eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Lengoaia eta Sistema Informatikoak Saila

Bilboko Industria Ingeniaritza Teknikoko Unibertsitate Eskola

Lengoaiak, Konputazioa eta Sistema Adimendunak

Kudeaketaren eta Informazio Sistemen Informatikaren Ingeniaritzako Gradua

2. Maila

2015-16 ikasturtea

3. Gaia: Automata finituak eta lengoaia erregularrak

JOSE GAINTZARAIN IBARMIA

Azken eguneraketa: 2015-9-2

GAIEN AURKIBIDEA

3. AUTOMATA FINITUAK ETA LENGOAIA ERREGULARRAK

Gai honetan automata finituak azalduko ditugu. "Automata finitu" era laburrean esateko eta idazteko AF inizialak erabiliko ditugu. Automata finituak egoeratan oinarritutako makinak dira. Lehenengo gaian ikusi genituen adibideetan, makinen memoria-ahalmena egoeren eta Sarrera/Irteerako gailuen bidez finkatzen zela esan genuen. Lehenengo bi adibideetan memoria egoeren bidez inplementatzen genuen eta hirugarrengoan egoeren eta Sarrera/Irteerako gailuaren bidez, hirugarren adibide horretan emaitza gordez joateko Sarrera/Irteerako gailua erabiltzen baikenuen. Gai honetan landuko ditugun automata finituetan, egoerak erabiliko dira memoria bezala. Egoera-kopurua finitua izango denez, memoria ahalmena ere mugatua izango da eta ondorioz kalkuluak burutzeko gaitasuna ere mugatua izango da.

Automata finitu deterministak (AFD), automata finitu ez deterministak (AFED) eta ε trantsizioak dituzten automata finitu ez deterministak (ε-AFED) bereiztuko ditugu eta hiru automata-mota horiek baliokideak direla ere frogatuko dugu. Gainera, beraien bidez defini daitezkeen lengoaiak, hau da, lengoaia erregularrak ere aurkeztuko ditugu.

Automata finitu bakoitza lengoaia bati lotuta doa. Automatata finitu bati A^*-ko hitz bat ematen zaionean, automatak hitz hori berari lotutako lengoaiakoa al den ala ez erabakiko du eta "Bai" edo "Ez" erantzungo du.

Jarraian automata finituak era zehatzagoan definituko dira.

3.1 Automata Finitu Deterministak (AFDak)

Automata finitu determinista bat boskote bat da (bost osagai dituen egitura bat da):

$$(Q, A, \delta, q_0, Y)$$

Bost osagai horien definizioa honako hau da:

- Q egoeren multzo finitua da: $q_0, q_1, q_2, q_3, \ldots$
- A alfabetoa da
- δ trantsizio funtzio determinista da. Q multzoko egoera bat eta A multzoko sinbolo bat emanda, egoera horretan gaudenean sinbolo hori irakurtzen badugu zein egoeratara joango garen adierazten du δ trantsizio funtzioak. δ funtzioaren mota honako hau da:

$$\delta : Q \times A \to Q$$

- q_0 hasierako egoera da. Hasierako egoera beti bakarra izango da.
- Y osagaia "Bai" erantzuten duten egoerez osatutako multzoa da. Y multzoa Q multzoaren azpimultzoa izango da, hau da, $Y \subseteq Q$.

AFD baten **funtzionamendua** honako hau da: hasieran AFDa q_0 egoeran egongo da eta sarrera bezala A^* multzoko hitz bat jasoko du. Hitz hori osatzen duten sinboloak banan-banan irakurriko ditu. Sinbolo bakoitza irakurtzean zein egoeratara pasatu erabakiko da δ trantsizio funtzioa erabiliz. Beraz q_i egoera batean gaudela A alfabetokoa den α sinboloa irakurtzen badugu eta $\delta(q_i, \alpha) = q_j$ baldin bada, q_j egoerara pasatuko gara. Hitzak beti finituak izango direnez, hitz bat irakurtzeko prozesua ere finitua izango da. Hitza irakurtzea bukatutakoan Q multzokoa den egoeraren batean egongo gara. Egoera hori Y multzokoa ere badenean, automatak "Bai" erantzungo du, hitza automatari lotutako lengoaiakoa dela adieraziz. Baina hitza bukatzean Y-koa ez den egoeraren batean gelditu bagara, automatak "Ez" erantzungo du, hitza automatari lotutako lengoaiakoa ez dela adieraziz.

Gai honetan ikusiko ditugun AFDetan Sarrera/Irteerako gailuan ez da ezer idatziko, bakarrik irakurri egingo da. Gainera irakurgailuak Sarrera/Irteerako gailutik sinbolo bat irakurtzen duen bakoitzean, irakurgailua eskuinera mugituko da posizio bat.

q_j Q multzoko egoera bat izanda eta w A^* multzoko hitz bat izanda, (q_j, w) bikoteari **konfigurazio determinista** deituko diogu. Konfigurazio deterministak $Q \times A^*$ motakoak dira. (q_j, w) erako konfigurazio determinista baten bidez q_j egoeran gaudela eta oraindik w hitza irakurtzeko daukagula adierazten da. Adibidez, A alfabetoa $\{a, b, c\}$ baldin bada, oraingo konfigurazio determinista $(q_3, caaba)$ baldin bada eta trantsizio funtzioak $\delta(q_3, c) = q_6$ betetzen dela adierazten badu, $caaba$ hitzeko lehenengo sinboloa irakurritakoan q_6 egoerara pasatuko gara eta konfigurazio determinista berria $(q_6, aaba)$ izango da. Konfigurazio determinista berrian ikus daitekeen bezala, irakurtzeko gelditzen den hitza laburragoa da orain (irakurri den sinboloa desagertu egin baita).

Notazioa arintzeko, *konfigurazio determinista* idatzi beharrean *konfigurazioa* idatziko dugu automata finitu deterministei buruzko atal honetan zehar.

AFD batek burututako **konputazioa** edo kalkulu-prozesua δ trantsizio funtzioan oinarritzen den δ^* funtzioaren bidez definitzen da. δ^* funtzioaren mota honako hau da:

$$\delta^* : Q \times A^* \rightarrow Q$$

Beraz, q_j egoera bat eta w hitz bat emanda (hau da, konfigurazio bat emanda), δ^* funtzioak egoera bat itzuliko du (q_j egoeratik abiatu eta w hitzeko sinbolo denak irakurri ondoren lortuko den egoera).

δ^* funtzioaren definizio formala honako hau da:

- $\delta^*(q_j, \varepsilon) = q_j$
- $\delta^*(q_j, \alpha w) = \delta^*(\delta(q_j, \alpha), w)$

Definizio horren bidez honako hau adierazten da: irakurtzeko gelditzen den hitza hutsa baldin bada, hau da, ε, orduan egoera berean jarraituko dugu. Irakurtzeko gelditzen den hitza hutsa ez bada, eta bere lehenengo sinboloa A-koa den α sinboloa baldin bada, δ trantsizio funtzioa aplikatuko zaie oraingo q_j egoerari eta α sinboloari eta egoera berri batera pasatuko gara. Egoera berrira pasatu ondoren w-eko gainontzeko sinboloak irakurtzen jarraitu beharko da:

$$\delta^*(\underbrace{\delta(q_j, \alpha)}_{\substack{\text{egoera} \\ \text{berria}}}, w)$$

Adibidez, demagun alfabetoa $A = \{a, b, c\}$ dela, oraingo konfigurazioa (q_3, aac) dela eta $\delta(q_3, a) = q_5$, $\delta(q_5, a) = q_2$ eta $\delta(q_2, c) = q_6$ direla. $\delta^*(q_3, aac)$ konputazioa urratsez urrats honela burutuko litzateke:

- aac hitza hutsa ez denez, $\delta^*(q_3, aac) = \delta^*(\delta(q_3, a), ac)$.
- $\delta(q_3, a) = q_5$ denez, $\delta^*(\delta(q_3, a), ac)$ espresioa $\delta^*(q_5, ac)$ bezala geratuko da.
- ac hitza hutsa ez denez, $\delta^*(q_5, ac) = \delta^*(\delta(q_5, a), c)$.
- $\delta(q_5, a) = q_2$ dela kontuan hartuz, $\delta^*(\delta(q_5, a), c)$ espresiotik $\delta^*(q_2, c)$ espresioa lortuko da.
- c hitza hutsa ez denez, $\delta^*(q_2, c) = \delta^*(q_2, c\varepsilon) = \delta^*(\delta(q_2, c), \varepsilon)$.
- $\delta(q_2, c) = q_6$ denez, $\delta^*(\delta(q_2, c), \varepsilon)$ espresiotik $\delta^*(q_6, \varepsilon)$ espresioa lortuko da.
- ε hitz hutsa geratu zaigunez, $\delta^*(q_6, \varepsilon)$ espresioaren balioa q_6 da.

Beraz, $\delta^*(q_3, aac) = q_6$.

Automata hauek **deterministak** direla esaten da, egoera bakoitzetik sinbolo bakoitzarekin zein egoeratara joan behar den garbi zehaztua dutelako δ funtzioaren bidez. Egoera bakoitzean sinbolo bakoitzeko mugimendu-aukera bakarra da. Automata finitu **ez deterministetan** egoera batzuetan sinbolo batzuekin zer egin behar den ez da adierazten eta kasu batzuetan sinbolo batekin egoera batetik egoera desberdinetara joateko aukera dago.

$D = (Q, A, \delta, q_0, Y)$ **AFD bati lotutako** $L(D)$ **lengoaia** D automatarengandik "Bai" erantzuna jasotzen duten hitzez osatutako lengoaia da. Beste era batera esanda, q_0-tik abiatuta Y multzokoa den egoera batean lagatzen gaituzten A^*-ko hitzez osatutako lengoaia da. Era formalean definizioa honako hau izango litzateke:

$$L(D) = \{w \mid w \in A^* \wedge \delta^*(q_0, w) \in Y\}$$

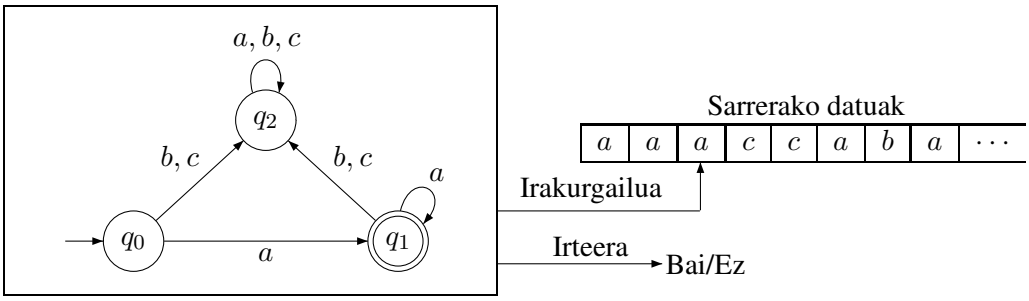
3.1.1 Adibidea: Hutsak ez diren eta bakarrik a sinboloaren errepikapenez eratuta dauden hitzez osatutako lengoaiari dagokion AFDa

Adibide honetako AFDa (3.1 irudia begiratu) 1. gaiko bigarren adibideko makina da. 1. gaian makina hori era informalean aurkeztu genuen baina orain era formalean aurkeztuko dugu gai honetan (3. gaian) emandako AFDen definizioa kontuan hartuz. AFD honek a, b eta c sinboloak eduki ditzakeen hitz bat jasoko du sarrera bezala. AFDak, emandako hitza hutsa ez den eta bakarrik a sinboloaren errepikapenez eratuta dagoen hitza al den erabakiko du. Horrela bada, "Bai" erantzuna itzuliko du eta bestela "Ez" erantzuna itzuliko du. Lengoaiaren definizio formala honako hau da:

$$\{w \mid w \in A^* \wedge |w|_a = |w| \wedge |w| \geq 1\}$$

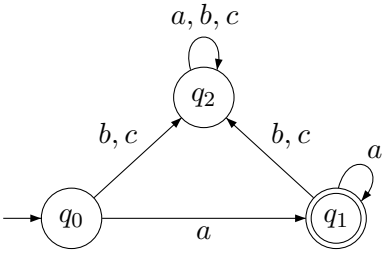
Beraz irteera "Bai" edo "Ez" izango da. AFDa hiru egoerez osatuta dago. q_0 hasierako egoera da. Hitz bat edo karaktere-kate bat irakurtzen hasten garenean q_0 egoeran gaude beraz. Lehenengo karakterea (edo sinboloa) a bada, AFDa q_1 egoerara igaroko da. Lehenengo karakterea b edo c bada, AFDa q_2 egoerara igaroko da. AFDa q_1 egoeran dagoenean, a sinboloa etortzen bada, AFDa egoera berean mantenduko da, hau da, q_1 egoeran. Bestalde, q_1 egoeran egonda, b edo c etortzen bada, q_2 egoerara igaroko da. AFDa q_2 egoeran baldin badago, etortzen dena etortzen dela ere, egoera berean mantenduko da. Karaktere-kate osoa irakurritakoan AFDa q_1 egoeran baldin badago, horrek esan nahiko du karaktere-katea ez dela hutsa eta a sinboloaren errepikapenez osatuta dagoela. Aldiz, karaktere-kate osoa irakurritakoan AFDa q_2 egoeran baldin badago, horrek esan nahiko du karaktere-katea ez dela hutsa eta ez dagoela a sinboloaren errepikapenez bakarrik osatuta. Azkenik, karaktere-kate osoa irakurritakoan AFDa q_0 egoeran baldin badago, horrek esan nahiko du karaktere-katea hutsa dela.

Beraz, q_1 egoeran bukatzen bada, jasotako hitza AFD honi dagokion lengoaiakoa da eta irteerako erantzuna "Bai" izango da, baina q_1 ez den egoeraren batean bukatzen bada, erantzuna "Ez" izango da, jasotako hitza AFD honi dagokion lengoaiako ez dela adieraziz. AFDari dagokion irudian q_1 egoera zirkulu bikoitzarekin agertzen da erantzuna baiezkoa izateko AFDak egoera honetan bukatu beharko duela adierazteko hain zuzen ere. Aipatzekoa da hitz bat jaso edo irakurritakoan q_0 egoeran bukatzeko aukera bakarra hitz hori hutsa izatea dela, hau da, emandako hitza hutsa bada, orduan AFDak q_0 egoeran bukatuko du eta bestela q_1 edo q_2 egoeratan bukatuko du.



3.1 Irudia: Hutsak ez diren eta bakarrik a sinboloaren errepikapenez eratuta dauden hitzei dagokien AFDa.

Gai honetan ikusiko ditugun AFD denek beti "Bai" edo "Ez" erantzuten dutenez eta sarrerako gailutik jasotako karaktere-katean ez dutenez inoiz aldaketarik egiten, AFDak diseinatzerakoan egoerak eta egoeren arteko trantsizioak adierazteko balio duten geziak bakarrik ipiniko ditugu. Beraz, Sarrera/Irteerako gailuak ez ditugu ipiniko. Honela, 3.1 irudiko AFDaren kasuan, 3.2 irudian ikus daitekeena bakarrik ipiniko genuke.



3.2 Irudia: 3.1 irudiko AFDaren grafiko laburtua.

Jarraian AFDaren osagaiak definituko dira boskote eran:

$$(Q, A, \delta, q_0, Y)$$

Osagai bakoitza honela geratuko da:

- $Q = \{q_0, q_1, q_2\}$
- $A = \{a, b, c\}$
- $\delta : Q \times A \to Q$ jarraian datorren taularen bidez definitutako trantsizio funtzioa da:

δ	a	b	c
q_0	q_1	q_2	q_2
q_1	q_1	q_2	q_2
q_2	q_2	q_2	q_2

Taula horren bidez $\delta(q_0, a)$-ren balioa q_1 dela, $\delta(q_0, b)$-ren balioa q_2 dela, eta abar adierazten da.

- q_0 hasierako egoera da. Hasierako egoera beti bakarra izango da.
- $Y = \{q_1\}$. Baietz erantzuten duten egoeren multzoa.

Orain (q_0, aaa) konfigurazioari dagokion konputazioa garatuko dugu urratsez urrats:

1. urratsa: $\delta^*(q_0, aaa) = \delta^*(\delta(q_0, a), aa) = \delta^*(q_1, aa)$

2. urratsa: $\delta^*(q_1, aa) = \delta^*(\delta(q_1, a), a) = \delta^*(q_1, a)$

3. urratsa: $\delta^*(q_1, a) = \delta^*(q_1, a\varepsilon) = \delta^*(\delta(q_1, a), \varepsilon) = \delta^*(q_1, \varepsilon)$

4. urratsa: $\delta^*(q_1, \varepsilon) = q_1$

Konputazioa Y multzokoa den q_1 egoeran amaitu denez, hitza lengoaiakoa da (ez da hutsa eta a sinboloaren errepikapenez eratutako dago).

AFD-en konputazioak konfigurazioz osatutako sekuentziak bezala edo adar bakarreko zuhaitz bezala ere adierazi ohi dira:

$$(q_0, aaa)$$
$$|$$
$$(q_1, aa)$$
$$|$$
$$(q_1, a)$$
$$|$$
$$(q_1, \varepsilon)$$

Jarraian $(q_0, abca)$ konfigurazioari dagokion konputazioa garatuko dugu urratsez urrats:

1. urratsa: $\delta^*(q_0, abca) = \delta^*(\delta(q_0, a), bca) = \delta^*(q_1, bca)$

2. urratsa: $\delta^*(q_1, bca) = \delta^*(\delta(q_1, b), ca) = \delta^*(q_2, ca)$

3. urratsa: $\delta^*(q_2, ca) = \delta^*(\delta(q_2, c), a) = \delta^*(q_2, a)$

4. urratsa: $\delta^*(q_2, a) = \delta^*(q_2, a\varepsilon) = \delta^*(\delta(q_2, a), \varepsilon) = \delta^*(q_2, \varepsilon)$

5. urratsa: $\delta^*(q_2, \varepsilon) = q_2$

Konputazioa Y multzokoa ez den q_2 egoeran amaitu denez, hitza ez da AFD honi dagokion lengoaiakoa.

