, q4, q0}, i) = {{q0}}
33 delta({q1, q8, q6, q10, q4, q19, q0}, +) = {{q1, q8, q2, q4, q19, q0}}
34 delta({q1, q8, q6, q10, q4, q19, q0}, -) = {{q8, q5, q7, q12, q9, q11, q13, q4, q19, q0}}
35 delta({q1, q8, q6, q10, q4, q19, q0}, i) = {{q19, q0}}
36 delta({q1, q8, q6, q10, q4, q0}, +) = {{q1, q8, q2, q4, q0}}
37 delta({q1, q8, q6, q10, q4, q0}, -) = {{q8, q5, q7, q12, q9, q11, q13, q4, q0}}
38 delta({q1, q8, q6, q10, q4, q0}, i) = {{q0}}
39 delta({q1, q8, q2, q12, q3, q15, q4, q16, q17, q18, q19, q0}, +) = {{q1, q8, q2, q12, q3, q15, q4, q16, q17,
    q18, q19, q0}}
40 delta({q1, q8, q2, q12, q3, q15, q4, q16, q17, q18, q19, q0}, -) = {{q8, q5, q12, q9, q13, q15, q4, q19, q0}}
41 delta({q1, q8, q2, q12, q3, q15, q4, q16, q17, q18, q19, q0}, i) = {{q19, q0}}
42 delta({q1, q8, q2, q12, q3, q15, q4, q16, q17, q18, q0}, +) = {{q1, q8, q2, q12, q3, q15, q4, q16, q17, q18,
    q19, q0}}
43 delta({q1, q8, q2, q12, q3, q15, q4, q16, q17, q18, q0}, -) = {{q8, q5, q12, q9, q13, q15, q4, q0}}
44 delta({q1, q8, q2, q12, q3, q15, q4, q16, q17, q18, q0}, i) = {{q0}}
45 delta({q1, q8, q2, q12, q3, q4, q19, q0}, +) = {{q1, q8, q2, q12, q3, q4, q19, q0}}
46 delta({q1, q8, q2, q12, q3, q4, q19, q0}, -) = {{q8, q5, q12, q9, q13, q4, q19, q0}}
47 delta({q1, q8, q2, q12, q3, q4, q19, q0}, i) = {{q19, q0}}
48 delta({q1, q8, q2, q12, q3, q4, q0}, +) = {{q1, q8, q2, q12, q3, q4, q0}}
49 delta({q1, q8, q2, q12, q3, q4, q0}, -) = {{q8, q5, q12, q9, q13, q4, q0}}
50 delta({q1, q8, q2, q12, q3, q4, q0}, i) = {{q0}}
51 delta({q1, q8, q2, q12, q15, q4, q16, q17, q19, q0}, +) = {{q1, q8, q2, q12, q3, q15, q4, q16, q17, q18,
    q19, q0}}
52 delta({q1, q8, q2, q12, q15, q4, q16, q17, q19, q0}, -) = {{q8, q5, q12, q9, q13, q15, q4, q19, q0}}
53 delta({q1, q8, q2, q12, q15, q4, q16, q17, q19, q0}, i) = {{q19, q0}}
54 delta({q1, q8, q2, q12, q15, q4, q16, q17, q0}, +) = {{q1, q8, q2, q12, q3, q15, q4, q16, q17, q18, q0}}
55 delta({q1, q8, q2, q12, q15, q4, q16, q17, q0}, -) = {{q8, q5, q12, q9, q13, q15, q4, q0}}
56 delta({q1, q8, q2, q12, q15, q4, q16, q17, q0}, i) = {{q0}}
57 delta({q1, q8, q2, q12, q4, q19, q0}, +) = {{q1, q8, q2, q12, q3, q4, q19, q0}}
58 delta({q1, q8, q2, q12, q4, q19, q0}, -) = {{q8, q5, q12, q9, q13, q4, q19, q0}}

```