Konputazio hori konfigurazioz osatutako sekuentzia bezala ere adieraz daiteke:

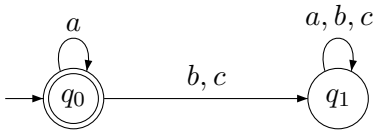
$$(q_0, abca)$$
$$|$$
$$(q_1, bca)$$
$$|$$
$$(q_2, ca)$$
$$|$$
$$(q_2, a)$$
$$|$$
$$(q_2, \varepsilon)$$

3.1.2 Adibidea: b-rik eta c-rik ez duten hitzez osatutako lengoaiari dagokion AFDa

Adibide honetako AFDak (3.3 irudian begiratu) a, b eta c sinboloak izan ditzakeen hitz bat jasoko du sarrera bezala. Emandako hitza b-rik eta c-rik ez duen hitz bat al den erabaki beharko du AFDak. Horrela, b-rik eta c-rik ez badu "Bai" erantzun beharko du eta bestela "Ez" erantzun beharko du. Lengoaiaren definizio formala honako hau da:

$$\{w \mid w \in A^* \wedge |w|_a = |w|\}$$

Adibide honetako AFDak bi egoera ditu. Hasierako egoera q_0 da eta ondorioz, hitz bat irakurtzen hasterakoan q_0 egoeran egongo da. Lehenengo karakterea b edo c baldin bada, AFDa q_1 egoerara igaroko da. AFDa q_1 egoeran baldin badago, etortzen dena etortzen dela ere egoera horretan bertan jarraituko du. Karaktere-katea irakurtzea bukatutakoan AFDa q_0 egoeran badago, badakigu karaktere-kateak ez duela b-rik eta c-rik. Bestalde, karaktere-katea irakurtzea bukatutakoan AFDa q_1 egoeran baldin badago, badakigu karaktere-katean gutxienez b bat edo c bat badagoela. Beraz, q_0 egoeran bukatuz gero, hitza AFD honi dagokion lengoaiakoa izango da eta "Bai" erantzuna itzuliko da, baina q_1 egoeran bukatuz gero, erantzuna "Ez" izango da, irakurritako hitza ez baita AFD honi dagokion lengoaiakoa izango. q_0 egoerak zirkulu bikoitza du erantzuna baiezkoa izateko hor bukatu behar dela adierazteko. Hitz hutsaren kasuan AFDak q_0 egoeran bukatuko du eta ondorioz hitz hutsarentzat ere erantzuna baiezkoa izango da.



3.3 Irudia: b-rik eta c-rik ez duten hitzez osatutako lengoaiari dagokion AFDa.

Jarraian AFDa boskote bezala definituko da.

$$(Q, A, \delta, q_0, Y)$$

Bost osagaiak honako hauek dira:

- $Q = \{q_0, q_1\}$
- $A = \{a, b, c\}$
- $\delta : Q \times A \to Q$ jarraian datorren taularen bidez definitutako trantsizio funtzioa da:

δ	a	b	c
q_0	q_0	q_1	q_1
q_1	q_1	q_1	q_1

  Taularen bidez $\delta(q_0, a)$-ren balioa q_0 dela, $\delta(q_0, b)$-ren balioa q_1 dela, eta abar adierazten da.

- q_0 hasierako egoera da. Hasierako egoera beti bakarra da.
- $Y = \{q_0\}$.

Orain (q_0, aaa) konfigurazioari dagokion konputazioa garatuko dugu urratsez urrats:

1. urratsa: $\delta^*(q_0, aaa) = \delta^*(\delta(q_0, a), aa) = \delta^*(q_0, aa)$

2. urratsa: $\delta^*(q_0, aa) = \delta^*(\delta(q_0, a), a) = \delta^*(q_0, a)$

3. urratsa: $\delta^*(q_0, a) = \delta^*(q_0, a\varepsilon) = \delta^*(\delta(q_0, a), \varepsilon) = \delta^*(q_0, \varepsilon)$

4. urratsa: $\delta^*(q_0, \varepsilon) = q_0$

Konputazioa Y multzokoa den q_0 egoeran amaitu denez, hitza lengoaiakoa da (ez du ez b-rik eta ez c-rik).

AFD-en konputazioak konfigurazioz osatutako sekuentziak bezala edo adar bakarreko zuhaitz bezala ere adieraz daitezke:

$$(q_0, aaa)$$
$$|$$
$$(q_0, aa)$$
$$|$$
$$(q_0, a)$$
$$|$$
$$(q_0, \varepsilon)$$

Jarraian $(q_0, abca)$ konfigurazioari dagokion konputazioa garatuko dugu urratsez urrats:

1. urratsa: $\delta^*(q_0, abca) = \delta^*(\delta(q_0, a), bca) = \delta^*(q_0, bca)$

2. urratsa: $\delta^*(q_0, bca) = \delta^*(\delta(q_0, b), ca) = \delta^*(q_1, ca)$

3. urratsa: $\delta^*(q_1, ca) = \delta^*(\delta(q_1, c), a) = \delta^*(q_1, a)$

4. urratsa: $\delta^*(q_1, a) = \delta^*(q_1, a\varepsilon) = \delta^*(\delta(q_1, a), \varepsilon) = \delta^*(q_1, \varepsilon)$

5. urratsa: $\delta^*(q_1, \varepsilon) = q_1$

Konputazioa Y multzokoa ez den q_1 egoeran amaitu denez, hitza ez da AFD honi dagokion lengoaiakoa.

Konputazio horri dagokion konfigurazio-sekuentzia honako hau da:

$$(q_0, abca)$$
$$|$$
$$(q_0, bca)$$
$$|$$
$$(q_1, ca)$$
$$|$$
$$(q_1, a)$$
$$|$$
$$(q_1, \varepsilon)$$

3.1.3 Adibidea: c-rik ez duten eta a eta b-ren agerpenak orden horretan tartekatuta dituzten hitzez osatutako lengoaiari dagokion AFDa

Adibide honetako AFDak (3.4 irudia begiratu) c-rik ez duten eta a eta b orden horretan tartetakuta dituzten $A = \{a, b, c\}$ alfabetoaren gainean definitutako hitzentzat "Bai" erantzungo du. Beraz AFD honi dagokion lengoaiako hitzetan ezin dira bi a jarraian agertu eta ezta bi b ere. Hitz hutsa lengoaiakoa da eta hutsak ez diren hitzen kasuan lehenengo sinboloak a izan behar du. Hala ere azkeneko sinboloa a edo b izan daiteke. Beraz ε, $abab$ eta aba hitzentzat AFDak "Bai" erantzungo luke, baina $abca$, $bababa$, aaa, cc eta aab hitzentzat "Ez" erantzungo luke. Lengoaiaren definizio formala honako hau da:

$$\{w \mid w \in A^* \wedge \exists k(k \geq 0 \wedge (w = (ab)^k \vee w = ((ab)^k)a))\}$$

AFDak hiru egoera ditu, q_0, q_1 eta q_2. Hasierako egoera q_0 da eta bai q_0 egoeran bukatzen bada eta bai q_1 egoeran bukatzen bada, AFDaren erantzuna "Bai" izango da.

Jarraian AFDa boskote bezala definituko da.

$$(Q, A, \delta, q_0, Y)$$

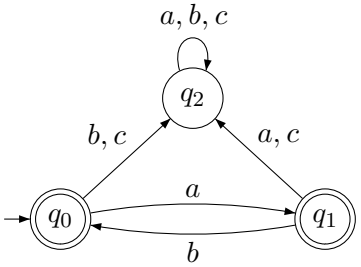
Bost osagaiak honako hauek dira:

- $Q = \{q_0, q_1, q_2\}$
- $A = \{a, b, c\}$
- $\delta : Q \times A \to Q$ jarraian datorren taularen bidez definitutako trantsizio funtzioa da:

δ	a	b	c
q_0	q_1	q_2	q_2
q_1	q_2	q_0	q_2
q_2	q_2	q_2	q_2

  Taula horren bidez $\delta(q_0, a)$-ren balioa q_2 dela, $\delta(q_0, b)$-ren balioa q_1 dela, eta abar adierazten da.

- q_0 hasierako egoera da. Hasierako egoera beti bakarra da.
- $Y = \{q_0, q_1\}$. Adibide honetan hasierako egoera Y multzoan ager daitekeela ikus dezakegu.



3.4 Irudia: c-rik ez eta a eta b, orden horretan, tartekatuta dituzten hitzen lengoaiari dagokion AFDa.

Orain $(q_0, abab)$ konfigurazioari dagokion konputazioa garatuko dugu urratsez urrats:

1. urratsa: $\delta^*(q_0, abab) = \delta^*(\delta(q_0, a), bab) = \delta^*(q_1, bab)$

2. urratsa: $\delta^*(q_1, bab) = \delta^*(\delta(q_1, b), ab) = \delta^*(q_0, ab)$

3. urratsa: $\delta^*(q_0, ab) = \delta^*(\delta(q_0, a), b) = \delta^*(q_1, b)$

4. urratsa: $\delta^*(q_1, b) = \delta^*(q_1, b\varepsilon) = \delta^*(\delta(q_1, b), \varepsilon) = \delta^*(q_0, \varepsilon)$

5. urratsa: $\delta^*(q_0, \varepsilon) = q_0$

Konputazioa Y multzokoa den q_0 egoeran amaitu denez, hitza AFD honi dagokion lengoaiakoa da.

Konputazio hori honela adieraz dezakegu grafikoki:

$$\begin{array}{c} (q_0, abab) \\ | \\ (q_1, bab) \\ | \\ (q_0, ab) \\ | \\ (q_1, b) \\ | \\ (q_0, \varepsilon) \end{array}$$

Jarraian (q_0, abc) konfigurazioari dagokion konputazioa garatuko dugu urratsez urrats:

1. urratsa: $\delta^*(q_0, abc) = \delta^*(\delta(q_0, a), bc) = \delta^*(q_1, bc)$

2. urratsa: $\delta^*(q_1, bc) = \delta^*(\delta(q_1, b), c) = \delta^*(q_0, c)$

3. urratsa: $\delta^*(q_0, c) = \delta^*(q_0, c\varepsilon) = \delta^*(\delta(q_0, c), \varepsilon) = \delta^*(q_2, \varepsilon)$

4. urratsa: $\delta^*(q_2, \varepsilon) = q_2$

Konputazioa Y multzokoa ez den q_2 egoeran amaitu denez, hitza ez da AFD honi dagokion lengoaiakoa.

Jarraian konputazio horri dagokion konfigurazio-sekuentzia dator:

$$\begin{array}{c} (q_0, abc) \\ | \\ (q_1 bc) \\ | \\ (q_0, c) \\ | \\ (q_2, \varepsilon) \end{array}$$

3.1.4 *Adibidea:* 10*-rekin bukatzen diren hitzez osatutako lengoaiari dagokion AFDa*

Adibide honetako AFDak (3.5 irudia begiratu) 10 katearekin bukatzen diren $A = \{0, 1\}$ alfabetoaren gainean definitutako hitzentzat "Bai" erantzungo du. Beraz, esate baterako 0011010 eta 0010 hitzen kasuan "Bai" erantzungo du, baina 000, 1101, ε eta 0 hitzen kasuan "Ez" erantzungo du. Lengoaiaren definizio formala honako hau da:

$$\{w \mid w \in A^* \wedge \exists u(u \in A^* \wedge w = u10)\}$$

AFDak hiru egoera ditu, q_0, q_1 eta q_2. Hasierako egoera q_0 da eta q_2 egoeran bukatuz gero erantzuna "Bai" izango da.

Jarraian AFDa boskote bezala definituko da.

$$(Q, A, \delta, q_0, Y)$$

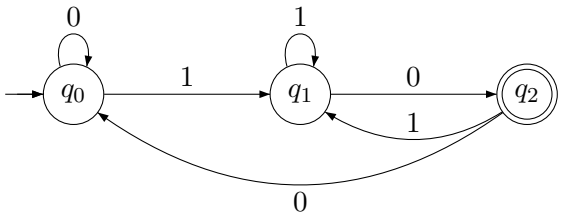
AFDaren bost osagaiak honako hauek dira:

- $Q = \{q_0, q_1, q_2\}$
- $A = \{0, 1\}$
- $\delta : Q \times A \to Q$ jarraian datorren taularen bidez definitutako trantsizio funtzioa da:

δ	0	1
q_0	q_0	q_1
q_1	q_2	q_1
q_2	q_0	q_1

  Taula horren bidez $\delta(q_0, 0)$-ren balioa q_0 dela, $\delta(q_0, 1)$-ren balioa q_1 dela, eta abar adierazten da.

- q_0 hasierako egoera da. Hasierako egoera beti bakarra da.
- $Y = \{q_2\}$.



3.5 Irudia: 10 katearekin bukatzen diren hitzen lengoaiari dagokion AFD-a.

Hona hemen $(q_0, 0010)$ konfigurazioari dagokion urratsez urrats garatutako konputazioa:

1. urratsa: $\delta^*(q_0, 0010) = \delta^*(\delta(q_0, 0), 010) = \delta^*(q_0, 010)$

2. urratsa: $\delta^*(q_0, 010) = \delta^*(\delta(q_0, 0), 10) = \delta^*(q_0, 10)$

3. urratsa: $\delta^*(q_0, 10) = \delta^*(\delta(q_0, 1), 0) = \delta^*(q_1, 0)$

4. urratsa: $\delta^*(q_1, 0) = \delta^*(q_1, 0\varepsilon) = \delta^*(\delta(q_1, 0), \varepsilon) = \delta^*(q_2, \varepsilon)$

5. urratsa: $\delta^*(q_2, \varepsilon) = q_2$

Konputazioa Y multzokoa den q_2 egoeran amaitu denez, hitza AFD honi dagokion lengoaiakoa da.

Hona hemen konputazio hori adar bakarreko zuhaitz bezala ipinita:

$$\begin{array}{c} (q_0, 0010) \\ | \\ (q_0, 010) \\ | \\ (q_0, 10) \\ | \\ (q_1, 0) \\ | \\ (q_2, \varepsilon) \end{array}$$

Beste adibide bat izateko, $(q_0, 111)$ konfigurazioari dagokion urratsez urrats garatutako konputazioa honako hau da:

1. urratsa: $\delta^*(q_0, 111) = \delta^*(\delta(q_0, 1), 11) = \delta^*(q_1, 11)$

2. urratsa: $\delta^*(q_1, 11) = \delta^*(\delta(q_1, 1), 1) = \delta^*(q_1, 1)$

3. urratsa: $\delta^*(q_1, 1) = \delta^*(q_1, 1\varepsilon) = \delta^*(\delta(q_1, 1), \varepsilon) = \delta^*(q_1, \varepsilon)$

4. urratsa: $\delta^*(q_1, \varepsilon) = q_1$

Konputazioa Y multzokoa ez den q_1 egoeran amaitu denez, hitza ez da AFD honi dagokion lengoaiakoa.

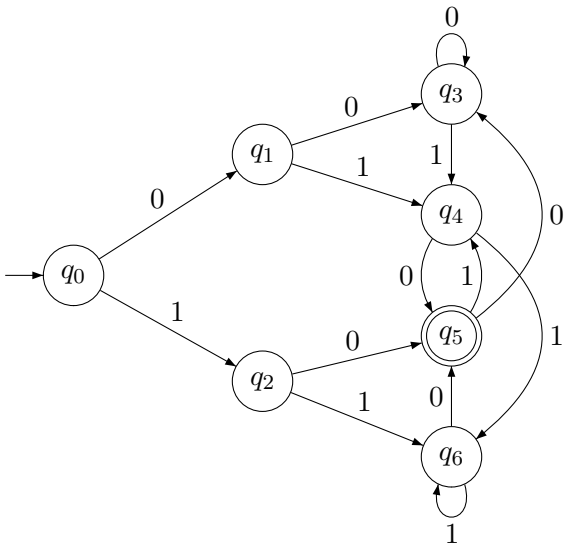
Konputazio hori jarraian erakusten den konfigurazio-sekuentzia bezala adieraz daiteke:

$$\begin{array}{c} (q_0, 111) \\ | \\ (q_1 11) \\ | \\ (q_1, 1) \\ | \\ (q_1, \varepsilon) \end{array}$$

Orokorrean, lengoaia batentzat AFD bat baino gehiago defini daitezke. Adibidez 3.6 irudian

$$\{w \mid w \in A^* \wedge \exists u(u \in A^* \wedge w = u10)\}$$

lengoaiako hitzei "Bai" eta lengoaia horretakoak ez diren hitzei "Ez" erantzuten dien beste AFD bat dugu. AFD honek 3.5 irudikoak baino egoera gehiago ditu baina baliokideak dira, lengoaia bera dagokie biei.



3.6 Irudia: 10 azpikatearekin amaitzen diren hitzen lengoaiarentzat beste AFD bat.

3.2 AFDen minimizazioa

AFDen diseinua lantzerakoan, 10 azpikatearekin bukatzen diren hitzez osatutako lengoaiaren adibidean erakutsi den bezala, lengoaia bakoitzarentzat AFD desberdinak diseina daitezke (3.5 eta 3.6 irudiak begiratu). AFDen arteko desberdintasuna egoera-kopuruan egon ohi da.

Gai honetan AFD bat nola minimiza daitekeen azalduko da. AFD bat emanda, beharrezkoak ez diren egoerak ezabatu eta lortu ahal den AFD txikiena itzultzen duen metodo bat aplikatuko dugu.

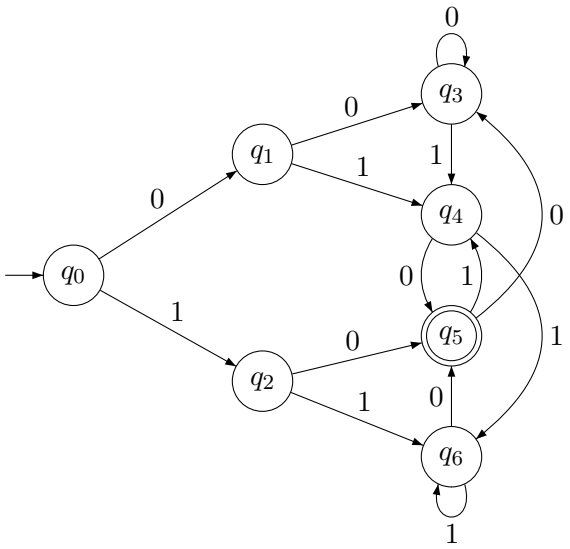
Metodoa adibideak garatuz azalduko da.

3.2.1 Minimizazioa: 10 azpikatearekin bukatzen diren hitzez osatutako lengoaiari dagokion AFDa

3.7 irudian $\{w \mid w \in A^* \wedge \exists u(u \in A^* \wedge w = u10)\}$ lengoaiari dagokion AFD bat ikus daiteke. Lengoaia hori $A = \{0, 1\}$ alfabetoaren gainean dafinituta dago.

AFD hori minimizatzeko jarraitu beharreko urratsak azalduko dira jarraian. Metodoa jokaera bera duten egoerak zein diren identifikatzean datza. Egoera batzuk beti jokaera bera badute, egoera horietako bat laga eta beste denak ezaba daitzeke. Horrela, jokaera-mota bakoitzeko egoera bat bakarrik gelditzuko da eta gelditzen diren egoera horiek beharrezkoak direla jakingo dugu.

Kontuan izan AFD bateko egoera bakoitzak memoria bezala jokatzen duela eta egoera bakoitza baldintzaren bat gogoratzeko erabiltzen dela. Baldintza bera gogoratzen duten egoera bat baino gehiago edukitzea alferrikakoa da. Horregatik metodo honek baldintza bera gogoratzen duten egoeratatik bat bakarrik mantenduko du.



3.7 Irudia: 10 azpikatearekin bukatzen diren hitzez osatutako lengoaiari dagokion AFDa.

Lehenengo zatiketa

Lehenengo zatiketan bi multzo sortuko dira. Batean Y multzokoak direnak sartuko dira ("Bai" erantzuten dutenak) eta bestean Y multzokoak ez direnak ("Ez" erantzuten dutenak). Multzo batean agertzen den azpiindize txikiena j baldin bada, multzoa $[q_j]$ bezala identifikatuko dugu.

$$[q_0] = \{q_0, q_1, q_2, q_3, q_4, q_6\}$$
$$[q_5] = \{q_5\}$$

Bigarren zatiketa

Gogoan izan δ trantsizio funtzioari dagokion taula honako hau dela:

δ	0	1
q_0	q_1	q_2
q_1	q_3	q_4
q_2	q_5	q_6
q_3	q_3	q_4
q_4	q_5	q_6
q_5	q_3	q_4
q_6	q_5	q_6

Orain egoera bakoitzetik zein egoeratara joango garen adierazi beharrean, egoera bakoitzetik zein multzotara joango garen adieraziz taula hori eraldatu egingo dugu. Horrela, q_1 egoeran gaudenean 0 sinboloa irakurriz q_3 egoerara joango garenez, eta q_3 egoera $[q_0]$ multzoan dagoenez, q_3 ipini beharrean $[q_0]$ ipiniko dugu. Kasu guztietan aldaketa hori egin behar da:

δ	0	1
q_0	$[q_0]$	$[q_0]$
q_1	$[q_0]$	$[q_0]$
q_2	$[q_5]$	$[q_0]$
q_3	$[q_0]$	$[q_0]$
q_4	$[q_5]$	$[q_0]$
q_5	$[q_0]$	$[q_0]$
q_6	$[q_5]$	$[q_0]$

Jarraian $[q_0]$ eta $[q_5]$ multzoak zatitu behar al diren aztertuko da. $[q_5]$ multzoan elementu bakarra dagoenez, hor ezin da zatiketarik egin. Beraz $[q_0]$ multzoan zentratuko gara.

$[q_0]$ multzoko egoeren jokaera aztertuz, q_0, q_1 eta q_3 egoerek jokaera bera dutela ikus dezakegu: 0 sinboloa irakurriz gero $[q_0]$ multzoko egoera batera doaz eta 1 sinboloa irakurriz $[q_0]$ multzoko egoera batera doaz. Beste aldetik, q_2, q_4 eta q_6 egoerek beste jokaera bat dute: 0 sinboloa irakurriz gero $[q_5]$ multzoko egoera batera doaz eta 1 sinboloa irakurriz $[q_0]$ multzoko egoera batera doaz. Ondorioz, bi jokaera desberdin horiek kontuan hartuz $[q_0]$ multzoa bitan zatituko dugu. Guztira hiru multzo geldituko zaizkigu:

$$[q_0] = \{q_0, q_1, q_3\}$$
$$[q_2] = \{q_2, q_4, q_6\}$$
$$[q_5] = \{q_5\}$$

GARRANTZITSUA: q_5 egoerak $[q_0]$ multzoko egoeren jokaera bera du, baina hala ere ez dira multzo berean sartu behar. Metodo honetan multzoak zatituz joan behar dugu, baina multzo desberdietako egoerak ez dira inoiz nahastu behar, nahiz eta jokaera bera izan.

Ez dago hirugarren zatiketarik

Bigarren zatiketa kontuan hartuz trantsizio-taula eguneratu egingo dugu:

δ	0	1
q_0	$[q_0]$	$[q_2]$
q_1	$[q_0]$	$[q_2]$
q_2	$[q_5]$	$[q_2]$
q_3	$[q_0]$	$[q_2]$
q_4	$[q_5]$	$[q_2]$
q_5	$[q_0]$	$[q_2]$
q_6	$[q_5]$	$[q_2]$

Jarraian $[q_0]$ eta $[q_2]$ multzoak zatitu beharrik ba al dagoen erabakiko dugu. Lehen esan den bezala, $[q_5]$ multzoan elementu bakarra dagoenez ezin da egin zatiketarik. Beraz $[q_0]$ eta $[q_2]$ multzoetan zentratuko gara.

$[q_0]$ multzokoak diren q_0, q_1 eta q_3 egoerek jokaera bera dutela ikus dezakegu: 0 sinboloa irakurtzean $[q_0]$ multzoko egoera batera doaz eta 1 sinboloa irakurriz $[q_2]$ multzoko egoera batera doaz. Beraz $[q_0]$ multzoa ez da zatitu behar. $[q_2]$ multzoan ere antzekoa gertatzen da. Multzo horretakoak diren q_2, q_4 eta q_6 egoerek jokaera bera dute: 0 sinboloa irakurtzen bada, $[q_5]$ multzoko egoera batera doaz eta 1 sinboloa irakurtzen bada, $[q_2]$ multzoko egoera batera doaz. Beraz $[q_2]$ multzoa ere ez da zatitu behar.

GARRANTZITSUA: q_5 egoerak $[q_0]$ multzoko egoeren jokaera bera du, baina hala ere ez dira multzo berean sartu behar. Metodo honetan multzoak zatituz joan behar dugu, baina multzo desberdietako egoerak ez dira inoiz nahastu behar, nahiz eta jokaera bera izan.

AFD txikiena

Hiru egoera-multzo daudela ikusi dugu metodoa aplikatuz:

$$[q_0] = \{q_0, q_1, q_3\}$$
$$[q_2] = \{q_2, q_4, q_6\}$$
$$[q_5] = \{q_5\}$$

Egoera multzo bakoitza egoera bat izango da AFD berrian. Azkeneko taula kontuan hartuz egoera berrien arteko trantsizioak kalkulatu beharko dira:

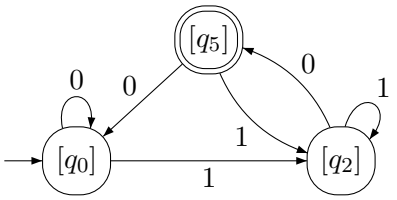
δ	0	1
q_0	$[q_0]$	$[q_2]$
q_2	$[q_5]$	$[q_2]$
q_5	$[q_0]$	$[q_2]$

Minimizatu beharreko AFDko q_0 hasierako egoera $[q_0]$ multzoan dagoenez, $[q_0]$ izango da AFD berriko hasierako egoera. Bestalde "Bai" erantzungo duen egoera $[q_5]$ izango da, minimizatu beharreko AFDan "Bai" erantzuten duten egoerez osatuta dagoelako. AFD berria 3.8 irudian ikus daiteke.

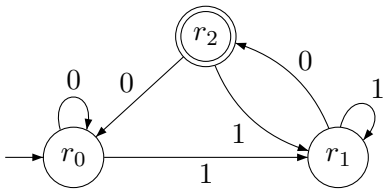
AFD berriko egoerak $0, 1, 2, 3, \ldots$ ordenean zenbatuta joan daitezen, berrizendatu egin ohi dira. Berrizendatu ondoren gelditu den AFD berria 3.9 irudian ikus daiteke.

3.2.2 Minimizazioa: b-rik eta c-rik ez duten hitzez osatutako lengoaiari dagokion AFDa

3.10 irudian $A = \{a, b, c\}$ alfabetoaren gainean definitutako $\{w \mid w \in A^* \wedge |w|_a = |w|\}$ lengoaiari dagokion AFD bat ikus dezakegu.

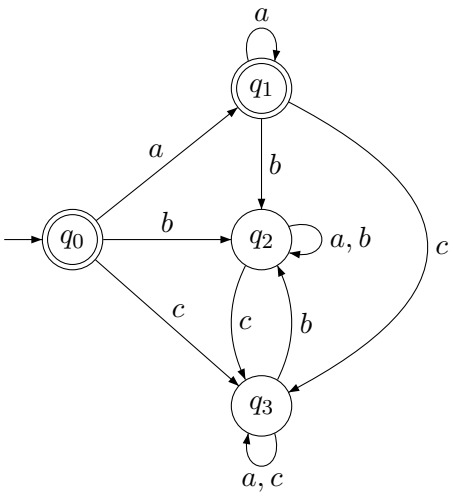


3.8 Irudia: 3.7 irudiko AFDa minimizatuz lortu den AFD berria.



3.9 Irudia: 3.8 irudiko AFDa berrizendatuz lortu den AFDa.

AFD hori minimizatzeko jarraitu beharreko urratsak azalduko dira jarraian. Lehenago esan den bezala, metodo honetan jokaera bera duten egoerak zein diren erabakitzen da eta jokaera-mota bakoitzeko egoera bat bakarrik lagatzen da, beste egoerak ezabatuz. AFDetan egoerak memoria bezala erabiltzen direnez eta egoera bakoitzak baldintzaren bat gogoratzeko balio duenez, baldintza bera gogoratzen duten egoera bat baino gehiago edukitzea alferrikakoa da. Horregatik metodo honek baldintza bera gogoratzen duten egoeratatik bat bakarrik mantenduko du.



3.10 Irudia: b-rik eta c-rik ez duten hitzez osatutako lengoaiari dagokion AFDa.

Lehenengo zatiketa

Lehenengo zatiketan bi multzo sortuko dira. Batean Y multzokoak direnak sartuko dira ("Bai" erantzuten dutenak) eta bestean Y multzokoak ez direnak ("Ez" erantzuten dutenak). Multzo batean agertzen den azpiindize txikiena j baldin bada, multzoa $[q_j]$ bezala identifikatuko dugu.

$$[q_0] = \{q_0, q_1\}$$
$$[q_2] = \{q_2, q_3\}$$

Ez dago bigarren zatiketarik

δ trantsizio funtzioari dagokion taula honako hau da:

δ	a	b	c
q_0	q_1	q_2	q_3
q_1	q_1	q_2	q_3
q_2	q_2	q_2	q_3
q_3	q_3	q_2	q_3

Orain egoera bakoitzetik zein egoeratara joango garen adierazi beharrean, egoera bakoitzetik zein multzotara joango garen adieraziz taula hori eraldatu egingo dugu. Horrela, q_1 egoeran gaudenean 0 sinboloa irakurriz q_1 egoerara joango garenez, eta q_1 egoera $[q_0]$ multzoan dagoenez, q_1 ipini beharrean $[q_0]$ ipiniko dugu. Kasu guztietan aldaketa hori egin behar da:

δ	a	b	c
q_0	$[q_0]$	$[q_2]$	$[q_2]$
q_1	$[q_0]$	$[q_2]$	$[q_2]$
q_2	$[q_2]$	$[q_2]$	$[q_2]$
q_3	$[q_2]$	$[q_2]$	$[q_2]$

Jarraian $[q_0]$ eta $[q_2]$ multzoak zatitu behar al diren erabakiko da.

$[q_0]$ multzokoak diren q_0 eta q_1 egoerek jokaera bera dutela ikus dezakegu: a sinboloa irakurtzean $[q_0]$ multzoko egoera batera doaz, b sinboloa irakurriz $[q_2]$ multzoko egoera batera doaz eta c sinboloa irakurriz $[q_2]$ multzoko egoera batera doaz. Beraz $[q_0]$ multzoa ez da zatitu behar.

$[q_2]$ multzoan ere antzekoa gertatzen da. $[q_2]$ multzoan q_2 eta q_3 egoerek jokaera bera dutela ikus dezakegu: a sinboloa irakurtzean $[q_2]$ multzoko egoera batera doaz, b sinboloa irakurriz $[q_2]$ multzoko egoera batera doaz eta c sinboloa irakurriz $[q_2]$ multzoko egoera batera doaz. Beraz $[q_2]$ multzoa ez da zatitu behar.

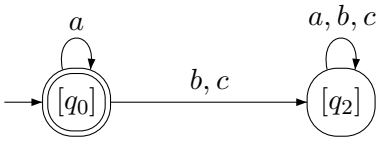
AFD txikiena

Beraz bi multzo gelditu dira:

$$[q_0] = \{q_0, q_1\}$$
$$[q_2] = \{q_2, q_3\}$$

Multzo bakoitza egoera bat izango da AFD minimizatuan. Egoeren arteko trantsizioak eraiki dugun azkeneko taula erabiliz kalkulatu behar dira:

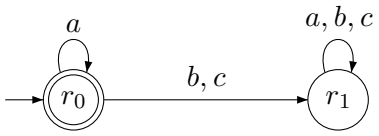
δ	a	b	c
q_0	$[q_0]$	$[q_2]$	$[q_2]$
q_2	$[q_2]$	$[q_2]$	$[q_2]$



3.11 Irudia: 3.10 irudiko AFDa minimizatuz lortu den AFDa.

Minimizatu beharreko AFDko q_0 hasierako egoera $[q_0]$ multzoan dagoenez, $[q_0]$ izango da AFD berriko hasierako egoera. Bestalde "Bai" erantzungo duen egoera $[q_0]$ izango da, minimizatu beharreko AFDan "Bai" erantzuten duten egoerez osatuta dagoelako. AFD berria 3.11 irudian ikus daiteke.

AFD berriko egoerak $0, 1, 2, 3, \ldots$ ordenean zenbatuta joan daitezen, berrizendatu egin ohi dira. Berrizendatu ondoren gelditu den AFD berria 3.12 irudian ikus daiteke.



3.12 Irudia: 3.11 irudiko AFDko egoerak berrizendatuz lortutako AFDa.

3.2.3 Minimizazioa: $a^j b^k$ erako hitzez osatutako lengoaiari dagokion AFDa

3.13 irudian $A = \{a, b\}$ alfabetoaren gainean definitutako honako lengoaia honi dagokion AFD bat ikus dezakegu:

$$\{w \mid w \in A^* \wedge \exists u, v(u \in A^* \wedge v \in A^* \wedge |u| = |u|_a \wedge |v| = |v|_b \wedge w = uv)\}$$

Lengoaia hori beste era honetara ere defini daiteke:

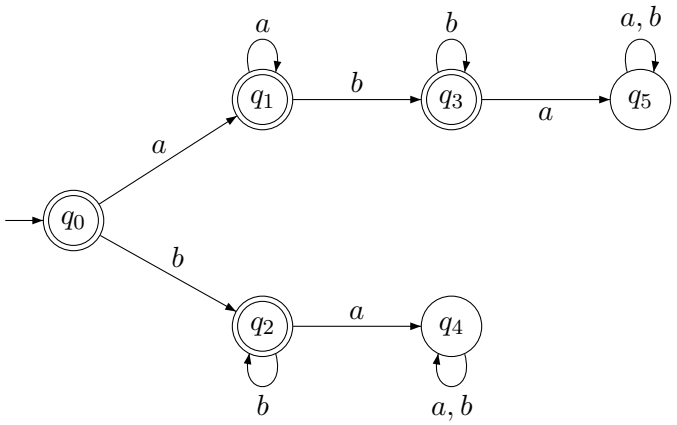
$$\{w \mid w \in A^* \wedge \exists j, k(j \in \mathbb{N} \wedge k \in \mathbb{N} \wedge w = a^j b^k)\}$$

Hor $\mathbb{N}$ zenbaki arrunten multzoa da: $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$.

AFD hori minimizatzeko jarraitu beharreko urratsak azalduko dira jarraian. Aurreko adibideetan ere esan den bezala, metodo honetan jokaera bera duten egoerak zein diren erabakitzen da eta jokaeramota bakoitzeko egoera bat bakarrik lagatzen da, beste egoerak ezabatuz. AFDetan egoerak memoria bezala erabiltzen direnez eta egoera bakoitzak baldintzaren bat gogoratzeko balio duenez, baldintza bera gogoratzen duten egoera bat baino gehiago edukitzea alferrikakoa da. Horregatik metodo honek baldintza bera gogoratzen duten egoeratatik bat bakarrik mantenduko du.

Lehenengo zatiketa

Lehenengo zatiketan bi multzo sortuko dira. Batean Y multzokoak direnak sartuko dira ("Bai" erantzuten dutenak) eta bestean Y multzokoak ez direnak ("Ez" erantzuten dutenak). Multzo batean agertzen den azpiindize txikiena j baldin bada, multzoa $[q_j]$ bezala identifikatuko dugu.



3.13 Irudia: $a^j b^k$ erako hitzez osatutako lengoaiari dagokion AFDa.

$$[q_0] = \{q_0, q_1, q_2, q_3\}$$
$$[q_4] = \{q_4, q_5\}$$

Bigarren zatiketa

δ trantsizio funtzioari dagokion taula honako hau da:

δ	a	b
q_0	q_1	q_2
q_1	q_1	q_3
q_2	q_4	q_2
q_3	q_5	q_3
q_4	q_4	q_4
q_5	q_5	q_5

Orain egoera bakoitzetik zein egoeratara joango garen adierazi beharrean, egoera bakoitzetik zein multzotara joango garen adieraziz taula hori eraldatu egingo dugu. Horrela, q_1 egoeran gaudenean b sinboloa irakurriz q_3 egoerara joango garenez, eta q_3 egoera $[q_0]$ multzoan dagoenez, q_3 ipini beharrean $[q_0]$ ipiniko dugu. Kasu guztietan aldaketa hori egin behar da:

δ	a	b
q_0	$[q_0]$	$[q_0]$
q_1	$[q_0]$	$[q_0]$
q_2	$[q_4]$	$[q_0]$
q_3	$[q_4]$	$[q_0]$
q_4	$[q_4]$	$[q_4]$
q_5	$[q_4]$	$[q_4]$

Jarraian $[q_0]$ eta $[q_4]$ multzoak zatitu behar al diren aztertuko da.

$[q_0]$ multzoko egoeren jokaera aztertuz, q_0 eta q_1 egoerek jokaera bera dutela ikus dezakegu. a sinboloa irakurriz gero $[q_0]$ multzoko egoera batera doaz eta b sinboloa irakurriz $[q_0]$ multzoko egoera

batera doaz. Beste aldetik, q_2 eta q_3 egoerek beste jokaera bat dute. a sinboloa irakurriz gero $[q_4]$ multzoko egoera batera doaz eta b sinboloa irakurriz $[q_0]$ multzoko egoera batera doaz. Ondorioz, bi jokaera desberdin horiek kontuan hartuz $[q_0]$ multzoa bitan zatituko dugu.

$$[q_0] = \{q_0, q_1\}$$
$$[q_2] = \{q_2, q_3\}$$

$[q_4]$ multzoan, q_4 eta q_5 egoerek jokaera bera dute. a sinboloa irakurriz gero $[q_4]$ multzoko egoera batera doaz eta b sinboloa irakurriz $[q_4]$ multzoko egoera batera doaz. Beraz $[q_4]$ multzoa ez da zatitu behar.