```

59 delta({q1, q8, q2, q12, q4, q19, q0}, i) = {{q19, q0}}
60 delta({q1, q8, q2, q12, q4, q0}, +) = {{q1, q8, q2, q12, q3, q4, q0}}
61 delta({q1, q8, q2, q12, q4, q0}, -) = {{q8, q5, q12, q9, q13, q4, q0}}
62 delta({q1, q8, q2, q12, q4, q0}, i) = {{q0}}
63 delta({q1, q8, q2, q3, q4, q19, q0}, +) = {{q1, q8, q2, q3, q4, q19, q0}}
64 delta({q1, q8, q2, q3, q4, q19, q0}, -) = {{q8, q5, q9, q4, q19, q0}}
65 delta({q1, q8, q2, q3, q4, q19, q0}, i) = {{q19, q0}}
66 delta({q1, q8, q2, q3, q4, q0}, +) = {{q1, q8, q2, q3, q4, q0}}
67 delta({q1, q8, q2, q3, q4, q0}, -) = {{q8, q5, q9, q4, q0}}
68 delta({q1, q8, q2, q3, q4, q0}, i) = {{q0}}
69 delta({q1, q8, q2, q4, q19, q0}, +) = {{q1, q8, q2, q3, q4, q19, q0}}
70 delta({q1, q8, q2, q4, q19, q0}, -) = {{q8, q5, q9, q4, q19, q0}}
71 delta({q1, q8, q2, q4, q19, q0}, i) = {{q19, q0}}
72 delta({q1, q8, q2, q4, q0}, +) = {{q1, q8, q2, q3, q4, q0}}
73 delta({q1, q8, q2, q4, q0}, -) = {{q8, q5, q9, q4, q0}}
74 delta({q1, q8, q2, q4, q0}, i) = {{q0}}
75 delta({q1, q6, q4, q19, q0}, +) = {{q1, q2, q4, q19, q0}}
76 delta({q1, q6, q4, q19, q0}, -) = {{q8, q5, q7, q9, q4, q19, q0}}
77 delta({q1, q6, q4, q19, q0}, i) = {{q19, q0}}
78 delta({q1, q6, q4, q0}, +) = {{q1, q2, q4, q0}}
79 delta({q1, q6, q4, q0}, -) = {{q8, q5, q7, q9, q4, q0}}
80 delta({q1, q6, q4, q0}, i) = {{q0}}
81 delta({q1, q2, q3, q4, q19, q0}, +) = {{q1, q2, q3, q4, q19, q0}}
82 delta({q1, q2, q3, q4, q19, q0}, -) = {{q5, q4, q19, q0}}
83 delta({q1, q2, q3, q4, q19, q0}, i) = {{q19, q0}}
84 delta({q1, q2, q3, q4, q0}, +) = {{q1, q2, q3, q4, q0}}
85 delta({q1, q2, q3, q4, q0}, -) = {{q5, q4, q0}}
86 delta({q1, q2, q3, q4, q0}, i) = {{q0}}
87 delta({q1, q2, q3, q19, q0}, +) = {{q1, q2, q3, q4, q19, q0}}
88 delta({q1, q2, q3, q19, q0}, -) = {{q19, q0}}
89 delta({q1, q2, q3, q19, q0}, i) = {{q19, q0}}
90 delta({q1, q2, q3, q0}, +) = {{q1, q2, q3, q4, q0}}
91 delta({q1, q2, q3, q0}, -) = {{q0}}
92 delta({q1, q2, q3, q0}, i) = {{q0}}
93 delta({q1, q2, q4, q19, q0}, +) = {{q1, q2, q3, q4, q19, q0}}
94 delta({q1, q2, q4, q19, q0}, -) = {{q5, q4, q19, q0}}
95 delta({q1, q2, q4, q19, q0}, i) = {{q19, q0}}
96 delta({q1, q2, q4, q0}, +) = {{q1, q2, q3, q4, q0}}
97 delta({q1, q2, q4, q0}, -) = {{q5, q4, q0}}
98 delta({q1, q2, q4, q0}, i) = {{q0}}
99 delta({q1, q2, q19, q0}, +) = {{q1, q2, q3, q19, q0}}
100 delta({q1, q2, q19, q0}, -) = {{q19, q0}}
101 delta({q1, q2, q19, q0}, i) = {{q19, q0}}
102 delta({q1, q2, q0}, +) = {{q1, q2, q3, q0}}
103 delta({q1, q2, q0}, -) = {{q0}}
104 delta({q1, q2, q0}, i) = {{q0}}
105 delta({q1, q19, q0}, +) = {{q1, q2, q19, q0}}
106 delta({q1, q19, q0}, -) = {{q19, q0}}
107 delta({q1, q19, q0}, i) = {{q19, q0}}
108 delta({q1, q0}, +) = {{q1, q2, q0}}
109 delta({q1, q0}, -) = {{q0}}
110 delta({q1, q0}, i) = {{q0}}
111 delta({q8, q5, q7, q12, q9, q11, q13, q15, q4, q19, q0}, +) = {{q1, q8, q6, q12, q10, q14, q15, q4, q16,
    q19, q0}}
112 delta({q8, q5, q7, q12, q9, q11, q13, q15, q4, q19, q0}, -) = {{q8, q5, q12, q9, q13, q15, q4, q19, q0}}
113 delta({q8, q5, q7, q12, q9, q11, q13, q15, q4, q19, q0}, i) = {{q19, q0}}
114 delta({q8, q5, q7, q12, q9, q11, q13, q15, q4, q0}, +) = {{q1, q8, q6, q12, q10, q14, q15, q4, q16, q0}}
115 delta({q8, q5, q7, q12, q9, q11, q13, q15, q4, q0}, -) = {{q8, q5, q12, q9, q13, q15, q4, q0}}
116 delta({q8, q5, q7, q12, q9, q11, q13, q15, q4, q0}, i) = {{q0}}
117 delta({q8, q5, q7, q12, q9, q11, q13, q4, q19, q0}, +) = {{q1, q8, q6, q12, q10, q14, q4, q19, q0}}

```



```

-----+--+--+--+--+--+--+--+-----+i)
10 |   where i = \Return
11 |   Output: 0 or 1
12 |   Date: November, 2018
13 |   Version: 1.0
14 |   UTM-version: 0.2357111317192329313741434753
15 |   Author: Copyright 2011 Gil Gasso Rovira, Marc Sanchez Pifarre, Francesc Xavier Bullich Parra
16 |   =====
17 |   Q = {1, 10, 11, 12, 13, 14, 15, 16, 17, 18, 2, 3, 4, 5, 6, 7, 8, 9}
18 |   Sigma = {+, -, i}
19 |   Initial state = 1
20 |   Final = {3}
21 |   =====
22 |   delta(1, +) = {14}
23 |   delta(1, -) = {1}
24 |   delta(1, i) = {1}
25 |   delta(10, +) = {11}
26 |   delta(10, -) = {17}
27 |   delta(10, i) = {1}
28 |   delta(11, +) = {11}
29 |   delta(11, -) = {18}
30 |   delta(11, i) = {1}
31 |   delta(12, +) = {11}
32 |   delta(12, -) = {1}
33 |   delta(12, i) = {1}
34 |   delta(13, +) = {12}
35 |   delta(13, -) = {1}
36 |   delta(13, i) = {1}
37 |   delta(14, +) = {13}
38 |   delta(14, -) = {1}
39 |   delta(14, i) = {1}
40 |   delta(15, +) = {2}
41 |   delta(15, -) = {15}
42 |   delta(15, i) = {1}
43 |   delta(16, +) = {4}
44 |   delta(16, -) = {16}
45 |   delta(16, i) = {1}
46 |   delta(17, +) = {5}
47 |   delta(17, -) = {17}
48 |   delta(17, i) = {1}
49 |   delta(18, +) = {10}
50 |   delta(18, -) = {18}
51 |   delta(18, i) = {1}
52 |   delta(2, +) = {8}
53 |   delta(2, -) = {15}
54 |   delta(2, i) = {1}
55 |   delta(3, +) = {3}
56 |   delta(3, -) = {3}
57 |   delta(3, i) = {3}
58 |   delta(4, +) = {7}
59 |   delta(4, -) = {15}
60 |   delta(4, i) = {1}
61 |   delta(5, +) = {9}
62 |   delta(5, -) = {16}
63 |   delta(5, i) = {1}
64 |   delta(6, +) = {3}
65 |   delta(6, -) = {15}
66 |   delta(6, i) = {1}
67 |   delta(7, +) = {7}
68 |   delta(7, -) = {16}

```