Bigarren zatiketaren ondoren hiru multzo gelditzen dira:

$$[q_0] = \{q_0, q_1\}$$
$$[q_2] = \{q_2, q_3\}$$
$$[q_4] = \{q_4, q_5\}$$

Ez dago hirugarren zatiketarik

Bigarren zatiketa kontuan hartuz trantsizio-taula eguneratu egingo dugu:

δ	a	b
q_0	$[q_0]$	$[q_2]$
q_1	$[q_0]$	$[q_2]$
q_2	$[q_4]$	$[q_2]$
q_3	$[q_4]$	$[q_2]$
q_4	$[q_4]$	$[q_4]$
q_5	$[q_4]$	$[q_4]$

Jarraian $[q_0]$, $[q_2]$ eta $[q_4]$ multzoak zatitu beharrik ba al dagoen erabakiko dugu.

$[q_0]$ multzokoak diren q_0 eta q_1 egoerek jokaera bera dutela ikus dezakegu: a sinboloa irakurtzean $[q_0]$ multzoko egoera batera doaz eta b sinboloa irakurriz $[q_0]$ multzoko egoera batera doaz. Beraz $[q_0]$ multzoa ez da zatitu behar. $[q_2]$ multzoan ere antzekoa gertatzen da. Multzo horretakoak diren q_2 eta q_3 egoerek jokaera bera dute: a sinboloa irakurtzen bada, $[q_4]$ multzoko egoera batera doaz eta b sinboloa irakurtzen bada, $[q_2]$ multzoko egoera batera doaz. Beraz $[q_2]$ multzoa ere ez da zatitu behar. Bukatzeko, $[q_4]$ multzoa ere ez da zatitu behar. Izan ere q_4 eta q_5 egoerek jokaera bera dute: a sinboloa irakurtzen bada, $[q_4]$ multzoko egoera batera doaz eta b sinboloa irakurtzen bada, $[q_4]$ multzoko egoera batera doaz.

AFD txikiena

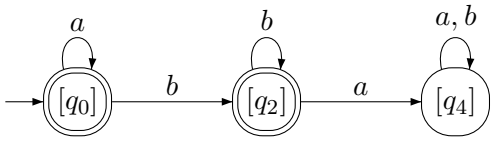
Hiru egoera-multzo daudela ikusi dugu metodoa aplikatuz:

$$[q_0] = \{q_0, q_1\}$$
$$[q_2] = \{q_2, q_3\}$$
$$[q_5] = \{q_4, q_5\}$$

Egoera-multzo bakoitza egoera bat izango da AFD berrian. Azkeneko taula kontuan hartuz egoera berrien arteko trantsizioak kalkulatu beharko dira:

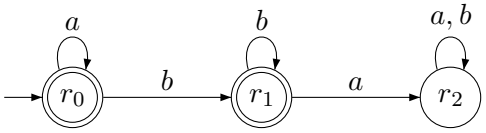
δ	a	b
q_0	$[q_0]$	$[q_2]$
q_2	$[q_4]$	$[q_2]$
q_4	$[q_4]$	$[q_4]$

Minimizatu beharreko AFDko q_0 hasierako egoera $[q_0]$ multzoan dagoenez, $[q_0]$ izango da AFD berriko hasierako egoera. Bestalde "Bai" erantzungo duten egoerak $[q_0]$ eta $[q_2]$ izango dira, minimizatu beharreko AFDan "Bai" erantzuten duten egoerez osatuta daudelako. AFD berria 3.14 irudian ikus daiteke.



3.14 Irudia: 3.13 irudiko AFDa minimizatuz lortutako AFDa.

AFD berriko egoerak $0, 1, 2, 3, \ldots$ ordenean zenbatuta joan daitezen, berrizendatu egin ohi dira. Egoerak berrizendatu ondoren gelditu den AFD berria 3.15 irudian ikus daiteke.



3.15 Irudia: 3.14 irudiko AFDa berrizendatuz lortu den AFDa.

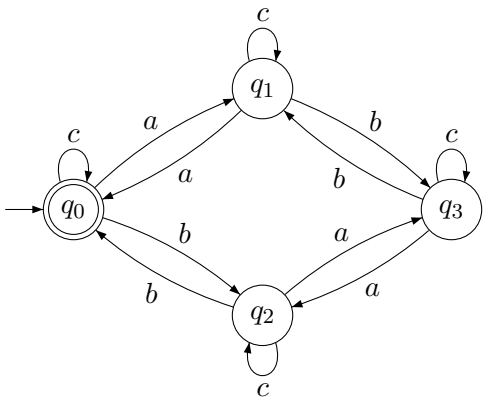
3.2.4 Minimizazioa: a eta b sinboloak kopuru bikoitian dituzten hitzen lengoaiari dogokion AFDa

3.16 irudian honako lengoaia honi dagokion AFD bat ikus daiteke:

$$\{w \mid w \in A^* \wedge |w|_a \bmod 2 = 0 \wedge |w|_b \bmod 2 = 0\}$$

Lengoaia hori $A = \{a, b, c\}$ alfabetoaren gainean definituta dago.

AFD hori minimizatzeko jarraitu beharreko urratsak azalduko dira jarraian. Metodo honetan jokaera bera duten egoerak zein diren erabakitzen da eta jokaera-mota bakoitzeko egoera bat bakarrik lagatzen da, beste egoerak ezabatuz. AFDetan egoerak memoria bezala erabiltzen direnez eta egoera bakoitzak baldintzaren bat gogoratzeko balio duenez, baldintza bera gogoratzen duten egoera bat baino gehiago edukitzea alferrikakoa da. Horregatik metodo honek baldintza bera gogoratzen duten egoeratatik bat bakarrik mantenduko du.



3.16 Irudia: a eta b kopuru bikoitian dituzten hitzen lengoaiari dagokion AFD bat.

Lehenengo zatiketa

Lehenengo zatiketan bi multzo sortuko dira. Batean Y multzokoak direnak sartuko dira ("Bai" erantzuten dutenak) eta bestean Y multzokoak ez direnak ("Ez" erantzuten dutenak). Multzo batean agertzen den azpiindize txikiena j baldin bada, multzoa $[q_j]$ bezala identifikatuko dugu.

$$[q_0] = \{q_0\}$$
$$[q_1] = \{q_1, q_2, q_3\}$$

Bigarren zatiketa

δ trantsizio funtzioari dagokion taula honako hau da:

δ	a	b	c
q_0	q_1	q_2	q_0
q_1	q_0	q_3	q_1
q_2	q_3	q_0	q_2
q_3	q_2	q_1	q_3

Orain egoera bakoitzetik zein egoeratara joango garen adierazi beharrean, egoera bakoitzetik zein multzotara joango garen adieraziz taula hori eraldatu egingo dugu. Horrela, q_1 egoeran gaudenean b sinboloa irakurriz q_3 egoerara joango garenez, eta q_3 egoera $[q_1]$ multzoan dagoenez, q_3 ipini beharrean $[q_1]$ ipiniko dugu. Kasu guztietan aldaketa hori egin behar da:

δ	a	b	c
q_0	$[q_1]$	$[q_1]$	$[q_0]$
q_1	$[q_0]$	$[q_1]$	$[q_1]$
q_2	$[q_1]$	$[q_0]$	$[q_1]$
q_3	$[q_1]$	$[q_1]$	$[q_1]$

Jarraian $[q_0]$ eta $[q_1]$ multzoak zatitu behar al diren aztertuko da.

$[q_0]$ multzoak egoera bakarra duenez, ezin da gehiago zatitu. $[q_1]$ multzoko egoeren jokaera aztertuz, q_0, q_1 eta q_2 egoerek jokaera desberdina dutela ikus dezakegu. Hasteko, q_1 egoeran egonda, a edo b sinboloak irakurriz gero $[q_1]$ multzoko egoera batera joan beharko da eta c sinboloa irakurriz gero $[q_0]$ multzoko egoera batera joan beharko da. Bestalde, q_2 egoeran egonda, a edo c sinboloak irakurriz gero $[q_1]$ multzoko egoera batera joan beharko da eta b sinboloa irakurriz gero $[q_0]$ multzoko egoera batera joan beharko da. Azkenik, q_3 egoeran egonda, a, b edo c sinboloak irakurriz gero $[q_1]$ multzoko egoera batera joan beharko da. Ondorioz, hiru jokaera desberdin horiek kontuan hartuz $[q_1]$ multzoa hirutan zatituko dugu.

$$[q_0] = \{q_0\}$$
$$[q_1] = \{q_1\}$$
$$[q_2] = \{q_2\}$$
$$[q_3] = \{q_3\}$$

Ez dago hirugarren zatiketarik

Multzo bakoitzak egoera bakarra duenez, ez dago multzo horiek gehiago zatitzerik. Beraz, behin betiko zatiketa honako hau da:

$$[q_0] = \{q_0\}$$
$$[q_1] = \{q_1\}$$
$$[q_2] = \{q_2\}$$
$$[q_3] = \{q_3\}$$

Ondorioz, 3.16 irudiko AFDtik ezin da egoerarik ezabatu, AFD hori izan daitekeen txikiena da adibide honetako lengoaiarentzat.

3.3 Automata Finitu Ez Deterministak (AFED)

Automata finitu ez determinista bat boskote bat da (bost osagai dituen egitura bat da):

$$(Q, A, \nu, q_0, Y)$$

Bost osagai horien definizioa honako hau da:

- Q egoeren multzo finitua da: $q_0, q_1, q_2, q_3, \ldots$
- A alfabetoa da
- ν trantsizio funtzio ez determinista da (ν sinboloa *nu* letra grekoa da). Q multzoko egoera bat eta A multzoko sinbolo bat emanda, egoera horretan gaudenean sinbolo hori irakurtzen badugu zein egoeretara joango garen adierazten du ν trantsizio funtzioak. ν funtzioaren mota honako hau da:

  $$\nu : Q \times A \rightarrow 2^Q$$

  Hor 2^Q multzoa $Q \rightarrow 2$ erako funtzio denez osatutako multzoa da eta $2 = \{0, 1\}$ da, 2 multzo horretan 0 balioak $False$ eta 1 balioak $True$ adierazten dutela ulertuz.
- q_0 hasierako egoera da. Hasierako egoera beti bakarra izango da.
- Y osagaia "Bai" erantzuten duten egoerez osatutako multzoa da. Y multzoa Q multzoaren azpimultzoa izango da, hau da, $Y \subseteq Q$.

AFD eta AFED-en arteko desberdintasun nagusiak honako hiru hauek dira:

1. AFD-etan, q_j egoeran egonda α sinbolo bat irakurtzen badugu, q_h beste egoera batera igaroko gara. AFED batean aldiz, q_j egoeran α sinbolo bat irakurtzen badugu, egoera-multzo batera pasatuko gara. Multzo hori hutsa, elementu bakarrekoa edo elementu bat baino gehiagokoa izan daiteke.
2. AFD-etan hitzeko sinbolo denak irakurri behar dira, hitz hutsa geratu arte. Gainera beti bukatu ahal izango dugu hitzen irakurketa. AFED-etan q_j egoera batean egonda α sinbolo bat irakurtzean gerta daiteke inora joaterik ez edukitzea. Kasu horretan, oraindik hitza irakurtzea bukatu ez bada ere, automata gelditu egingo da eta "Ez" erantzungo du, Y multzoko egoera batean al dagoen ala ez kontuan hartu gabe.
3. AFD-etan hitzaren irakurketa beti burutu ahal izango denez, hitza bukatutakoan Y egoera batean baldin bagaude "Bai" erantzungo da. AFED-etan, "Bai" erantzuteko hitz osoa bukatu beharko da eta gainera Y multzokoa den egoera batean bukatu beharko da. Beraz, AFED batean "Ez" erantzuteko bi aukera daude: hitza osorik irakurri ondoren Y-koa ez den egoera batean aurkitzea edo hitza irakurri ezinda gelditzea. Hitza irakurri ezinda gelditzen bagara, berdin da azkeneko egoera Y-koa al den ala ez.

AFED baten **funtzionamendua** honako hau da: hasieran AFEDa q_0 egoeran egongo da eta sarrera bezala A^* multzoko hitz bat jasoko du. Hitz hori osatzen duten sinboloak banan-banan irakurriko ditu. Sinbolo bakoitza irakurtzean zein egoeretara pasatu erabakiko da ν trantsizio funtzioa erabiliz. Beraz q_i egoeran batean gaudela A alfabetokoa den α sinboloa irakurtzen badugu eta $\nu(q_i, \alpha) = \{q_{j_1}, q_{j_2}, \ldots, q_{j_n}\}$ baldin bada, $\{q_{j_1}, q_{j_2}, \ldots, q_{j_n}\}$ egoeretara pasatuko gara. Hor $\nu(q_i, \alpha) = \varnothing$ baldin bada, ezin da aurrera jarraitu eta "Ez" erantzungo da. Hitzak beti finituak izango direnez, hitz bat

irakurtzeko prozesua ere finitua izango da. Gainera, lehenago esan den bezala, hitza irakurri ezinda era geldi gaitezke. Kasu horretan "Ez" erantzungo da. Hitza irakurtzea bukatutakoan 2^Q multzokoa den S egoera-multzo batean egongo gara. Kontuan izan $S \subseteq Q$ ere beteko dela. S multzoan Y multzokoa den egoeraren bat baldin badago, orduan automatak "Bai" erantzungo du, hitza automatari lotutako lengoaiakoa dela adieraziz. Baina hitza bukatzean S multzoan Y multzokoa den egoerarik ez badago, automatak "Ez" erantzungo du, hitza automatari lotutako lengoaiakoa ez dela adieraziz.

S multzoa 2^Q multzoko elementu bat izanda, hau da, S multzoa Q-ren azpimultzo bat izanda eta w elementua A^* multzoko hitz bat izanda, (S, w) bikoteari **konfigurazio ez determinista** deituko diogu. Konfigurazio ez deterministak $2^Q \times A^*$ motakoak dira. (S, w) erako konfigurazio ez determinista baten bidez S multzoko egoeretan gaudela eta oraindik w hitza irakurtzeko daukagula adierazten da. Adibidez, A alfabetoa $\{a, b, c\}$ baldin bada, oraingo konfigurazio ez determinista $(\{q_1, q_4\}, caaba)$ baldin bada eta ν trantsizio funtzioak $\nu(q_1, c) = \{q_2, q_6\}$ eta $\nu(q_4, c) = \{q_3, q_5, q_6\}$ betetzen dela adierazten badu, $caaba$ hitzeko lehenengo sinboloa irakurritakoan $\{q_2, q_3, q_5, q_6\}$ egoera-multzora pasatuko gara eta konfigurazio ez determinista berria $(\{q_2, q_3, q_5, q_6\}, aaba)$ izango da. Konfigurazio ez determinista berrian ikus daitekeen bezala, irakurtzeko gelditzen den hitza laburragoa da orain (irakurri den sinboloa desagertu egin baita).

Konfigurazio deterministak $Q \times A^*$ motakoak direnez eta konfigurazio ez deterministak $2^Q \times A^*$ motakoak direnez, konfigurazio deterministak eta ez deterministak bereiztea erraza da. Konfigurazio deterministak (q_j, w) erakoak izango dira eta konfigurazio ez deterministak $(\{q_{j_1}, q_{j_2}, \ldots, q_{j_n}\}, w)$ erakoak izango dira. Egitura aztertuz bi konfigurazio motak bereiztea erraza denez, *konfigurazio determinista* edo *konfigurazio ez determinista* idatzi beharrean *konfigurazioa* idatziko dugu automata finituei buruzko gai honetan zehar. Hala ere, konfigurazio bat determinista edo ez determinista dela azpimarratu nahi dugunean, ezaugarri hori zehaztuko dugu.

AFED batek burututako **konputazioa** edo kalkulu-prozesua ν trantsizio funtzioan oinarritzen den ν^* funtzioaren bidez definitzen da. ν^* funtzioaren mota honako hau da:

$$\nu^* : 2^Q \times A^* \rightarrow 2^Q$$

Beraz, S egoera-multzo bat eta w hitz bat emanda, (hau da, konfigurazio ez determinista bat emanda), ν^* funtzioak egoera-multzo bat itzuliko du (S multzoko egoeretatik abiatuz eta, ahal bada, w hitzeko sinbolo denak irakurriz, zein egoeretara iritsiko garen adieraziz).

ν^* funtzioaren definizio formala honako hau da:

- $\nu^*(S, \varepsilon) = S$
- $\nu^*(\varnothing, w) = \varnothing$
- $\nu^*(S, \alpha w) = \nu^*(\bigcup_{q_j \in S} \nu(q_j, \alpha), w)$

Definizio horren bidez honako hau adierazten da: irakurtzeko gelditzen den hitza hutsa baldin bada, hau da, ε, orduan S egoera-multzo berean jarraituko dugu. Irakurtzeko gelditzen den hitza hutsa ez bada, eta S multzo hutsa baldin bada, hau da, $\varnothing$ baldin bada, ez gaude inon eta beraz ezin dugu inora joan eta hori multzo hutsean egongo garela esanez adierazten da. S multzoa ere hutsa ez bada eta irakurtzeko gelditzen den hitza hutsa ez bada, eta hitz horretako lehenengo sinboloa A-koa den α sinboloa baldin bada, ν trantsizio funtzioa aplikatuko zaio S-ko egoera bakoitzari eta α sinboloari eta S-ko egoeretatik α-ren bidez irisgarriak diren egoeretara pasatuko gara. Egoera-multzo berrian kokatu ondoren w-eko gainontzeko sinboloak irakurtzen jarraitu beharko da era berean:

Esate baterako, demagun S multzoa $\{q_2, q_4, q_5, q_7\}$ dela. ν^*-ren bidezko urrats bat honela planteatuko genuke:

$$\nu^*(\{q_2, q_4, q_5, q_7\}, \alpha w) = \nu^*(\nu(q_2, \alpha) \cup \nu(q_4, \alpha) \cup \nu(q_5, \alpha) \cup \nu(q_7, \alpha), w)$$

Hor $\nu(q_2, \alpha)$-ren bidez q_2-tik α sinboloaren bidez irisgarriak diren egoeren multzoa adierazten da, $\nu(q_4, \alpha)$-ren bidez q_4-tik α sinboloaren bidez irisgarriak diren egoeren multzoa adierazten da, eta abar.