```

69 delta(7, i) = {1}
70 delta(8, +) = {6}
71 delta(8, -) = {15}
72 delta(8, i) = {1}
73 delta(9, +) = {9}
74 delta(9, -) = {17}
75 delta(9, i) = {1}
76 =====
77 End MFA definition
78 =====

```

Reconeixen el mateix llenguatge, per tant demostrem per construcció l'equivalència dels $FXBP_{DFA}$ i $FXBP_{NFA}$.

2.8 Regular expression

La definició formal d'una expressió regular està explicada al *capítol 1.3 de la referència [1]*, concretament a la *Definition 1.52*. S'explica que de la mateixa manera que una expressió aritmètica retorna un valor (numèric) el valor d'una expressió regular és un llenguatge.

“When we want to distinguish between a regular expression R and the language that it describes, we write $L(R)$ to be the language of R .” - Michael Sipser

Es poden trobar exemples d'expressions regulars a l'exemple 1.53, pàgina 65 de la referència [1].

2.8.1 Equivalència de les expressions regulars i FA

Al llibre es comenta que les expressions regulars i els autòmats finits son equivalents en quan el poder descriptiu. Sent LR un llenguatge regular, al *Theorem 1.54 de la referència [1]* es demostra que un llenguatge és regular si i només si existeix una expressió regular que el descriu (La demostració és ambidireccional).

Podem doncs extreure l'expressió regular sobre l'autòmata $ADFA$ utilitzant l'algoritme explicat al llibre. En aquest cas l'automatització d'aquest procediment està disponible mitjançant el *programa referenciat a [6]*.

Sent doncs $FXBP_{MDFA}$ l'autòmat $FXBP_{DFA}$ representat de manera mínima definim $FXBP_{RE}$ com l'expressió regular equivalent a $FXBP_{DFA}$.

2.8.2 Expressió regular equivalent a $FXBP_{DFA}$

Definim els canvis de símbols :

- $p = -$
- $m = +$
- $i = \boxed{\leftarrow}$

Es fan els canvis de signe per evitar problemes a l'hora de validar el funcionament de l'expressió regular a l'hora d'implementar-la en qualsevol evaluador. El símbol '+' en les expressions regulars és una restricció de l'estrella de Kleen i representa l'aparició de com a mínim 1 cop el símbol anterior (no té un màxim establert).

```

1 (i|p|m(p|i)|mm(i|p)|mmm(i|p)|mmmmm*i|mmmmm*pp*i|mmmmm*pp*m(mm*pp*m)*(i|mm*i|mm*pp*i)|
  mmmm*pp*m(mm*pp*m)*pp*i|mmmmm*pp*m(mm*pp*m)*pp*m(mm*pp*m)*(i|mm*(i|pp*i))|
  mmmm*pp*m(mm*pp*m)*pp*m(mm*pp*m)*p(p|mmm*p)*(i|mi|mmm*i|mpp*i)|mmmmm*pp*m(mm
  *pp*m)*pp*m(mm*pp*m)*p(p|mmm*p)*mpp*m(pp*m)*(i|pp*i)|mmmmm*pp*m(mm*pp*m)*pp*m(
  mm*pp*m)*p(p|mmm*p)*mpp*m(pp*m)*m(pp*m(pp*m)*m)*(i|pp*i|pp*m(pp*m)*(i|pp*i))|mmmmm*
  pp*m(mm*pp*m)*pp*m(mm*pp*m)*p(p|mmm*p)*mpp*m(pp*m)*m(pp*m(pp*m)*m)*m(pp*m(pp*m)
  *m(pp*m(pp*m)*m)*m*(i|pp*i|pp*m(pp*m)*(i|pp*i)|pp*m(pp*m)*m(pp*m(pp*m)*m)*(i|pp*i|pp*m(
  pp*m)*(i|pp*i))))*mmmmm*pp*m(mm*pp*m)*pp*m(mm*pp*m)*p(p|mmm*p)*mpp*m(pp*m)*m(pp*
  m(pp*m)*m(pp*m(pp*m)*m(pp*m(pp*m)*m)*m*(i|m|p)*

```

Podem dividir, a grans termes, l'expressió regular en dues parts. La primera part és la part que ens controla tots els loops possibles que ens podem trobar en un mot.

Primera part.

En aquest tros d'expressió regular podem veure com controla els loops que se'ns poden produir als estats q0,q1,q2,q3 i q4.

```

1 (i|p|m(p|i)|mm(i|p)|mmm(i|p)|mmmmm*i

```

Aquí tenim el patró fins a l'estat q5 amb un retorn a l'inici per un 'i'.

```

1 mmmm*pp*i

```

En aquesta part es contemplen les possibilitats que hi ha de crear loops amb els estats q4, q5, q6, tenint en compte els salts a inici produïts per l'input d'un símbol 'i' en cadascun d'aquests estats.

```

1 mmmm*pp*m(mm*pp*m)*(i|mm*i|mm*pp*i)

```

Ara ja tenim una cel·la aïllada i controlem que després ens vingui una altre 'i'. En aquest punt ens trobem a l'estat q7.

```

1 mmmm*pp*m(mm*pp*m)*pp*i

```

En aquesta part es contemplen les possibilitats que hi ha de crear loops amb els estats q7, q8, q9, tenint en compte els salts a inici produïts per l'input d'un símbol 'i' en cadascun d'aquests estats.

```

1 mmmm*pp*m(mm*pp*m)*pp*m(mm*pp*m)*(i|mm*(i|pp*i))

```

En aquesta part es contemplen les possibilitats que hi ha de crear loops amb els estats q10, q11, q12, tenint en compte els salts a inici produïts per l'input d'un símbol 'i' en cadascun d'aquests estats. En el cas de mpp*i saltaria des de l'estat q13.

```

1 mmmm*pp*m(mm*pp*m)*pp*m(mm*pp*m)*p(p|mmm*p)*(i|mi|mmm*i|mpp*i)

```

En aquesta part es contemplen les possibilitats que hi ha de crear loops amb els estats q13, q14, tenint en compte els salts a inici produïts per l'input d'un símbol 'i' en cadascun d'aquests estats.

```

1 mmmm*pp*m(mm*pp*m)*pp*m(mm*pp*m)*p(p|mmm*p)*mpp*m(pp*m)*(i|pp*i)

```

Finalment entrem amb els últims 4 símbols '+' . En aquesta part es contemplen les possibilitats que hi ha de crear loops amb els estats q13, q14, q15, tenint en compte els salts a inici produïts per l'input d'un símbol 'i' en cadascun d'aquests estats.

```

1 mmmm*pp*m(mm*pp*m)*pp*m(mm*pp*m)*p(p|mmm*p)*mpp*m(pp*m)*m(pp*m(pp*m)*m)*(i|pp*i|pp
  *m(pp*m)*(i|pp*i))

```

En aquesta part es contemplen les possibilitats que hi ha de crear loops amb els estats q14, q15, q16, tenint en compte els salts a inici produïts per l'input d'un símbol 'i' en cadascun d'aquests estats (inclòs l'estat q13).

```

1 mmmmm*pp*m(mm*pp*m)*pp*m(mm*pp*m)*p(p|mmm*p)*mpp*m(pp*m)*m(pp*m(pp*m)*m)*m(pp*m(
  pp*m)*m(pp*m(pp*m)*m)*m)*(i|pp*i|pp*m(pp*m)*(i|pp*i)|pp*m(pp*m)*m(pp*m(pp*m)*m)*(i|pp*i|
  pp*m(pp*m)*(i|pp*i))))

```

La segona part on contolem que es trobi tot el patró fins a l'últim estat.

Segona part

Podem veure la part de l'expressió regular que contempla que es compleixi sencer fins arribar a l'últim estat.

```

1 *mmmmmm*pp*m(mm*pp*m)*pp*m(mm*pp*m)*p(p|mmm*p)*mpp*m(pp*m)*m(pp*m(pp*m)*m)*m(pp*m(
  pp*m)*m(pp*m(pp*m)*m)*m)*m(i|m|p)*

```

2.9 Regular Grammar

Les gramàtiques regulars son una especificació de les gramàtiques lliures de context. En el llibre no se'n parla de gramàtiques regulars, es parla sobre gramàtiques lliures de context (*CFG*).

Així doncs una gramàtica regular és capaç de generar tots els mots d'un llenguatge regular. Al dir tots es parla del conjunt infinit de mots finits possibles que $FXBP_{DFA}$ pot acceptar. En aquest cas utilitzem el *programari referenciat a [6]* per extreure la gramàtica regular sobre l'autòmata $FXBP_{MDFA}$.

Quan dues gramàtiques regulars generen els mateixos mots es diu que son equivalents.

2.9.1 $FXBP_{rg}$ (generada amb jflap).

Representem ϵ amb el caràcter 3.

```

1 S -> iS|pS|mA
2 A -> iS|mB|pS
3 B -> pS|iS|mC
4 C -> iS|mD|pS
5 D -> iS|pE|mD
6 E -> iS|mF|pE
7 F -> iS|mD|pG
8 G -> mI|iS|pG
9 H -> pG|iS|mH
10 I -> pM|mH|iS
11 J -> mK|pL|iS
12 K -> iS|pM|mK
13 L -> mN|pL|iS
14 M -> mJ|iS|pM
15 N -> mO|iS|pL
16 O -> iS|mP|pL
17 P -> mQ|pL|iS
18 Q -> 3|pQ|mQ|iQ

```

Una gramàtica regular està compresa dins de les gramàtiques lliures de context i es pot simplificar, ja que pot presentar ambigüetats *Definition 2.7, referència [1]*. S'entén ambigüetat en una *CFG* com l'existència de 2 maneres de generar un únic mot, llavors es diu que el mot és derivat de manera ambigua.

Determinar si una gramàtica és ambigua comporta l'existència d'un mot generat per més d'una de les possibles branques de l'arbre de parsing. Per poder determinar que no hi ha dues branques possibles per un determinat mot es treballen les *CFG* cercant la seva forma normal de chomsky.

2.9.2 Chomsky Normal Form

S'utilitzen les CNF (Chomsky Normal Form), *Definition 2.8 de la pàgina 107 de la referència [1]*, per representar les gramàtiques regulars de manera simplificada i sense ambigüetats. Utilitzem l'eina web [7] per passar la nostra CFG a CNF.