Beste adibide baten laguntzaz, ν^*-ren funtzionamendua hobeto zehaztuko dugu. Demagun alfabetoa $A = \{a, b, c\}$ dela, oraingo konfigurazioa $(\{q_1, q_4\}, caaba)$ dela eta ν honela definituta dagoela (hor agertzen diren kasuetarako):

- $\nu(q_1, c) = \{q_2, q_6\}$
- $\nu(q_4, c) = \{q_3, q_5, q_6\}$
- $\nu(q_2, a) = \varnothing$
- $\nu(q_3, a) = \{q_7\}$
- $\nu(q_5, a) = \varnothing$
- $\nu(q_6, a) = \{q_5, q_6\}$
- $\nu(q_7, a) = \{q_7\}$
- $\nu(q_5, b) = \{q_6\}$
- $\nu(q_6, b) = \{q_2, q_4\}$
- $\nu(q_7, b) = \{q_1, q_2\}$

$\nu^*(q_3, caab)$ konputazioa urratsez urrats honela burutuko litzateke:

- $caab$ hitza hutsa ez denez, $\nu^*(\{q_1, q_4\}, caab) = \nu^*(\nu(q_1, c) \cup \nu(q_4, c), aab)$.
- $\nu(q_1, c) = \{q_2, q_6\}$ eta $\nu(q_4, c) = \{q_3, q_5, q_6\}$ direnez, $\nu^*(\nu(q_1, c) \cup \nu(q_4, c), aab)$ espresioa $\nu^*(\{q_2, q_3, q_5, q_6\}, aab)$ bezala geratuko da.
- aab hitza hutsa ez denez, $\nu^*(\{q_2, q_3, q_5, q_6\}, aab) = \nu^*(\nu(q_2, a) \cup \nu(q_3, a) \cup \nu(q_5, a) \cup \nu(q_6, a), ab)$.
- $\nu(q_2, a) = \varnothing$, $\nu(q_3, a) = \{q_7\}$, $\nu(q_5, a) = \varnothing$ eta $\nu(q_6, a) = \{q_5, q_6\}$ direla kontuan hartuz, $\nu^*(\nu(q_2, a) \cup \nu(q_3, a) \cup \nu(q_5, a) \cup \nu(q_6, a), ab)$ espresiotik $\nu^*(\{q_5, q_6, q_7\}, ab)$ lortuko da.
- ab hitza hutsa ez denez, $\nu^*(\{q_5, q_6, q_7\}, ab) = \nu^*(\nu(q_5, a) \cup \nu(q_6, a) \cup \nu(q_7, a), b)$.
- $\nu(q_5, a) = \varnothing$, $\nu(q_6, a) = \{q_5, q_6\}$ eta $\nu(q_7, a) = \{q_7\}$ betetzen denez, $\nu^*(\nu(q_5, a) \cup \nu(q_6, a) \cup \nu(q_7, a), b)$ espresiotik $\nu^*(\{q_5, q_6, q_7\}, b)$ espresioa lortuko da.
- b hitza hutsa ez denez, $\nu^*(\{q_5, q_6, q_7\}, b) = \nu^*(\nu(q_5, b) \cup \nu(q_6, b) \cup \nu(q_7, b), \varepsilon)$.
- $\nu(q_5, b) = \{q_6\}$, $\nu(q_6, b) = \{q_2, q_4\}$ eta $\nu(q_7, b) = \{q_1, q_2\}$ izateagatik, $\nu^*(\nu(q_5, b) \cup \nu(q_6, b) \cup \nu(q_7, b), \varepsilon)$ espresiotik $\nu^*(\{q_1, q_2, q_4, q_6\}, \varepsilon)$ espresioa lortuko da.
- ε hitz hutsa geratzen denez, $\nu^*(\{q_1, q_2, q_4, q_6\}, \varepsilon)$ espresioaren balioa $\{q_1, q_2, q_4, q_6\}$ da.

Beraz, $\nu^*(\{q_1, q_4\}, caab) = \{q_1, q_2, q_4, q_6\}$.

konputazio hori konfigurazio ez deterministez eratutako honako sekuentzia honen bidez adieraz daiteke:

$$(\{q_1, q_4\}, caab)$$
$$\vdash$$
$$(\{q_2, q_3, q_5, q_6\}, aab)$$
$$\vdash$$
$$(\{q_5, q_6, q_7\}, ab)$$
$$\vdash$$
$$(\{q_5, q_6, q_7\}, b)$$
$$\vdash$$
$$(\{q_1, q_2, q_4, q_6\}, \varepsilon)$$

Egoera batean gaudenean sinbolo batentzat gezirik ez, gezi bakarra edo gezi bat baino gehiago egon daitezkeenez, automata hauek **ez deterministak** direla esaten da.

$N = (Q, A, \nu, q_0, Y)$ **AFED bati lotutako** $L(N)$ **lengoaia** N automatarengandik "Bai" erantzuna jasotzen duten hitzez osatutako lengoaia da. Beste era batera esanda, q_0-tik abiatuta osorik irakurri daitezkeen eta Y multzokoa den egoeraren bat baduen egoera-multzo batean lagatzen gaituzten A^*-ko hitzez osatutako lengoaia da. Era formalean definizioa honako hau izango litzateke:

$$L(N) = \{w \mid w \in A^* \wedge (\nu^*(\{q_0\}, w) \cap Y) \neq \varnothing\}$$

3.3.1 Adibidea: Hutsak ez diren eta bakarrik a sinboloaren errepikapenez eratuta dauden hitzez osatutako lengoaiari dagokion AFED-a

Adibide honetako AFEDak (3.17 irudia begiratu) a, b eta c sinboloak eduki ditzakeen hitz bat jasoko du sarrera bezala. AFED horri hitz bat emandakoan, hitz hori hutsa ez den eta bakarrik a sinboloaren errepikapenez eratuta dagoen hitza al den erabakiko du. Horrela bada, "Bai" erantzuna itzuliko du eta bestela "Ez" erantzuna itzuliko du. Lengoaiaren definizio formala honako hau da:

$$\{w \mid w \in A^* \wedge |w|_a = |w| \wedge |w| \geq 1\}$$

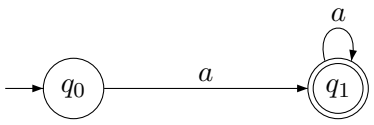
Beraz irteera "Bai" edo "Ez" izango da. AFEDa bi egoerez osatuta dago. q_0 hasierako egoera da. Hitz bat edo karaktere-kate bat irakurtzen hasten garenean q_0 egoeran gaude beraz. Lehenengo karakterea (edo sinboloa) a bada, AFEDa q_1 egoerara igaroko da, beraz, $\nu(q_0, a) = \{q_1\}$. Lehenengo karakterea b edo c bada, ezin da inora joan eta ondorioz $\nu(q_0, b) = \nu(q_0, c) = \varnothing$. AFEDa q_1 egoeran dagoenean, a sinboloa etortzen bada, AFEDa egoera berean mantenduko da, hau da, q_1 egoeran, hau da, $\nu(q_1, a) = \{q_1\}$. Bestalde, q_1 egoeran egonda, b edo c etortzen bada, ezingo da inora joan, $\nu(q_1, b) = \nu(q_1, c) = \varnothing$. Karaktere-kate osoa irakurtzea lortzen bada eta gainera q_1 egoeran bukatzen bada, horrek esan nahiko du karaktere-katea ez dela hutsa eta a sinboloaren errepikapenez osatuta dagoela. Kasu horretan $\nu^*(\{q_0\}, w) = \{q_1\}$ beteko da eta ondorioz $\nu^*(\{q_0\}, w) \cap Y \neq \varnothing$ izango da, izan ere $Y = \{q_1\}$ da eta $\{q_1\} \cap \{q_1\} \neq \varnothing$ izango baita eta erantzuna "Bai" izango da. Hasierako hitza ε baldin bada, AFEDak q_0 egoeran bukatuko du, hau da, $\nu^*(\{q_0\}, \varepsilon) = \{q_0\}$. Eta $\{q_0\} \cap Y = \varnothing$ denez, hau da, $\{q_0\} \cap \{q_1\} = \varnothing$ denez, AFEDak "Ez" erantzungo du. Aldiz, AFEDak ez badu lortzen karaktere-kate osoa irakurtzea, inora joan ezinda gelditu ko da. Beraz kasu horretan $\nu^*(\{q_0\}, w) = \varnothing$ izango da eta $\varnothing \cap Y = \varnothing$ denez, hau da, $\varnothing \cap \{q_1\} = \varnothing$ denez, AFED-ak "Ez" erantzungo du.

Jarraian AFDaren osagaiak definituko dira boskote eran:

$$(Q, A, \nu, q_0, Y)$$

Osagai bakoitza honela geratuko da:

- $Q = \{q_0, q_1\}$
- $A = \{a, b, c\}$



3.17 Irudia: Hutsak ez diren eta bakarrik a sinboloaren errepikapenez eratuta dauden hitzei dagokien AFEDa.

- $\nu : Q \times A \to 2^Q$ jarraian datorren taularen bidez definitutako trantsizio funtzioa da:

ν	a	b	c
q_0	$\{q_1\}$	$\varnothing$	$\varnothing$
q_1	$\{q_1\}$	$\varnothing$	$\varnothing$

  Taula horren bidez $\nu(q_0, a)$-ren balioa $\{q_1\}$ dela, $\nu(q_0, b)$-ren balioa $\varnothing$ dela, eta abar adierazten da.

- q_0 hasierako egoera da. Hasierako egoera beti bakarra izango da.
- $Y = \{q_1\}$.

Orain $(\{q_0\}, aaa)$ konfigurazioari dagokion konputazioa garatuko dugu urratsez urrats:

1. urratsa: $\nu^*(\{q_0\}, aaa) = \nu^*(\nu(q_0, a), aa) = \nu^*(\{q_1\}, aa)$

2. urratsa: $\nu^*(\{q_1\}, aa) = \nu^*(\nu(q_1, a), a) = \nu^*(\{q_1\}, a)$

3. urratsa: $\nu^*(\{q_1\}, a) = \nu^*(\{q_1\}, a\varepsilon) = \nu^*(\nu(q_1, a), \varepsilon) = \nu^*(\{q_1\}, \varepsilon)$

4. urratsa: $\nu^*(\{q_1\}, \varepsilon) = \{q_1\}$

Konputazioa $\{q_1\}$ multzoan bukatzen denez eta multzo horretan gutxienez elementu bat, q_1 elementua, Y multzokoa denez, $\{q_1\} \cap Y \neq \varnothing$ betetzen da eta ondorioz hitz hori adibide honetako lengoaiakoa da (ez da hutsa eta a sinboloaren errepikapenez eratutako dago).

Konputazio hau konfigurazio ez deterministez osatutako sekuentzia bezala honela adieraz daiteke:

$$(\{q_0\}, aaa)$$
$$|$$
$$(\{q_1\}, aa)$$
$$|$$
$$(\{q_1\}, a)$$
$$|$$
$$(\{q_1\}, \varepsilon)$$

Jarraian $(\{q_0\}, abca)$ konfigurazioari dagokion konputazioa garatuko dugu urratsez urrats:

1. urratsa: $\nu^*(\{q_0\}, abca) = \nu^*(\nu(q_0, a), bca) = \nu^*(\{q_1\}, bca)$

2. urratsa: $\nu^*(\{q_1\}, bca) = \nu^*(\nu(q_2, b), ca) = \nu^*(\varnothing, ca)$

3. urratsa: $\nu^*(\varnothing, ca) = \varnothing$

Bukaerako egoera-multzoa hutsa denez ($\varnothing$) eta $\varnothing$ multzoan Y multzoko elementurik ez dagoenez, hitza ez da adibide honetako lengoiakoa.

Konputazio hori grafikoki honela adieraz daiteke:

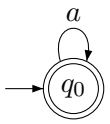
$$\begin{array}{c}(\{q_0\}, abca)\\ \vert \\ (\{q_1\}, bca)\\ \vert \\ (\varnothing, bca)\end{array}$$

3.3.2 Adibidea: b-rik eta c-rik ez duten hitzez osatutako lengoaiari dagokion AFEDa

Adibide honetako AFEDak (3.18 irudian begiratu) a, b eta c sinboloak izan ditzakeen hitz bat jasoko du sarrera bezala. Emandako hitza b-rik eta c-rik ez duen hitz bat al den erabaki beharko du AFEDak. Horrela, b-rik eta c-rik ez badu "Bai" erantzun beharko du eta bestela "Ez" erantzun beharko du. Lengoaiaren definizio formala honako hau da:

$$\{w \mid w \in A^* \wedge |w|_a = |w|\}$$

Adibide honetako AFEDak egoera bakarra du. Hasierako egoera q_0 da eta ondorioz, hitz bat irakurtzen hasterakoan q_0 egoeran egongo da. Lehenengo karakterea a baldin bada, AFEDa q_0 egoeran geratuko da. Lehenengo karakterea b edo c baldin bada, AFEDak ezingo du inora joan eta ezta q_0-n geratu ere. q_0 egoeran mantenduz hitz osoa irakurtzea lortzen bada, badakigu karaktere-kateak ez duela b-rik eta c-rik eta "Bai" erantzuna itzuliko da. Bestalde, karaktere-katea irakurtzea ez bada lortzen, badakigu karaktere-kateak baduela b edo c-ren bat eta "Ez" erantzuna itzuliko da.



3.18 Irudia: b-rik eta c-rik ez duten hitzez osatutako lengoaiari dagokion AFEDa.

Jarraian AFEDa boskote bezala definituko da.

$$(Q, A, \nu, q_0, Y)$$

Bost osagaiak honako hauek dira:

- $Q = \{q_0, q_1\}$
- $A = \{a, b, c\}$
- $\nu : Q \times A \to 2^Q$ jarraian datorren taularen bidez definitutako trantsizio funtzioa da:

ν	a	b	c
q_0	$\{q_0\}$	$\varnothing$	$\varnothing$

Taularen bidez $\nu(q_0, a)$-ren balioa $\{q_0\}$ dela, $\nu(q_0, b)$-ren balioa $\varnothing$ dela, eta abar adierazten da.

- q_0 hasierako egoera da. Hasierako egoera beti bakarra da.
- $Y = \{q_0\}$.

Orain $(\{q_0\}, aaa)$ konfigurazioari dagokion konputazioa garatuko dugu urratsez urrats:

1. urratsa: $\nu^*(\{q_0\}, aaa) = \nu^*(\nu(q_0, a), aa) = \nu^*(\{q_0\}, aa)$

2. urratsa: $\nu^*(\{q_0\}, aa) = \nu^*(\nu(q_0, a), a) = \nu^*(\{q_0\}, a)$

3. urratsa: $\nu^*(\{q_0\}, a) = \nu^*(\{q_0\}, a\varepsilon) = \nu^*(\nu(q_0, a), \varepsilon) = \nu^*(\{q_0\}, \varepsilon)$

4. urratsa: $\nu^*(\{q_0\}, \varepsilon) = \{q_0\}$

Konputazioa $\{q_0\}$ multzoan bukatzen denez eta multzo horretan gutxienez elementu bat, q_0 elementua, Y multzokoa denez, $\{q_0\} \cap Y \neq \varnothing$ betetzen da eta ondorioz hitz hori adibide honetako lengoaiakoa da (ez du ez b-rik eta ez c-rik).

Konputazio hori grafikoki honako konfigurazio ez deterministen zerrenda bezala adieraz daiteke:

$$(\{q_0\}, aaa)$$
$$|$$
$$(\{q_0\}, aa)$$
$$|$$
$$(\{q_0\}, a)$$
$$|$$
$$(\{q_0\}, \varepsilon)$$

Jarraian $(\{q_0\}, abca)$ konfigurazio ez deterministari dagokion konputazioa garatuko dugu urratsez urrats:

1. urratsa: $\nu^*(\{q_0\}, abca) = \nu^*(\nu(q_0, a), bca) = \nu^*(\{q_0\}, bca)$

2. urratsa: $\nu^*(\{q_0\}, bca) = \nu^*(\nu(q_0, b), ca) = \nu^*(\varnothing, ca)$

3. urratsa: $\nu^*(\varnothing, ca) = \varnothing$

Konputazioa multzo hutsean ($\varnothing$) bukatzen denez eta multzo hutsean Y multzokoa den elementurik ez dagoenez, $\varnothing \cap Y \neq \varnothing$ betetzen da eta ondorioz hitz hori ez da adibide honetako lengoaiakoa (badu b-ren bat edo c-ren bat).

Konputazio hori grafikoki honako konfigurazio ez deterministen zerrenda bezala adieraz daiteke:

$$(\{q_0\}, abca)$$
$$|$$
$$(\{q_0\}, bca)$$
$$|$$
$$(\varnothing, ca)$$

3.3.3 Adibidea: c-rik ez duten eta a eta b-ren agerpenak orden horretan tartekatuta dituzten hitzez osatutako lengoaiari dagokion AFEDa

Adibide honetako AFEDak (3.19 irudia begiratu) c-rik ez duten eta a eta b orden horretan tartetakuta dituzten $A = \{a, b, c\}$ alfabetoaren gainean definitutako hitzentzat "Bai" erantzungo du. Beraz AFED honi dagokion lengoaiako hitzetan ezin dira bi a jarraian agertu eta ezta bi b ere. Hitz hutsa lengoaiakoa da eta hutsak ez diren hitzen kasuan lehenengo sinboloak a izan behar du. Hala ere azkeneko sinboloa a edo b izan daiteke. Beraz ε, $abab$ eta aba hitzentzat AFEDak "Bai" erantzungo luke, baina $abca$, $bababa$, aaa, cc eta aab hitzentzat "Ez" erantzungo luke. Lengoaiaren definizio formala honako hau da:

$$\{w \mid w \in A^* \wedge \exists k(k \geq 0 \wedge (w = (ab)^k \vee w = ((ab)^k)a))\}$$

Lengoaia hori beste era honetara ere defini daiteke:

$$\begin{array}{ll}\{w \mid & w \in A^* \wedge \forall k((1 \leq k \leq |w| \wedge k \bmod 2 = 1) \rightarrow w(k) = a) \\ & \wedge \forall k((1 \leq k \leq |w| \wedge k \bmod 2 = 0) \rightarrow w(k) = b)\}\end{array}$$

AFEDak bi egoera ditu, q_0 eta q_1. Hasierako egoera q_0 da eta hitz osoa irakurtzea lortzen bada, bai q_0 egoeran bukatzen bada eta bai q_1 egoeran bukatzen bada, AFEDaren erantzuna "Bai" izango da.