```
1 S-> RS|US|TA
2 A-> RS|TB|US
3 B-> US|TC|RS
4 C-> US|RS|TD
5 D-> RS|UE|TD
6 E-> RS|TF|UE
7 F-> RS|TD|UG
8 G-> TI|RS|UG
9 H-> UG|RS|TH
10 I-> UM|TH|RS
11 J-> TK|UL|RS
12 K-> RS|UM|TK
13 L-> TN|UL|RS
14 M-> TJ|RS|UM
15 N-> TO|RS|UL
16 O-> UL|RS|TP
17 P-> TQ|UL|RS|m
18 Q-> UQ|TQ|RQ|p|m|i
19 R-> i
20 T-> m
21 U-> p
```

En aquest format la CNF està en la seva mínima expressió i no conté ambigüetats.

2.10 Referència al codi

El codi de l'autòmata finit determinista està a la classe DFA representada per els fitxers DFA.h i DFA.cpp a l'arrel del directori de la pràctica.

2.11 Reconeixibilitat i Decidibilitat

Tot llenguatge reconegut per un autòmata finit determinista o indeterminista forma part dels llenguatges lliures de context i per definició **son reconeixibles per PDA i per TM i decidibles**.

Chapter 3: Push-Down Automata

3.1 Definició del context

Sigui Φ_{PDA} l'espai de funcionament lògic del nostre autòmata com l'espai estipulat a la *Secció 3.Patrons[2]*. Per tant acabem d'aquirir totes les definicions assumides a l'enunciat del problema, *veure [2]*.

Fem les següents afirmacions sobre Υ_{PDA} :

- $\omega_{PDA} \in P_0 \wedge P_0 \in P$

Fem les següents afirmacions sobre Ψ_{PDA} :

- retorna True quan GGR_{PDA} Accepta, és a dir quan $\omega_{PDA} \in PIG$.
- retorna False altrament.

on : PIG és el conjunt de patrons que compleixen la propietat de patró implícit global *definida a [2] apartat 3.3.2*.

3.2 Anàlisi per el disseny

PIG requereix un nombre de cel·les actives (+) estrictament superior al doble de les negatives (-) :

$$|+| > 2 * |-|$$

Necesitem una manera de contar tant el nombre de cel·les actives com les inactives. Per aixó ens servirem de la pila. Utilitzant el lema del bombeig podem detectar si $L(GGR_{PDA}) \notin RL$ on RL son el conjunt dels llenguatges regulars, *Theorem 1.70 [1]*, Deduïm per tant que $L(GGR_{PDA})$ no és un llenguatge regular i que necessitem un autòmata més potent per reconèixer mots de $L(GGR_{PDA})$.

3.3 Definició informal de l'autòmata

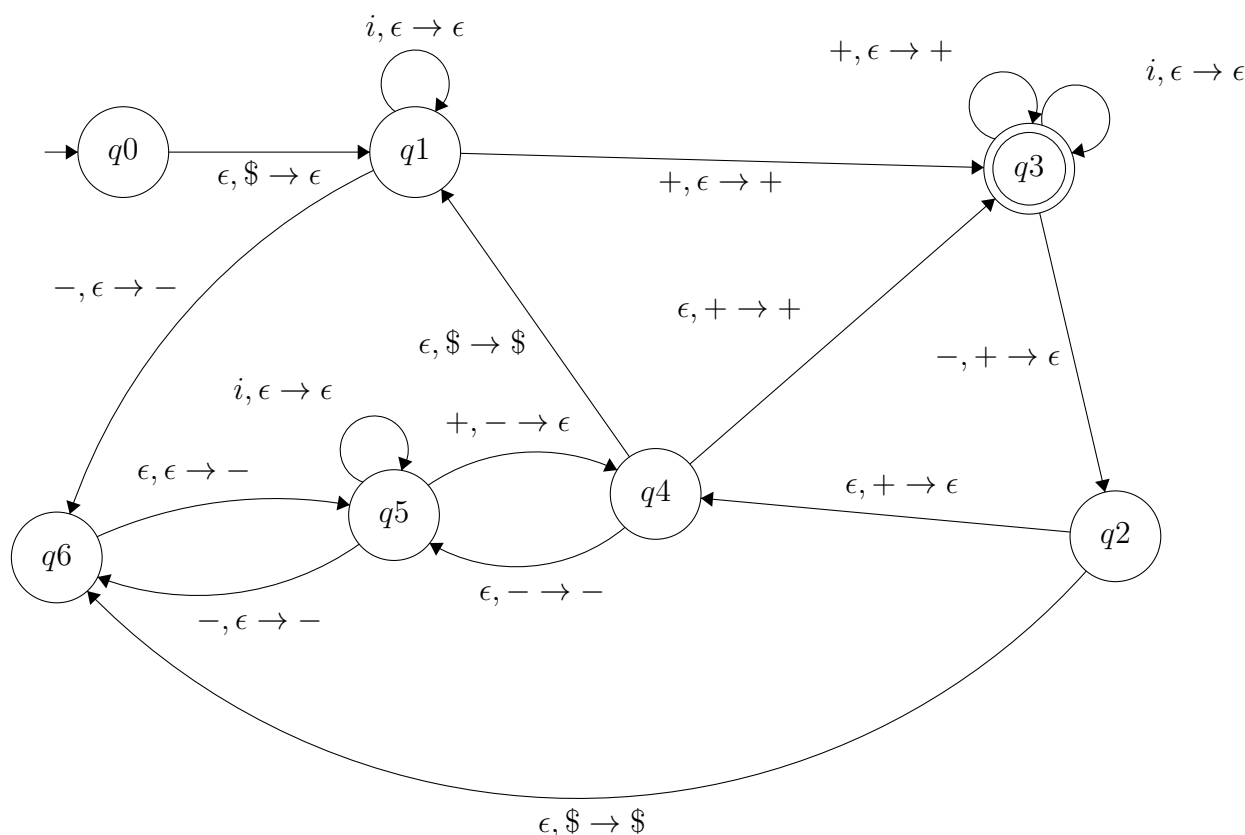
Primerament cal observar possibles propietats del problema que ens puguin servir per a la construcció de l'autòmata. Per poder contar correctament que hi ha estrictament el doble de positius que negatius GGR_{PDA} fara servir les següents regles:

- Les cel·les actives es contabilitzaran 1 sol cop a la pila.
- Les cel·les inactives es contabilitzaran 2 cops a la pila.
- Una cel·la activa pot cancel·lar 1 inactiva a la pila.

- Una cel·la inactiva pot cancel·lar 2 actives a la pila
- La prioritat de cada cel·la sera cancel·lar les de simbol contrari que hi hagi a la pila.
- Si no hi ha cel·les del simbol contrari a la pila, s'empilaran.
- A la pila només hi podrà haber 1 tipus de simbol simultaniament (apart del simbol de control de pila buida \$).
- Quan el mot hagi estat llegit s'acceptara només si hi ha 1 o més cel·les actives a la pila.

Realitzem un mètode constructiu per treure el diagrama d'estats i transicions que representa l'autòmata.

3.4 Diagrama GGR_{PDA}



3.5 Definició formal de GGR_{PDA}

$$GGR_{PDA} = \{ Q, \Sigma^*, \Gamma^*, \delta, q_0, q_3 \}$$

- $Q = \{ q_0, q_1, q_2, q_3, q_4, q_5, q_6 \}$
- $\Sigma^* = \{ +, -, \boxed{\leftarrow}, \epsilon \}$
- $\Gamma^* = \{ +, -, \$, \epsilon \}$
- $\delta = Q \times \Sigma^* \times \Gamma^* \rightarrow Q$
- $q_0 \in Q$
- $q_6 \subseteq Q$

3.5.1 Taula δ GGR_{PDA}

Table 3.1: δ table for + and - inputs								
input	+				-			
stack	+	-	\$	ϵ	+	-	\$	ϵ
->Q0								
Q1				$\{(Q3,+)\}$				$\{(Q6,-)\}$
Q2								
Q3				$\{(Q3,+)\}$	$\{(Q2,\epsilon)\}$			
Q4								
Q5		$\{(Q4,\epsilon)\}$						$\{(Q6,-)\}$
Q6								

Table 3.2: δ table for ϵ and \$ inputs								
input	ϵ				\$			
stack	+	-	\$	ϵ	+	-	\$	ϵ
->Q0								$\{Q1,\$ \}$
Q1				$\{(Q1,\epsilon)\}$				
Q2					$\{(Q4,\epsilon)\}$		$\{(Q6,\$)\}$	
Q3				$\{(Q3,\epsilon)\}$				
Q4					$\{(Q3,+)\}$	$\{(Q5,-)\}$	$\{(Q1,\$)\}$	
Q5				$\{(Q5,\epsilon)\}$				
Q6								$\{(Q5,-)\}$

3.6 Equivalència entre PDA i CFG

Els PDA i el CFG son equivalents a nivell de potència.

Com diu al *Theorem 2.20 de la referència [1]*, un llenguatge és lliure de context si i només si un autòmata de Pila és capaç de reconeixè'l i ho demostra al *Lema 2.21 i al 2.27 [1]*.

Afirmem doncs, que poguent representar l'autòmata GGR_{PDA} com un Push-Down autòmata existeix una CFG capaç de generar tots els mots de GGR_{PDA} .

3.7 Gramàtica lliure de context

Veure *Formal Definition de CFG, Definition 2.2, pàg 104 [1]*.

Denotem la construcció dels possibles mots com la obligació d'afegir sempre un símbol + a un ω_{PDA} que tingui com a mínim el doble de símbols '+' que de '-'.
Expressem $GGRCFG_{PDA} = (V, T, P, S)$

- $V = S, A, B$
- $T = +, -, i$
- $P = S \rightarrow A, A \rightarrow B+A \mid B+B, B \rightarrow -B+B+ \mid +B-B+ \mid +B+B- \mid BiB \mid \epsilon$

Aquesta gramàtica és capaç de generar tots els mots del $L(GGR_{PDA})$.

3.8 Ambigüitat

Una gramàtica lliure de context pot donar pas a diferents generacions d'un mateix mot, *Definition 2.7, pàg 108 [1]*.

Podem detectar ambigüitat amb el mot ω_{PDA} on $\omega_{PDA} = '++-+'$, ens ajudem de *l'eina sobre autòmats de la pàgina web d'standford [8]* per demostrar-la.

Representem l'ambigüitat mitjançant les següents taules :

Table 3.3: Ambigüitat Figura 1

RULE	APPLICATION	RESULT
Start \rightarrow S	Start	S
S \rightarrow A	S	A
A \rightarrow B+B	A	B+B
B $\rightarrow \epsilon$	B+B	+B
B \rightarrow +B-B+	+B	++B-B+
B $\rightarrow \epsilon$	++B-B+	++B-B+
B $\rightarrow \epsilon$	++-B+	++-+

Table 3.4: Ambigüitat Figura 2

RULE	APPLICATION	RESULT
Start \rightarrow S	Start	S
S \rightarrow A	S	A
A \rightarrow B+B	A	B+B
B \rightarrow +B-B+	B+B	+B+B-+B
B $\rightarrow \epsilon$	+B+B-+B	++B-+B
B $\rightarrow \epsilon$	++B-+B+	++-+B
B $\rightarrow \epsilon$	++-+B	++-+

3.9 Chomsky Normal Form

Veure Definició 2.8, pàg 109 [1].

"Qualsevol llenguatge lliure de context és generat per una gramàtica lliure de context en forma normal de chomsky" - Theorem 2.9 - Michael Sipser.

Per tant per generar la teotalitat el $L(GGR_{PDA})$ ens n'asseguem passant la GGR_{CFGPDA} a CNF . Emprem el programa *JFLAP [6]* que segueix els passos descrits a *Exemple 2.10, pàg 110 [1]*.

- 1 S \rightarrow BF|EB|EA|BO|+|BE
- 2 B \rightarrow CB|i|BC|DL|DK|DJ|EI|EH|EG|EW|EV|EU|BT|DR|EQ|EP
- 3 F \rightarrow EA
- 4 E \rightarrow +
- 5 A \rightarrow EB|+|BE|EA|BO|BF
- 6 O \rightarrow EB
- 7 C \rightarrow i
- 8 D \rightarrow -
- 9 L \rightarrow EE
- 10 K \rightarrow EN
- 11 N \rightarrow BE
- 12 J \rightarrow BL