Jarraian AFDa boskote bezala definituko da.

$$(Q, A, \nu, q_0, Y)$$

Bost osagaiak honako hauek dira:

- $Q = \{q_0, q_1, q_2\}$
- $A = \{a, b, c\}$
- $\nu : Q \times A \rightarrow 2^Q$ jarraian datorren taularen bidez definitutako trantsizio funtzioa da:

ν	a	b	c
q_0	$\{q_1\}$	$\varnothing$	$\varnothing$
q_1	$\{q_0\}$	$\varnothing$	$\varnothing$

  Taula horren bidez $\delta(q_0, a)$-ren balioa $\{q_1\}$ dela, $\delta(q_0, b)$-ren balioa $\varnothing$ dela, eta abar adierazten da.

- q_0 hasierako egoera da. Hasierako egoera beti bakarra da.
- $Y = \{q_0, q_1\}$. Adibide honetan hasierako egoera Y multzoan ager daitekeela ikus dezakegu.

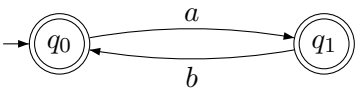
Orain $(\{q_0\}, abab)$ konfigurazio ez deterministari dagokion konputazioa garatuko dugu urratsez urrats:

1. urratsa: $\nu^*(\{q_0\}, abab) = \nu^*(\nu(q_0, a), bab) = \nu^*(\{q_1\}, bab)$

2. urratsa: $\nu^*(\{q_1\}, bab) = \nu^*(\nu(q_1, b), ab) = \nu^*(\{q_0\}, ab)$

3. urratsa: $\nu^*(\{q_0\}, ab) = \nu^*(\nu(q_0, a), b) = \nu^*(\{q_1\}, b)$

4. urratsa: $\nu^*(\{q_1\}, b) = \nu^*(\nu(q_1, b), \varepsilon) = \nu^*(\{q_0\}, \varepsilon)$



3.19 Irudia: c-rik ez eta a eta b, orden horretan, tartekatuta dituzten hitzen lengoaiari dagokion AFEDa.

5. urratsa: $\nu^*(\{q_0\}, \varepsilon) = \{q_0\}$

Konputazioa $\{q_0\}$ multzoan bukatzen denez eta multzo horretan gutxienez elementu bat, q_0 elementua, Y multzokoa denez, $\{q_0\} \cap Y \neq \varnothing$ betetzen da eta ondorioz hitz hori adibide honetako lengoaiakoa da.

Konputazio hori grafikoki honako konfigurazio ez deterministen zerrenda bezala adieraz daiteke:

$$(\{q_0\}, abab)$$
$$|$$
$$(\{q_1\}, bab)$$
$$|$$
$$(\{q_0\}, ab)$$
$$|$$
$$(\{q_1\}, b)$$
$$|$$
$$(\{q_0\}, \varepsilon)$$

Jarraian $(\{q_0\}, abc)$ konfigurazioari dagokion konputazioa garatuko dugu urratsez urrats:

1. urratsa: $\nu^*(\{q_0\}, abc) = \nu^*(\nu(q_0, a), bc) = \nu^*(\{q_1\}, bc)$

2. urratsa: $\nu^*(\{q_1\}, bc) = \nu^*(\nu(q_1, b), c) = \nu^*(\{q_0\}, c)$

3. urratsa: $\nu^*(\{q_0\}, c) = \nu^*(\nu(q_0, c), \varepsilon) = \nu^*(\varnothing, \varepsilon)$

4. urratsa: $\nu^*(\varnothing, \varepsilon) = \varnothing$

Konputazioa multzo hutsean ($\varnothing$) bukatzen denez eta multzo hutsean Y multzokoa den elementurik ez dagoenez, $\varnothing \cap Y \neq \varnothing$ betetzen da eta ondorioz hitz hori ez da adibide honetako lengoaiakoa.

Konputazio hori grafikoki honako konfigurazio ez deterministen zerrenda bezala adieraz daiteke:

$$(\{q_0\}, abc)$$
$$|$$
$$(\{q_1\}, bc)$$
$$|$$
$$(\{q_0\}, c)$$
$$|$$
$$(\varnothing, \varepsilon)$$

3.3.4 Adibidea: 10-rekin bukatzen diren hitzez osatutako lengoaiari dagokion AFEDa

Adibide honetako AFEDak (3.20 irudia begiratu) 10 katearekin bukatzen diren $A = \{0, 1\}$ alfabetoaren gainean definitutako hitzentzat "Bai" erantzungo du. Beraz, esate baterako 0011010 eta 0010 hitzen kasuan "Bai" erantzungo du, baina 000, 1101, ε eta 0 hitzen kasuan "Ez" erantzungo du. Lengoaiaren definizio formala honako hau da:

$$\{w \mid w \in A^* \wedge \exists u(u \in A^* \wedge w = u10)\}$$

Lengoaia hori beste era honetara ere defini daiteke:

$$\{w \mid w \in A^* \wedge |w| \geq 2 \wedge w(|w| - 1) = 1 \wedge w(|w|) = 0\}$$

AFED-ak hiru egoera ditu, q_0, q_1 eta q_2. Hasierako egoera q_0 da eta hitz osoa irakurtzea lortzen bada eta gainera q_2 egoeran bukatzen bada erantzuna "Bai" izango da.

Jarraian AFEDa boskote bezala definituko da.

$$(Q, A, \nu, q_0, Y)$$

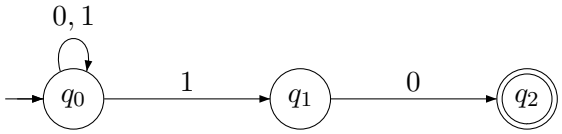
AFEDaren bost osagaiak honako hauek dira:

- $Q = \{q_0, q_1, q_2\}$
- $A = \{0, 1\}$
- $\nu : Q \times A \to 2^Q$ jarraian datorren taularen bidez definitutako trantsizio funtzioa da:

ν	0	1
q_0	$\{q_0\}$	$\{q_0, q_1\}$
q_1	$\{q_2\}$	$\varnothing$
q_2	$\varnothing$	$\varnothing$

  Taula horren bidez $\nu(q_0, 0)$-ren balioa $\{q_0\}$ dela, $\nu(q_0, 1)$-ren balioa $\{q_0, q_1\}$ dela, eta abar adierazten da.

- q_0 hasierako egoera da. Hasierako egoera beti bakarra da.
- $Y = \{q_2\}$.



3.20 Irudia: 10 azpikatearekin bukatzen diren hitzen lengoaiari dagokion AFED-a.

Hona hemen $(\{q_0\}, 0010)$ konfigurazio ez deterministari dagokion urratsez urrats garatutako konputazioa:

1. urratsa: $\nu^*(\{q_0\}, 0010) = \nu^*(\nu(q_0, 0), 010) = \nu^*(\{q_0\}, 010)$

2. urratsa: $\nu^*(\{q_0\}, 010) = \nu^*(\nu(q_0, 0), 10) = \nu^*(\{q_0\}, 10)$

3. urratsa: $\nu^*(\{q_0\}, 10) = \nu^*(\nu(q_0, 1), 0) = \nu^*(\{q_0, q_1\}, 0)$

4. urratsa: $\nu^*(\{q_0, q_1\}, 0) = \nu^*(\nu(q_0, 0) \cup \nu(q_1, 0), \varepsilon) = \nu^*(\{q_0\} \cup \{q_2\}, \varepsilon) = \nu^*(\{q_0, q_2\}, \varepsilon)$

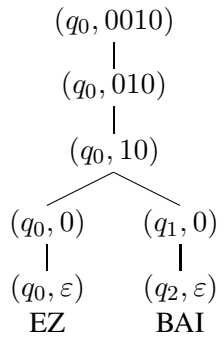
5. urratsa: $\nu^*(\{q_0, q_2\}, \varepsilon) = \{q_0, q_2\}$

Konputazioa $\{q_0, q_2\}$ multzoan bukatzen denez eta multzo horretan gutxienez elementu bat, q_2 elementua, Y multzokoa denez, $\{q_0, q_2\} \cap Y \neq \varnothing$ betetzen da eta ondorioz hitz hori adibide honetako lengoaiakoa da.

Konputazio hori grafikoki honako konfigurazio ez deterministen zerrenda bezala adieraz daiteke:

$$(\{q_0\}, 0010)$$
$$|$$
$$(\{q_0\}, 010)$$
$$|$$
$$(\{q_0\}, 10)$$
$$|$$
$$(\{q_0, q_1\}, 0)$$
$$|$$
$$(\{q_0, q_2\}, \varepsilon)$$

Konfigurazio deterministez eratutako zuhaitz bezala ere adieraz daieteke konputazio hori:



Gutxienez adarretako bat Y multzokoa den egoera batez eta ε hitz hutsaz eratutako konfigurazio determinista batean bukatzen denez, 0010 hitza adibide honetako lengoaiakoa da.

Beste adibide bat izateko, $(\{q_0\}, 111)$ konfigurazioari dagokion urratsez urrats garatutako konputazioa honako hau da:

1. urratsa: $\nu^*(\{q_0\}, 111) = \nu^*(\nu(q_0, 1), 11) = \nu^*(\{q_0, q_1\}, 11)$

2. urratsa: $\nu^*(\{q_0, q_1\}, 11) = \nu^*(\nu(q_0, 1) \cup \nu(q_1, 1), 1) = \nu^*(\{q_0, q_1\} \cup \varnothing, 1) = \nu^*(\{q_0, q_1\}, 1)$

3. urratsa: $\nu^*(\{q_0, q_1\}, 1) = \nu^*(\nu(q_0, 1) \cup \nu(q_1, 1), \varepsilon) = \nu^*(\{q_0, q_1\} \cup \varnothing, \varepsilon) = \nu^*(\{q_0, q_1\}, \varepsilon)$

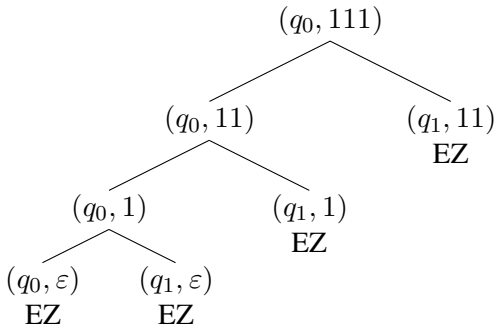
4. urratsa: $\nu^*(\{q_0, q_1\}, \varepsilon) = \{q_0, q_1\}$

Konputazioa $\{q_0, q_1\}$ multzoan bukatzen denez eta multzo horretan Y multzokoa den elementurik ez dagoenez, $\{q_0, q_1\} \cap Y = \varnothing$ betetzen da eta ondorioz hitz hori adibide honetako lengoaiakoa ez da.

Konputazio hori honako konfigurazio ez deterministen zerrenda bezala ipini daiteke:

$$(\{q_0\}, 111)$$
$$|$$
$$(\{q_0, q_1\}, 11)$$
$$|$$
$$(\{q_0, q_1\}, 1)$$
$$|$$
$$(\{q_0, q_1\}, \varepsilon)$$

Konfigurazio deterministez eratutako zuhaitz bezala adieraziz honela geratuko litzateke konputazio hori:



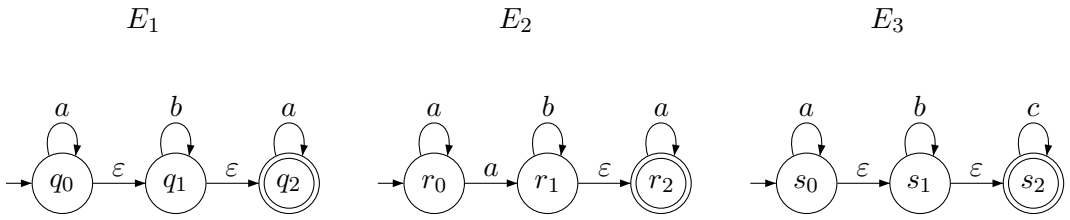
Y multzokoa den egoera batez eta ε hitz hutsaz eratutako konfigurazio determinista batean bukatzen den adarrik ez dagoenez, 111 hitza adibide honetako lengoaiakoa ez da.

3.4 ε trantsizioak dituzten AFED-ak

Automata finituen barruan, hirugarren mota bat bereiz dezakegu: ε trantsizioak dituzten AFED-ak. Automata hauek ε-AFED bezala laburtuko dira.

Automata hauetan trantsizio hutsak edo ε bidezko trantsizioak ditugu. Grafikoki adierazten direnean, trantsizio hutsei dagozkien gezietan ε ipini ohi da.

Jarraian hiru adibide ditugu, E_1, E_2 eta E_3 ε-AFED-ak



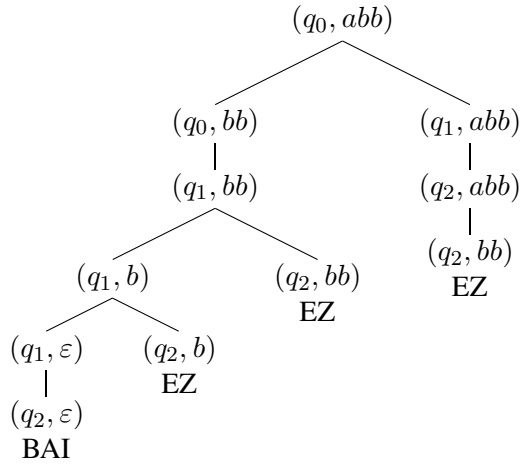
ε trantsizioen bidez egoera batetik bestera joan gaitezke sinbolorik irakurri gabe.

E_1 ε-AFED-ari dagokion lengoaia

$$\{w \mid w \in A^* \wedge \exists u, v, x(u \in A^* \wedge v \in A^* \wedge x \in A^* \wedge |u|_a = |u| \wedge |v|_b = |v| \wedge |x|_a = |x| \wedge w = uvx)\}$$

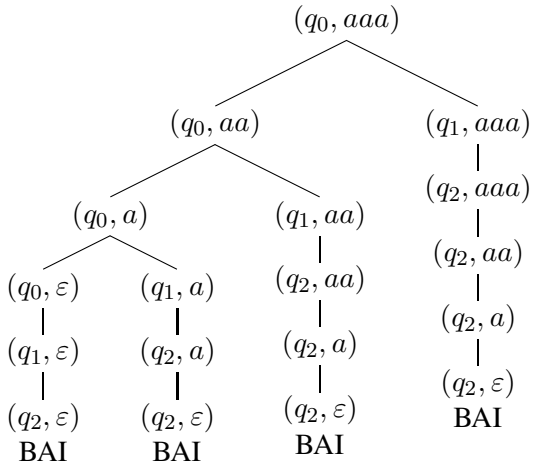
da.

abb hitza lengoaia horretakoa dela erabakitzeko jarraituko litzatekeen den prozesua honela adieraz dezakegu zuhaitz baten bidez:



Zuhaitz horretan garbi ikus daiteke urrats batzuetan ez dela sinbolorik irakurtzen edo kontsumitzen. Hori ε trantsizioak daudelako da.

aaa hitza lengoaia horretakoa dela erabakitzeko jarraituko litzatekeen den prozesua honela adieraz dezakegu trantsizio deterministez osatutako zuhaitz baten bidez:



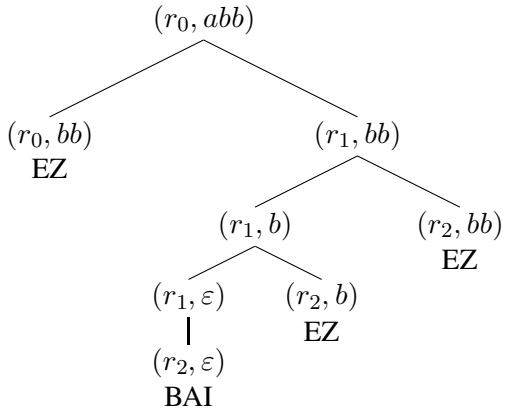
Zuhaitz horretan ere garbi ikus daiteke urrats batzuetan ez dela sinbolorik irakurtzen edo kontsumitzen. Hori ε trantsizioak daudelako da.

E_2 ε-AFED-ari dagokion lengoaiaren honela defini daiteke:

$$\{w \mid w \in A^* \wedge \exists u, v, x(u \in A^* \wedge v \in A^* \wedge x \in A^* \wedge |u|_a = |u| \wedge |v|_b = |v| \wedge |x|_a = |x| \wedge w = auvx)\}$$

da.

abb hitza lengoaia horretakoa dela erabakitzeko jarraitzen den prozesua honela adieraz dezakegu zuhaitz baten bidez:

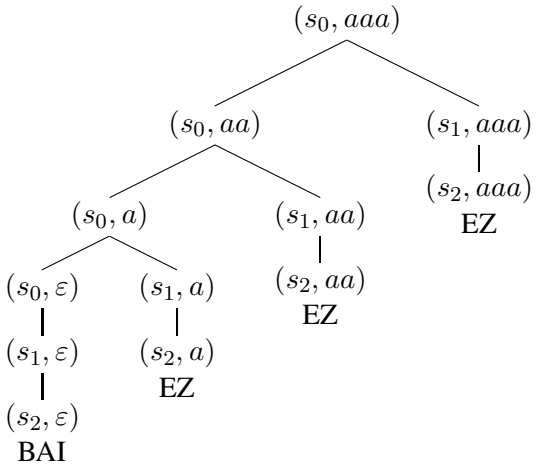


Zuhaitz honetan ere urrats batzuetan ez da sinbolorik irakurtzen edo kontsumitzen. Hori ε trantsizioak daudelako da.

E_3 ε-AFED-ari dagokion lengoaia honela defini daiteke:

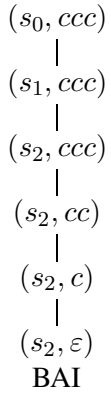
$$\{w \mid w \in A^* \wedge \exists u, v, x(u \in A^* \wedge v \in A^* \wedge x \in A^* \wedge |u|_a = |u| \wedge |v|_b = |v| \wedge |x|_c = |x| \wedge w = uvx)\}$$

aaa hitza lengoaia horretakoa dela erabakitzeko jarraituko litzatekeen prozesua honela adieraz dezakegu konfigurazio deterministez eratutako zuhaitz baten bidez:



Zuhaitz horretan erakusten den bezala, urrats batzuetan ez da sinbolorik kontsumitzen edo irakurtzen, hau da, egoeraz aldatu arren hitz berarekin jarraitzen da. Hori ε trantsizioengatik gertatzen da.

ccc hitza lengoaia horretakoa dela erabakitzeko jarraituko litzatekeen prozesua honela adieraz dezakegu konfigurazio deterministez eratutako zuhaitz baten bidez:



ε trantsizioak direla eta, zuhaitz honetan ere urrats batzuetan ez da sinbolorik kontsumitzen edo irakurtzen, hau da, egoeraz aldatu arren hitz berarekin jarraitzen da.