```
13 I -> DE
14 H -> DN
15 G -> BI
16 W -> ED
17 V -> EM
18 M -> BD
19 U -> BW
20 T -> CB
21 R -> BK
22 Q -> BH
23 P -> BV
```

3.10 Referència al codi

El codi del Push-Down Automata és a la classe PDA representada per els fitxers PDA.h i PDA.cpp a l'arrel del directori de la pràctica.

3.11 Reconeixibilitat i Decidibilitat

Tot llenguatge reconegut per un push-down autòmata determinista o indeterminista forma part dels llenguatges lliures de context i per definició **son reconeixibles per TM i decidibles**.

Chapter 4: Turing Machine

4.1 Definició del Context

Sigui Φ_{TM} l'espai de funcionament lògic de la nostra màquina de Turing com l'espai estipulat a la *Secció 3. Patrons* [2]. Per tant acabem d'adquirir totes les definicions assumides a l'enunciat del problema, veure [2].

Fem les següents afirmacions sobre Υ_{TM} :

- $\omega \in L^2$

Fem les següents afirmacions sobre Ψ_{TM} :

- retorna -1 quan QReject
- retorna t quan QAccept
- loop altrament

on : L^2 definit a l'apartat 4. *Llenguatge Formal de l'enunciat del problema* [2].

4.2 Disseny descendent i pseudocodi

Donat el problema presentat a l'*exercici 5.3 que parla de Llenguatge Recursivament Enumerable*, [2]. Proposem el següent disseny descendent explicat també en pseudocodi per resoldre el problema PE on Pe son els patrons explícits comentats a l'*enunciat del problema* [2] apartat 3.4.

- Llegim patterns
- return eval(patterns)

On patterns està compost dels patrons P1 i P2, i eval serà l'algoritme encarregat d'intentar decidir l'acceptació o rebuig de si P2 és un patró generacional fruit d'una evolució de P1 retornant el tick en el que P1 deriva a P2, **sempre que el pugui**.

Es pretén treballar amb patrons de manera genèrica, la millor manera de poder comparar si un patró és el mateix que un altre, és mitjançant la seva representació mínima. Per tant actuem sobre l'entrada per realitzar la següent operació sobre patrons :

4.2.1 Minimització

- PRE : Donat $P_{y,x}$ on $x \geq 0$
- POST : Retorna $P_{y,0}$

Les minimitzacions contemplan la possibilitat que un patró estigui embolcallat dins de files amb cèl·lules inactives o columnes de cèl·lules inactives. *Veiem apartat 3.1 Definició de patrons [2]*. La idea és extreure les files sobrants de cèl·lules inactives de la part superior i inferior del patró, transposar el patró, aplicar la extracció de les cel·les inactives de la part superior i inferior altre cop i tornar a transposar el patró retornant així la seva representació mínima.

- Input P1
- MinimizeByRows(P1)
- Transpose(P1)
- MinimizeByRows(P1)
- Return Transpose(P1)

4.2.2 Minimitzar per files

La idea general és extreure les files plenes de cel·les inactives tant a principi com a final de patró. Comencem amb el patró inicial, només ens podem trobar cel·les inactives a les primeres files. Les regles son :

1. Ignorar les files que no continguin cel·les actives.
2. Quan trobem primera activa ho notifiquem i desem la fila en el patró resultant
3. per cada fila
 - (a) Si no n'hi ha, acumulem fila en patró temporal
 - (b) si hi ha cel·la activa, persistim acumulat en patró resultant

Per contra ignorarem al final totes les files acumulades que no tinguin cel·la activa.

4.2.3 Transposició

Operació matricial de transposició. *Veiem Referència bibliogràfica [5]*.

4.2.4 Algoritme eval

L'objectiu és anar comptant el nombre de ticks que trigarà P1 en evolucionar a P2, per tant s'haurà de pensar l'evolució de P1 fent l'algoritme performTick que ens permeti crear el nou patró a partir de l'anterior.

A cada tick, el que fem és mirar cel·la a cel·la si P1 és igual en totes les posicions a P2. Per mirar això, fem un recorregut per les dues matrius alhora comparant les posicions entre elles.

- tickCounter = 0
- While P1 != P2

- P1 = performTick(P1)
- tickCounter++;
- fi while
- Return tickCounter

4.2.5 Algoritme performTick

Hem de preveure que el patró P1 pot créixer, per tant el patró resultant pot ser més gran que P1. Si afirmem que P1 creix en nombre de files i columnes cap a P1', podem dir que existeix un patró $P1' > P1$ on :

$$P1' > P1 = (|files(P1')| > |files(P1)|) \wedge (|columnes(P1')| > |columnes(P1)|)$$

Per tant, si el patró P1 evoluciona fóra dels límits de P1, P1' estarà preparat per poder representar-lo, en cas que no evolucioni fóra dels límits de la representació de P1, la minimització del mateix prèvia a la comparació farà la feina de representar-lo de manera inferior en nombre de files i columnes si escau.

Per cada cel·la de P1' s'hi posa el resultat de la funció newCellValue que aplica les regles d'evolució per cada cel·la.

4.2.6 Algoritme newCellValue

Compararà donada una fila i una columna, les cel·les adjacents a aquesta fila i columna per determinar les característiques a seguir de l'evolució, explicades a l'*apartat 3.4 Model, Enunciat de la pràctica [2]*.

Aquest algoritme fa un recorregut **constant** per les 8 posicions on hi h pot haver un possible canvi. Concretament mira els veïns de la cel·la tractada i determina en funció de les normes especificades l'*apartat 3.4 Model, Enunciat de la pràctica [2]* si la cel·la es manté en el seu estat o si reb un canvi.

4.2.7 Referència al codi

El codi de la Màquina de Turing es troba al directori / del directori de la pràctica, concretament en formen part :

- classe TM (TM.h i TM.cpp): Representa la màquina de turing.
- classe MatrixPattern (MatrixPattern.h i MatrixPattern.cpp): Representa la manera de tractar els patrons dins de la màquina de turing.
- classe Utils (Utils.h i Utils.cpp): Eines útils per al tractament global de la mateixa màquina de turing i per el main del programa.

4.3 Reconeixibilitat del problema

Definim MSP_{TM0} com la primera versió de la MSP_{TM1} , on la primera versió és la màquina Turing Reconeixible del llenguatge L^2 .

MSPTM accepta tots els mots que pertanyen al llenguatge L^2 descrit a l'*apartat 5.3 Llenguatge Recursivament Enumerable, [2]*. Per construcció demostrem que el problema és Turing Reconeixible presentant un programa en C++.

Si un mot w pertany a L^2 llavors MSP_{TM0} el reconeix, altrament loop.

En aquesta primera versió de la màquina MSP_{TM0} ens centrem en la reconeixença del llenguatge i per tant només es contempla l'acceptació determinant que la màquina entrarà en un Q_{Accept} quan es compleixi la igualtat en l'evolució. Altrament no podem determinar que la màquina (en aquesta primera versió) entri mai en un Q_{Reject} .

Exemple d'execució

Input : amb P1 i P2

```

1  ++++++
2  +-----+
3  +-+++++-+
4  +-+-----+
5  +-+----+
6  +-+----+
7  +-+----+
8  +-+----+
9  +-+-----+
10 +-+++++-+
11 +-+-----+
12 ++++++
13
14 -----+-----
15 -----+-----
16 -----+-----
17 -----
18 -----+-----
19 -----+-----
20 +++-+-+-----+
21 +++-+-+-----+
22 +++-+-+-----+
23 +++-+-+-----+
24 +++-+-+-----+
25 +++-+-+-----+
26 -----+-----
27 -----+-----
28 -----
29 -----+-----
30 -----+-----
31 -----+-----

```

Output :

```

1  t = 3

```

Concluïm que :

$P2 = P1^3$

4.4 Decidibilitat del problema

La decidibilitat planteja el món dels problemes que presenten complexitats finites en la seva resolució. En aquest cas proposem diverses modificacions per **acostar** la implementació de $MSP_{TM0} \rightarrow MSP_{TM}$ on MSP_{TM} reconeix un conjunt més ampli de patrons.

En el nostre escenari el problema sabem que és reconeixible, però encara no podem afirmar que el problema sigui, o no, decidible, per tant anirem treballant sobre MSP_{TM0} aplicant modificacions per intentar portar el problema al món dels decidibles. Per fer-ho estudiarem en les següents seccions els patrons i els seus comportaments.

4.4.1 Comportaments de l'evolució de patrons

Comentem les següents propietats detectades en el comportament de l'evolució de patrons.

- Derivació
- Estabilització
 - Extinció
- Oscil·lació
- Creixement

4.4.2 Derivació

Per parlar de la decidibilitat del problema el primer cas que ens apareix és que el problema sigui turing reconeixible tal com s'ha demostrat a l'apartat [Reconeixibilitat del problema]. Llavors la derivació serà el que en decidibilitat es diu acceptació. Quan P1 evoluciona amb $t = n$ a P2, llavors ω pertany al llenguatge $L(MSP_{TM})$ i la màquina accepta.

4.4.3 Estabilització

MSP_{TM0} no detecta quan $P2! = P1^t$, és a dir, no detecta cap cas en el que es pugui afirmar que un patró P1 no tindrà una evolució a P2. L'estabilització d'un patró s'enten com :

$$P1^t = P1^{t+1}$$

En aquest cas direm que el patró s'ha estabilitzat per que :

$$\nexists P1^{t+n!} = P1^t$$

on : $n > 0$

I per tant en aquest cas es pot afirmar que P1 no evolucionarà mai fins P2 si no ho ha fet en $P1^t$.

Modifiquem $MSP_{TM0} \rightarrow MSP_{TM1}$ per què rebutgi (*QReject*) aquest cas. Amb aquesta acció podem descartar patrons com a candidats de formar part de L^2 . S'ha acotat el conjunt de ω possibles a formar part de L^2 .

Exemple d'execució

Input : amb P1 i P2

```
1
2 ++++++
3 +------+
4 +-+++++-+
5 +-+-----+-+
6 +-+----+--+
7 +-+----+--+
8 +-+----+--+
9 +-+----+--+
10 +-+-----+-+
11 +-+++++-+
12 +------+
13 ++++++
```

```

14
15 -----+++++-----
16 -----+++++-----
17 -----+++++-----

```

Output :

t = -1

Concluïm que :

$$P2! = P1^8 \wedge P1^7 = P1^8$$

Extinció

L'extinció d'un patró és un cas particular de l'estabilització. Un patró es pot extingit, i un cop extingit no pot tornar a aparèixer cap cel·la activa. Si $P1^y$ on y fa P1 extingit llavors :

$$\nexists P1^n \{n | n \in \mathbb{N} \wedge n > y\}$$

Veïem per la següent entrada :