3.4.1 ε trantsizioak dituzten Automata Finitu Ez Deterministen definizio formala (ε-AFED-ak)

ε trantsizioak dituen E Automata Finitu Ez Determinista bat honelako boskote bat da:

$$(Q, A, \lambda, q_0, Y)$$

Hor:

- Q egoren multzo finitua da: $q_0, q_1, q_2, q_3, \ldots$
- A alfabetoa da.
- λ [1] ε trantsizioak dituen trantsizio-funtzio ez determinista da. Q multzoko egoera bat eta $A \cup \{\varepsilon\}$ multzoko egoera bat emanda, λ trantsizio funtzioak zein egoeretara edo egoera-multzora pasatu behar den adierazten du. Beraz, λ funtzioaren mota honako hau izango da:

[1] *λ lambda letra grekoa da*

$$\lambda : Q \times (A \cup \{\varepsilon\}) \rightarrow 2^Q$$

Hor, 2^Q-ren bidez $Q \rightarrow 2$ erako funtzio denen multzoa adierazten da, $2 = \{0, 1\}$ izanda. Gainera 2 multzoan 0 balioak $False$ adierazten du eta 1 balioak $True$ adierazten du. Beste era batera esanda, 2^Q multzoa Q multzoaren azpimultzo denez osatutako multzoa da.

- q_0 hasierako egoera da. Hasierako egoera beti bakarra izaten da.
- Y osagaia "Bai" erantzuten duten egoerez osatutako multzoa da. Ondorioz, Y multzoa Q multzoaren azpimultzoa da, hau da, $Y \subseteq Q$.

ε-AFED-etan egoera batetik beste batera sinbolorik kontsumitu gabe igaro gaitezke bi egoera horien artean trantsizio hutsa, hau da, ε sinboloa duen trantsizioa baldin badago. Eta hori da AFED eta ε-AFED-en arteko arteko oinarrizko desberdintasuna.

AFED-etan bezala, ε-AFED-etan ere **konfigurazio ez deterministа** bat (S, w) erako bikote bat da, hor S osagaia Q multzoaren azpimultzo bat (eta ondorioz, 2^Q multzoko elementu bat) eta w osagaia A^* multzoko elementu bat izanda. Beraz, ε-AFED-etan ere konfigurazio ez deterministak $2^Q \times A^*$ motakoak dira. Ondorioz, AFED-etan eta ε-AFED-etan erabiltzen diren konfigurazio ez deterministen artean ez dago desberdintasunik.

ε-AFED batek burututako **konputazioa** λ trantsizio funtzioan oinarritzen den λ^* funtzioaren bidez definitzen da. λ^* funtzioaren mota honako hau da:

$$\lambda^* : 2^Q \times A^* \rightarrow 2^Q$$

δ eta ν trantsizio funtzioekin gertatzen den bezala, λ funtzioak urrats bakar bat adierazten du eta, δ^* eta ν^* funtzioekin gertatzen den bezala, λ^* funtzioak egoera-multzo batetik abiatuz hitz oso bat irakurriz zein egoeretara iritsi gaitezkeen adierazten du. Beraz, $\lambda^*(\{q_{j_1}, q_{j_2}, \ldots, q_{j_k}\}, w)$ espresioaren esanahia $\{q_{j_1}, q_{j_2}, \ldots, q_{j_k}\}$ egoeretatik w hitzeko sinboloak eta ε erabiliz zein egoeretara iritsi gaitezkeen da.

ε-AFED-etan sinbolorik irakurri gabe aurrera egin daitekeenez (egoeraz alda gaitezkeenez) aurreratze berezi hori Itxidura-ε bezala adieraziko dugun funtzioaren bidez adieraziko dugu.

Itxidura-ε funtzioaren mota honako hau da:

$$\text{Itxidura-}\varepsilon :: Q \times \underbrace{(Q \times (A \cup \{\varepsilon\}) \rightarrow 2^Q)}_{\text{hau } \lambda\text{-ren mota da}} \rightarrow 2^Q$$

Itxidura-ε funtzioaren definizio formala honako hau da:

$$\text{Itxidura-}\varepsilon(q, \lambda) = \begin{cases} \{q\} & \lambda(q, \varepsilon) = \varnothing \text{ betetzen bada} \\ \{q\} \cup \bigcup_{q_{j_\ell} \in \lambda(q,\varepsilon)} \text{Itxidura-}\varepsilon(q_{j_\ell}, \lambda) & \text{bestela} \end{cases}$$

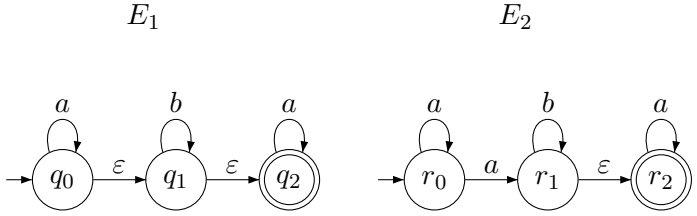
$\bigcup_{q_{j_\ell} \in \lambda(q,\varepsilon)} \text{Itxidura-}\varepsilon(q_{j_\ell}, \lambda)$ espresioaren esanahia honako hau da: $\lambda(q, \varepsilon) = \{q_{j_1}, q_{j_2}, \ldots, q_{j_k}\}$ baldin bada, hau da, sinbolorik irakurri gabe q egoeratik urrats bakar batean irisgarriak diren egoerak $\{q_{j_1}, q_{j_2}, \ldots, q_{j_k}\}$ baldin badira, orduan errekurtsiboki prozesua errepikatu egin beharko da $\{q_{j_1}, q_{j_2}, \ldots, q_{j_k}\}$ multzoko egoera bakoitzetik irisgarriak diren egoerak kalkulatzeko. Beraz, $\lambda(q, \varepsilon) = \{q_{j_1}, q_{j_2}, \ldots, q_{j_k}\}$ baldin bada:

$$\bigcup_{q_{j_\ell} \in \lambda(q,\varepsilon)} \text{Itxidura-}\varepsilon(q_{j_\ell}, \lambda) = \text{Itxidura-}\varepsilon(q_{j_1}, \lambda) \cup \text{Itxidura-}\varepsilon(q_{j_2}, \lambda) \cup \ldots \cup \text{Itxidura-}\varepsilon(q_{j_k}, \lambda)$$

Itxidura-ε funtzioa multzoetara honela hedatzen da:

$$\text{Itxidura-}\varepsilon^*(S, \lambda) = \bigcup_{q \in S} \text{Itxidura-}\varepsilon(q, \lambda)$$

Jarraian atal honel hasieran aurkeztutako E_1 eta E_2 ε-AFED-ak berriro hartuko ditugu:



E_1-en $\lambda(q_0, \varepsilon) = \{q_1\}$ da eta Itxidura-$\varepsilon(q_0, \lambda) = \{q_0, q_1, q_2\}$. Bestalde, $\lambda(q_1, \varepsilon) = \{q_2\}$ da eta Itxidura-$\varepsilon(q_1, \lambda) = \{q_1, q_2\}$ da. Azkenik, $\lambda(q_2, \varepsilon) = \varnothing$ da eta Itxidura-$\varepsilon(q_2, \lambda) = \{q_2\}$ da. Gainera Itxidura-$\varepsilon^*(\{q_1, q_2\}, \lambda) = \{q_1, q_2\}$ da eta Itxidura-$\varepsilon^*(\{q_0, q_2\}, \lambda) = \{q_0, q_1, q_2\}$ da.

E_2 hartzen badugu, $\lambda(r_0, \varepsilon) = \varnothing$ da eta Itxidura-$\varepsilon(r_0, \lambda) = \{r_0\}$. Beste aldetik, $\lambda(r_1, \varepsilon) = \{r_2\}$ da eta Itxidura-$\varepsilon(r_1, \lambda) = \{r_1, r_2\}$ da. Bukatzeko, $\lambda(r_2, \varepsilon) = \varnothing$ da eta Itxidura-$\varepsilon(r_2, \lambda) = \{r_2\}$ da. Gainera, Itxidura-$\varepsilon^*(\{r_1, r_2\}, \lambda) = \{r_1, r_2\}$ da eta Itxidura-$\varepsilon^*(\{r_0, r_2\}, \lambda) = \{r_0, r_2\}$.

λ^*-ren definizio formala honako hau da:

- $\lambda^*(S, \varepsilon) =$ Itxidura-$\varepsilon^*(S, \lambda)$
- $\lambda^*(\varnothing, w) = \varnothing$
- $\lambda^*(S, \alpha w) = \lambda^*(\bigcup_{q_j \in \text{Itxidura-}\varepsilon^*(S,\lambda)} \lambda(q_j, \alpha), w)$

hor S egoera-multzo bat da, α A-ko sinbolo bat da eta w A^*-ko hitz bat da.

Definizio horren bidez λ^*-ren jokabidea zein den adierazten da. Sarrera bezala emandako hitza ε hitz hutsa baldin bada, S oraingo egoeretan eta S-ko egoera horietatik sinbolorik gabe irisgarriak diren egoeretan geratuko gara. Sarrera bezala emandako S multzoa hutsa baldin bada, hau da, $\varnothing$, orduan ez gaude egoeraren batean eta ondorioz ezingo dugu beste egoeraren batera iritsi, eta hori $\varnothing$ egoera-multzo hutsera iritsiko garela esanez adierazten da. Bestalde, S multzo hutsa ez bada eta sarrerako w hitza hutsa ez bada eta bere lehenengo osagaia (A-koa izango den sinboloa) α baldin bada, ε trantsizioen bidez S multzotik irisgarriak diren egoerak hartu (Itxidura-$\varepsilon^*(S, \lambda)$) eta egoera horietako bakoitzari eta α sinboloari λ aplikatuko zaie. Egoera-multzo berrian kokatu ondoren, w hitzeko gainontzeko sinboloak irakurtzen jarraitu beharko da.

Adibidez, S multzoa $\{q_2, q_4, q_5, q_7\}$ dela eta Itxidura-$\varepsilon^*(S, \lambda)$ multzoa $\{q_2, q_4, q_5, q_6, q_7, q_{10}\}$ dela kontsideratzen badugu,

$$\lambda^*(\{q_2, q_4, q_5, q_7\}, \alpha w) = \lambda^*(\lambda(q_2, \alpha) \cup \lambda(q_4, \alpha) \cup \lambda(q_5, \alpha) \cup \lambda(q_6, \alpha) \cup \lambda(q_7, \alpha) \cup \lambda(q_{10}, \alpha), w)$$

Hor $\lambda(q_2, \alpha)$ espresioak q_2 egoeratik α sinboloaren bidez irisgarriak diren egoerak zein diren adierazten du, $\lambda(q_4, \alpha)$ espresioak q_4 egoeratik α sinboloaren bidez irisgarriak diren egoerak zein diren adierazten du eta abar.

Beste adibide bat erabiliz, oraindik gehiago zehaztuko dugu. Demagun $A = \{a, b, c\}$ alfabetoa daukagula eta une honetako konfigurazioa $(\{q_1, q_4\}, caab)$ dela eta λ funtzioa honela definituta dagoela jarraian aipatzen diren kasuetarako:

- $\lambda(q_1, c) = \{q_2, q_6\}$

- $\lambda(q_1, \varepsilon) = \{q_8\}$
- $\lambda(q_2, a) = \varnothing$
- $\lambda(q_3, a) = \{q_7\}$
- $\lambda(q_4, c) = \{q_3, q_5, q_6\}$
- $\lambda(q_5, a) = \varnothing$
- $\lambda(q_5, b) = \{q_6\}$
- $\lambda(q_6, b) = \{q_2, q_4\}$
- $\lambda(q_6, a) = \{q_5, q_6\}$
- $\lambda(q_7, a) = \{q_7\}$
- $\lambda(q_7, b) = \{q_1, q_2\}$
- $\lambda(q_8, c) = \{q_{10}, q_{11}\}$
- $\lambda(q_{10}, \varepsilon) = \{q_{12}\}$
- $\lambda(q_{10}, a) = \{q_{10}\}$
- $\lambda(q_{10}, b) = \{q_{10}\}$
- $\lambda(q_{11}, a) = \varnothing$
- $\lambda(q_{12}, a) = \{q_{12}\}$
- $\lambda(q_{12}, b) = \{q_{13}\}$

Urratsez urrats $\lambda^*(\{q_1, q_4\}, caab)$ konputazioa honela kalkulatuko litzateke:

- $caab$ hitza hutsa ez denez eta Itxidura-$\varepsilon^*(\{q_1, q_4\}, \lambda) = \{q_1, q_4, q_8\}$ denez,
  $\lambda^*(\{q_1, q_4\}, caab) = \lambda^*(\lambda(q_1, c) \cup \lambda(q_4, c) \cup \lambda(q_8, c), aab)$
  betetzen da.
- $\lambda(q_1, c) = \{q_2, q_6\}$ denez, $\lambda(q_4, c) = \{q_3, q_5, q_6\}$ eta $\lambda(q_8, c) = \{q_{10}, q_{11}\}$ betetzen direnez,
  $\lambda^*(\lambda(q_1, c) \cup \lambda(q_4, c) \cup \lambda(q_8, c), aab)$
  espresioa honela gelditzuko da $\lambda^*(\{q_2, q_3, q_5, q_6, q_{10}, q_{11}\}, aab)$.
- $\lambda(q_8, c)$ espresioak urrats bakar bat adierazten duenez, bere balioa $\{q_{10}, q_{11}\}$ da, eta ez $\{q_{10}, q_{11}, q_{12}\}$ nahiz eta q_{12} irisgarria izan q_{10} egoeratik ε trantsizio baten bidez.
- aab hitza hutsa ez denez eta Itxidura-$\varepsilon^*(\{q_2, q_3, q_5, q_6, q_{10}, q_{11}\}, \lambda) = \{q_2, q_3, q_5, q_6, q_{10}, q_{11}, q_{12}\}$ denez, honako hau daukagu:
  $\lambda^*(\{q_2, q_3, q_5, q_6, q_{10}, q_{11}\}, aab) =$
  $= \lambda^*(\lambda(q_2, a) \cup \lambda(q_3, a) \cup \lambda(q_5, a) \cup \lambda(q_6, a) \cup \lambda(q_{10}, a) \cup \lambda(q_{11}, a) \cup \lambda(q_{12}, a), ab).$
- $\lambda(q_2, a) = \varnothing$, $\lambda(q_3, a) = \{q_7\}$, $\lambda(q_5, a) = \varnothing$, $\lambda(q_6, a) = \{q_5, q_6\}$, $\lambda(q_{10}, a) = \{q_{10}\}$, $\lambda(q_{11}, a) = \varnothing$ eta $\lambda(q_{12}, a) = \{q_{12}\}$ direla kontuan hartuz,
  $\lambda^*(\lambda(q_2, a) \cup \lambda(q_3, a) \cup \lambda(q_5, a) \cup \lambda(q_6, a) \cup \lambda(q_{10}, a) \cup \lambda(q_{11}, a) \cup \lambda(q_{12}, a), ab)$
  espresiotik $\lambda^*(\{q_5, q_6, q_7, q_{10}, q_{12}\}, ab)$ espresioa lortzen da.

- ab hitza hutsa ez denez eta Itxidura-$\varepsilon^*(\{q_5, q_6, q_7, q_{10}, q_{12}\}, \lambda) = \{q_5, q_6, q_7, q_{10}, q_{12}\}$ denez, honako hau daukagu:
  $\lambda^*(\{q_5, q_6, q_7, q_{10}, q_{12}\}, ab) = \lambda^*(\lambda(q_5, a) \cup \lambda(q_6, a) \cup \lambda(q_7, a) \cup \lambda(q_{10}, a) \cup \lambda(q_{12}, a), b)$.

- $\lambda(q_5, a) = \varnothing$, $\lambda(q_6, a) = \{q_5, q_6\}$, $\lambda(q_7, a) = \{q_7\}$, $\lambda(q_{10}, a) = \{q_{10}\}$ eta $\lambda(q_{12}, a) = \{q_{12}\}$ betetzen direnez, $\lambda^*(\lambda(q_5, a) \cup \lambda(q_6, a) \cup \lambda(q_7, a) \cup \lambda(q_{10}, a) \cup \lambda(q_{12}, a), b)$ espresiotik $\lambda^*(\{q_5, q_6, q_7, q_{10}, q_{12}\}, b)$ espresioa lortzen da.

- b hitza hutsa ez denez, eta Itxidura-$\varepsilon^*(\{q_5, q_6, q_7, q_{10}, q_{12}\}, \lambda) = \{q_5, q_6, q_7, q_{10}, q_{12}\}$ denez, honako hau daukagu:
  $\lambda^*(\{q_5, q_6, q_7, q_{10}, q_{12}\}, b) = \lambda^*(\lambda(q_5, b) \cup \lambda(q_6, b) \cup \lambda(q_7, b) \cup \lambda(q_{10}, b) \cup \lambda(q_{12}, ab), \varepsilon)$.

- $\lambda(q_5, b) = \{q_6\}$, $\lambda(q_6, b) = \{q_2, q_4\}$, $\lambda(q_7, b) = \{q_1, q_2\}$, $\lambda(q_{10}, b) = \{q_{10}\}$ eta $\lambda(q_{12}, b) = \{q_{13}\}$ betetzen direnez, $\lambda^*(\lambda(q_5, b) \cup \lambda(q_6, b) \cup \lambda(q_7, b) \cup \lambda(q_{10}, b) \cup \lambda(q_{12}, b), \varepsilon)$ espresiotik $\lambda^*(\{q_1, q_2, q_4, q_6, q_{10}, q_{13}\}, \varepsilon)$ espresioa lortzen da.

- ε hitz hutsa geratzen denez, $\lambda^*(\{q_1, q_2, q_4, q_6, q_{10}, q_{13}\}, \varepsilon)$ espresioaren balioa Itxidura-$\varepsilon^*(\{q_1, q_2, q_4, q_6, q_{10}, q_{13}\}, \lambda)$ da, hau da, $\{q_1, q_2, q_4, q_6, q_{10}, q_{13}\}$.

Beraz, $\lambda^*(\{q_1, q_4\}, caab) = \{q_1, q_2, q_4, q_6, q_{10}, q_{13}\}$.