```

1 -----
2 -+-----+-
3 -----
4
5 --+-
6 --+-

```

La següent sortida :

t = -1

conclusió

P1 s'extingeix per t=1, P1² s'estabilitza.

Veïem un altre exemple d'execució :

```

1 -----
2 -+-----+-
3 -----
4
5 ----
6 ----

```

La següent sortida

t = 1

Conclusió

Fent l'anterior entrada estem detectant si P1 tendirà a la seva extinció.

4.4.4 Oscil·lació

El problema de l'oscil·lació comporta un repte més gran que els que ens hem plantejat fins ara, Detectem que un patró pot ser que oscili, és a dir que :

$$P1^1 = P1^3 = P1^5 = P1^n \{n | n \in \mathbb{N} \% 2 \neq 0 \wedge n > 5\}$$

L'expressió anterior no està acotada a un finit de possibilitats. n és infinit. Per tant podria quedar oscil·lant amb la conseqüència que en termes de decidibilitat el problema entrés en loop. El repte de detectar aquest fet és plantejar-se la comparació en aquest cas amb una finestra de 3 patrons, i la comparació de $P1^{t+1}$ amb $P1^t$ i $P1^{t-1}$ on $t > 0$. D'aquesta manera, la oscil·lació amb una finestra de 3 és decidible, presentem la màquina MSP_{TM} com la màquina capaç de reconèixer aquestes oscil·lacions.

Veïem per la següent entrada :

```

1  ++++++
2  +-----+
3  +-+++++-+
4  +-+-----+-+
5  +-+---+---+-+
6  +-+---+---+-+
7  +-+---+---+-+
8  +-+---+---+-+
9  +-+++++-+
10 +-----+
11 ++++++
12
13 ++++++
14 +-----+
15 +-+++++-+
16 +-+-----+-+
17 +-+---+---+-+
18 +-+---+---+-+
19 +-+---+---+-+
20 +-+---+---+-+
21 +-+++++-+
22 +-----+
23 ++++++

```

La següent sortida

t = -1

Conclusió

El patró oscil·la amb finestra de 3 per t = 481.

El problema de la oscil·lació és que en aquest cas ha entrat en joc una altre variable, la variable finestra. I és que si n és la finestra i n és infinit, llavors, totes les possibilitats en la oscil·lació també ho són.

En aquest escenari $n \rightarrow \infty$, referenciant-nos al problema de l'aturada explicat a 4.2. *Halting problem*, [1], veiem que requerim de un problema indecidible per poder decidir la resposta del nostre problema. Necessitem saber abans de provar totes les possibilitats infinites de n si donat un mot ω la MSP_{TM} no queda en loop, per poder afirmar que el problema representat amb la màquina MSP_{TM} és decidable. **Aquest fet fa que el problema de detectar la oscil·lació per $n \rightarrow \infty$, sigui indecidible.**

Veiem per la següent entrada :

```

1  -+-
2  -++
3  +-+
4
5  ---+
6  ---

```

silence to the eternity...

Conclusió

Si estudiem el cas de manera detallada ens trobem que l'evolució d'aquest patró segueix la següent evolució:

$P1^0$

```

1  -+-
2  -++
3  +-+

```

$P1^1$

```

1  -++
2  +-+

```

3 --+

$P1^2$

1 ++-

2 -++

3 +- -

$P1^3$

1 +++

2 --+

3 +- -

$P1^4$

1 --+-

2 -++

3 +- +

4.4.5 Creixement

El creixement és indeterminat, és a dir, detectar que un patró sempre creix és afirmar que per tots els $t > 0$ $P1^t < P1^{t+1}$ *ont* $\rightarrow \infty$. Aquest cas no es pot determinar sense altre cop la màquina que reconeix el llenguatge del 4.2. *Halting problem*, [1]. Per tant no podem determinar que un patró deriva a un altre en un t si tenim ∞ t per mirar, afirmant que l'evolució de P1 sempre creix.

Si un patró sempre creix, podríem donar una resposta negativa (modificant MSP_{TM}), el problema és determinar que sempre creix.

Chapter 5: Continguts de l'entrega

- Codi
- Exemples L^2
- Sortides
- Documentació

5.1 Codi

El codi del programa es troba al directori arrel repartit en fitxers .h i .cpp. Es disposa de 4 parts diferenciades.

- DFA : fitxers DFA.h/.cpp i Alphabet.h/.cpp
- PDA : fitxers PDA.h/.cpp i Alphabet.h/.cpp
- TM : fitxers TM.h/.cpp, MatrixPattern.h/.cpp
- Programa principal : main.h/.cpp i Utils.h/.cpp (i tots els anteriors)

La classe DFA conté el codi per l'autòmata finit determinista, la classe PDA conté el codi per l'autòmata de pila i la classe TM i MatrixPattern són els encarregats de gestionar el programa en C que representa la Màquina de Turing.

La classe Main i la classe Utils són complementaris per poder generar la fusió i que tot encaixi sense problema. La classe Alphabet conté les enumeracions utilitzades per el PDA i el DFA.

5.2 Exemples L^2

El directori data conté exemples de L^2 que poden ser executats a la fusió, és a dir al programa principal anomenat main.

5.3 Sortides

Utilitzant el paràmetre -d es pot entrar en mode Déu. El mode Déu permet rebre fitxers que continguin ω de L^2 , i en algun dels casos permet generar-los també. En cas que es vulguin generar, es pot escollir la sortida en format csv on s'hi representen, el nombre de files i columnes de P1 on P1 és el patró que es farà evolucionar, els clocks de la màquina que triga en realitzar l'acció i el nombre de ticks que ha durat l'execució.

Adicionalment es demana informació sobre la generació que també s'escriu a la sortida.

Totes les sortides generades per nosaltres es poden trobar dins del directori logs del directori de l'entrega.

5.4 Documentació

Els fitxers per generar la documentació en pdf estan dins del directori documentation. L'informe consta d'un sol fitxer en format pdf que conté totes les descripcions i les conclusions a les que hem arribat situat al directori arrel de l'entrega.

Bibliography

- [1] Fundamental of Computation - Michael Sipser
- [2] Problema Patterns (i.e., Cerca de Patrons) - Jaume Rigau
- [3] Lexical analisys - part 4
- [4] Universal Turing Machine - Manual
- [5] MatrixPattern Transposition Wikipedia
- [6] JFlap Automaton software
- [7] Generador de formes normals de chomsky
- [8] eina web de la pàgina web d'standford