Konputazio hori konfigurazio ez deterministez osatutako honako sekuentzia honen bidez adieraz daiteke:

$$
\begin{array}{c}
(\{q_1, q_4\}, caab) \equiv_{\text{Itxidura-}\varepsilon^*} (\{q_1, q_4, q_8\}, caab) \\
| \\
(\{q_2, q_3, q_5, q_6, q_{10}, q_{11}\}, aab) \equiv_{\text{Itxidura-}\varepsilon^*} (\{q_2, q_3, q_5, q_6, q_{10}, q_{11}, q_{12}\}, aab) \\
| \\
(\{q_5, q_6, q_7, q_{10}, q_{12}\}, ab) \equiv_{\text{Itxidura-}\varepsilon^*} (\{q_5, q_6, q_7, q_{10}, q_{12}\}, ab) \\
| \\
(\{q_5, q_6, q_7, q_{10}, q_{12}\}, b) \equiv_{\text{Itxidura-}\varepsilon^*} (\{q_5, q_6, q_7, q_{10}, q_{12}\}, b) \\
| \\
(\{q_1, q_2, q_4, q_6, q_{10}, q_{13}\}, \varepsilon) \equiv_{\text{Itxidura-}\varepsilon^*} (\{q_1, q_2, q_4, q_6, q_{10}, q_{13}\}, \varepsilon)
\end{array}
$$

Lerro bakoitzean une horretako egoera multzoa eta egoera horietatik ε trantsizioen bidez irisgarriak diren egoerak agertzen dira.

$E = (Q, A, \lambda, q_0, Y)$ ε-AFED bat emanda, **ε-AFED horri dagokion lengoaia**, ε-AFED horrek "Bai" erantzuten dien hitz guztiez osatuta dago. Beste era batera esanda, q_0 egoeratik abiatuz, osorik irakurri ondoren Y multzokoa den egoeraren bat duen egoera-multzo batean lagatzen gaituzten hitzez osatutako lengoaia da. E-ri dagokion lengoaia hori era formalean honela definituko genuke:

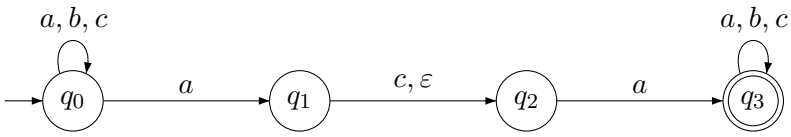
$$L(E) = \{w \mid w \in A^* \wedge (\lambda^*(\{q_0\}, w) \cap Y) \neq \varnothing\}$$

3.4.2 *Adibidea: aa katea edo aca katea duten hitzez osatutako lengoaiari dagokion ε-AFED-a*

Adibide honetako ε-AFED-ak (begiratu 3.21 irudia) a, b eta c sinboloak izan ditzaketen hitzak hartuko ditu sarrerako datu bezala. ε-AFED-ak, datu bezala jaso duen hitzak aa edo aca katea ba al duen erabakiko du. Baiezko kasuan "Bai" eta ezezko kasuan "Ez" erantzunez. Lengoaia horren definizio formala honako hau da:

$$\{w \mid w \in A^* \wedge \exists u, v(u \in A^* \wedge v \in A^* \wedge (w = uaav \vee w = uacav)\}$$

ε-AFED honek lau egoera ditu. Hasierako egoera q_0 da. Karaktere-kate bat irakurtzen hastean q_0 egoeran egongo gara. Lehenengo karakterea a baldin bada, ε-AFED hau q_0 eta q_1 egoeretara igaroko da, hau da, $\lambda(q_0, a) = \{q_0, q_1\}$. Lehenengo karakterea b edo c baldin bada, q_0 egoeran gelditukо da, hau da, $\lambda(q_0, b) = \{q_0\}$ eta $\lambda(q_0, c) = \{q_0\}$. ε-AFED hau q_1 egoeran baldin badago, c-rekin q_2-ra igaroko da, hau da, $\lambda(q_1, c) = \{q_2\}$. Bestalde a edo b-rekin ezingo du inora joan baina q_2 egoerara igaro daiteke sinbolorik irakurri gabe. Beraz, $\lambda(q_1, a) = \lambda(q_1, b) = \varnothing$ eta $\lambda(q_1, \varepsilon) = \{q_2\}$. ε-AFED hau q_2 egoeran baldin badago, a-rekin q_3 egoerara igaroko da, hau da, $\lambda(q_2, a) = \{q_3\}$. Aldiz, b edo c-rekin inora ezingo du joan. Beraz, $\lambda(q_2, b) = \lambda(q_2, c) = \varnothing$. Azkenik, ε-AFED hau q_3 egoeran baldin badago, bai a-rekin, bai b-rekin eta bai c-rekin, q_3 egoeran gelditukо da, hau da, $\lambda(q_3, a) = \lambda(q_3, b) = \lambda(q_3, c) = \{q_3\}$. Sarrerako datu bezala w hitza emanda, q_0-tik abiatuz eta hitz hori irakurriz lortzen den azken konfigurazioa (S, ε) baldin bada, lortzen den azken konfigurazioa el ε-AFED honek "Bai" erantzungo du $q_3 \in S$ betetzen bada. Hasierako w hitza ε baldin bada, ε-AFED honek q_0 egoeran bukatuko du, hau da, $\lambda^*(\{q_0\}, \varepsilon) = \{q_0\}$. Hor $\{q_0\} \cap Y = \varnothing$ betetzen denez, hau da, $\{q_0\} \cap \{q_3\} = \varnothing$ betetzen denez, ε-AFED honek "Ez" erantzungo du. ε-AFED honek hasierako datu bezala emandako w hitza irakurri ezinda geratzen bada, inora ezingo du joan. Hori $\lambda^*(\{q_0\}, w) = \varnothing$ betetzen dela esanez adierazten da. $\varnothing \cap Y = \varnothing$ denez, hau da, $\varnothing \cap \{q_3\} = \varnothing$ denez, ε-AFED "Ez" erantzungo du. Hasierako hitz osoa irakurtzea lortu arren, q_3 egoerara ez iritsiz gero "Ez" erantzungo da.



3.21 Irudia: aa katea edo aca katea duten hitzez osatutako lengoaiari dagokion ε-AFED-a.

ε-AFED honen osagaiak honako boskote honetan agertzen dira:

$$(Q, A, \lambda, q_0, Y)$$

Hor:

- $Q = \{q_0, q_1, q_2, q_3\}$
- $A = \{a, b, c\}$
- $\lambda : Q \times (A \cup \{\varepsilon\}) \to 2^Q$ jarraian erakusten den taularen bidez definitutako trantsizio funtzioa da:

λ	a	b	c	ε
q_0	$\{q_0, q_1\}$	$\{q_0\}$	$\{q_0\}$	$\varnothing$
q_1	$\varnothing$	$\varnothing$	$\{q_2\}$	$\{q_2\}$
q_2	$\{q_3\}$	$\varnothing$	$\varnothing$	$\varnothing$
q_3	$\{q_3\}$	$\{q_3\}$	$\{q_3\}$	$\varnothing$

Taula horren bidez $\lambda(q_0, a) = \{q_0, q_1\}$ dela, $\lambda(q_0, b) = \{q_0\}$ dela, eta abar adierazten da.

- q_0 hasierako egoera da. Hasierako egoera beti bakarra izan ohi da.
- $Y = \{q_3\}$.

Jarraian $(\{q_0\}, aaa)$ konfigurazio ez deterministari dagokion konputazioa garatuko da urratsez urrats:

1. urratsa: $\lambda^*(\{q_0\}, aaa) = \lambda^*(\lambda(q_0, a), aa) = \lambda^*(\{q_0, q_1\}, aa)$. Kasu honetan Itxidura-$\varepsilon^*(\{q_0\}) = \{q_0\}$ dela kontuan hartu da.

2. urratsa: $\lambda^*(\{q_0, q_1\}, aa) = \lambda^*(\lambda(q_0, a) \cup \lambda(q_1, a) \cup \lambda(q_2, a), a) = \lambda^*(\{q_0, q_1, q_3\}, a)$. Hor Itxidura-$\varepsilon^*(\{q_0, q_1\}) = \{q_0, q_1, q_2\}$ dela kontuan izan da.

3. urratsa: $\lambda^*(\{q_0, q_1, q_3\}, a) = \lambda^*(\lambda(q_0, a) \cup \lambda(q_1, a) \cup \lambda(q_2, a) \cup \lambda(q_3, a), \varepsilon) = \lambda^*(\{q_0, q_1, q_3\}, \varepsilon)$. Kasu honetan Itxidura-$\varepsilon^*(\{q_0, q_1, q_3\}) = \{q_0, q_1, q_2, q_3\}$ daukagu.

4. urratsa: $\lambda^*(\{q_0, q_1, q_3\}, \varepsilon) = \{q_0, q_1, q_2, q_3\}$, izan ere, Itxidura-$\varepsilon^*(\{q_0, q_1, q_3\}) = \{q_0, q_1, q_2, q_3\}$.

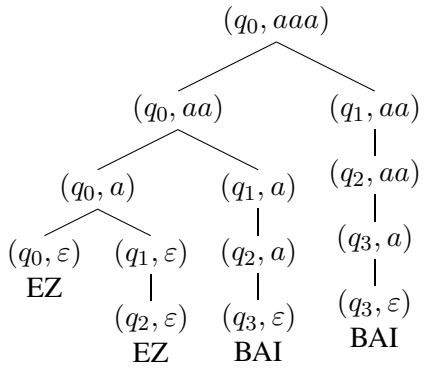
Konputazioa $\{q_0, q_1, q_2, q_3\}$ egoera-multzoan bukatzen denez, eta multzo horretan gutxienez egoera bat Y-koa denez, hau da, $\{q_0, q_1, q_2, q_3\} \cap Y \neq \varnothing$ betetzen denez, aaa hitza lengoaiakoa da (aa katea edo aca katea du).

Konputazio hori konfigurazio ez deterministez osatutako honako sekuentzia honen bidez adieraz daiteke:

$$
\begin{array}{c}
(\{q_0\}, aaa) \equiv_{\text{Itxidura-}\varepsilon^*} (\{q_0\}, aaa) \\
| \\
(\{q_0, q_1\}, aa) \equiv_{\text{Itxidura-}\varepsilon^*} (\{q_0, q_1, q_2\}, aa) \\
| \\
(\{q_0, q_1, q_3\}, a) \equiv_{\text{Itxidura-}\varepsilon^*} (\{q_0, q_1, q_2, q_3\}, a) \\
| \\
(\{q_0, q_1, q_3\}, \varepsilon) \equiv_{\text{Itxidura-}\varepsilon^*} (\{q_0, q_1, q_2, q_3\}, \varepsilon) \\
\text{BAI}
\end{array}
$$

Horko osagai bakoitzean, ε trantsizioak kontuan hartuz lortzen den itxidurarekin, hau da, Itxidura-ε^* funtzioarekin osatutako konfigurazio baliokidea erakusten da.

Konputazio hori trantsizio deterministez eratutako honako zuhaitz honen bidez ere adieraz daiteke:



Ezkerretik hasita, "Bai" erantzuten duen lehenengo adarrak bigarren aa azpikatea atzematen du, hau da, $a\underline{aa}$. Ezkerretik hasita, "Bai" erantzuten duen bigarren adarrak lehenengo aa azpikatea atzematen du, hau da, $\underline{aa}a$.

Jarraian $(\{q_0\}, abca)$ konfigurazio ez deterministari dagokion konputazioa garatuko da urratsez urrats:

1. urratsa: $\lambda^*(\{q_0\}, abca) = \lambda^*(\lambda(q_0, a), bca) = \lambda^*(\{q_0, q_1\}, bca)$. Hor Itxidura-$\varepsilon^*(\{q_0\}) = \{q_0\}$ dela kontuan izan da.

2. urratsa: $\lambda^*(\{q_0, q_1\}, bca) = \lambda^*(\lambda(q_0, b) \cup \lambda(q_1, b) \cup \lambda(q_2, b), ca) = \lambda^*(\{q_0\}, ca)$. Kasu honetan Itxidura-$\varepsilon^*(\{q_0, q_1\}) = \{q_0, q_1, q_2\}$ betetzen da.

3. urratsa: $\lambda^*(\{q_0\}, ca) = \lambda^*(\lambda(q_0, c), a) = \lambda^*(\{q_0\}, a)$. Kasu honetan Itxidura-$\varepsilon^*(\{q_0\}) = \{q_0\}$ daukagu.

4. urratsa: $\lambda^*(\{q_0\}, a) = \lambda^*(\lambda(q_0, a), \varepsilon) = \lambda^*(\{q_0, q_1\}, \varepsilon)$. Urrats honetan Itxidura-$\varepsilon^*(\{q_0\}) = \{q_0\}$ dela kontuan izan behar da.

5. urratsa: $\lambda^*(\{q_0, q_1\}, \varepsilon) = \{q_0, q_1, q_2\}$, izan ere Itxidura-$\varepsilon^*(\{q_0, q_1\}) = \{q_0, q_1, q_2\}$ baita.

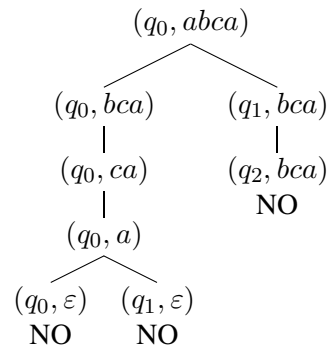
Bukaeran lortutako egoera-multzoan Y-koa den egoerarik ez dagoenez, $abca$ hitza ez da ε-AFED honi dagokion lengoaiakoa.

Konputazio hori deterministak ez diren konfigurazioz osatutako honako sekuentziaren bidez adieraz daiteke:

$$
\begin{array}{c}
(\{q_0\}, abca) \equiv_{\text{Itxidura-}\varepsilon^*} (\{q_0\}, abca) \\
| \\
(\{q_0, q_1\}, bca) \equiv_{\text{Itxidura-}\varepsilon^*} (\{q_0, q_1, q_2\}, bca) \\
| \\
(\{q_0\}, ca) \equiv_{\text{Itxidura-}\varepsilon^*} (\{q_0\}, ca) \\
| \\
(\{q_0\}, a) \equiv_{\text{Itxidura-}\varepsilon^*} (\{q_0\}, a) \\
| \\
(\{q_0, q_1\}, \varepsilon) \equiv_{\text{Itxidura-}\varepsilon^*} (\{q_0, q_1, q_2\}, \varepsilon) \\
\text{EZ}
\end{array}
$$

Horko osagai bakoitzean, ε trantsizioak kontuan hartuz lortzen den itxidurarekin, hau da, Itxidura-ε^* funtzioarekin osatutako konfigurazio baliokidea erakusten da.

Konputazio hori trantsizio deterministez eratutako honako zuhaitz honen bidez ere adieraz daiteke:



Adar denetan ezezkoa lortzen da.

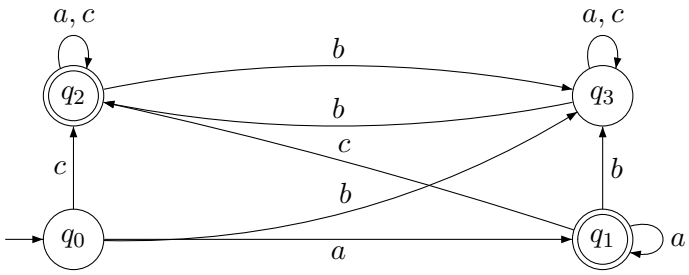
3.4.3 Adibidea: Bakarrik a sinboloaren errepikapenez osatuta dauden edo b-ren agerpen-kopuru bikoitia duten hitz ez hutsez eratutako lengoaiari dagokion ε-AFED-a

Adibide honetako ε-AFED-ak a, b eta c sinboloak izan ditzaketen hitzak hartuko ditu sarrerako datu bezala. ε-AFED-ak, datu bezala jaso duen hitzak bakarrik a sinboloaren errepikapenez osatuta al dauden edo b-ren agerpen-kopuru bikoitia al duten hitz ez hutsak al diren erabakiko du. Baiezko kasuan "Bai" eta ezezko kasuan "Ez" erantzunez. Lengoaia horren definizio formala honako hau da:

$$\{w \mid w \in A^* \wedge |w| \geq 1 \wedge (|w|_a = |w| \vee |w|_b \bmod 2 = 0)\}$$

Baina ε-AFED hori aurkeztu aurretik, lengoaia bera definitzen duten AFD bat eta AFED bat erakutsiko dira. Horrela hiru formalismo horiek alderatu ahal izango ditugu eta gehienetan ε-AFED-ak AFED-ak eta AFD-ak baino ulerterrazagoak direla ikusiko dugu. Era berean. A su vez AFED-ak AFD-ak baino ulerterrazagoak direla ere erakutsiko da. Izan ere, adibide honetan kontrolatu beharreko baldintzak bakoitza bere aldetik aztertu ahal izango da AFED-a erabiliz, diseinua erraztuz.

Hasteko, 3.22 irudian adibide honetako lengoaiari dagokion AFD bat erakusten da.



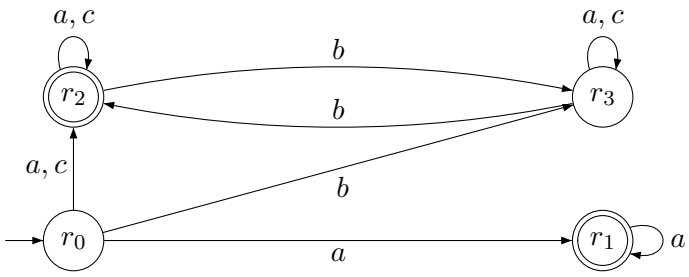
3.22 Irudia: Bakarrik a sinboloaren errepikapenez osatuta dauden edo b-ren agerpen-kopuru bikoitia duten hitz ez hutsez eratutako lengoaiari dagokion AFD-a.

Hor, 3.22 irudian, q_1 egoerak orain arte irakurri diren sinbolo denak a izan direla eta gainera gutxienez bat irakurri dela eta b-rik ez dela irakurri (eta ondorioz b-ren agerpen-kopurua bikoitia dela) gogoratzen du. Bestalde, q_2 egoeran gaudenean badakigu gutxienez a ez den sinboloren bat irakurri dela eta b kopurua bikoitia dela. Bukatzeko, q_3 egoeran gaudenean badakigu gutxienez a ez den sinboloren bat irakurri dela eta b kopurua bakoitia dela. Hor q_1-etik q_2-ra eta q_1-etik q_3-ra doazen tratsizioak direla eta, aztertu beharreko bi baldintzen egiaztapena (bakarrik a agertzea edo b kopurua bikoitia izatea) nahastuta dago. Bakarrik a sinboloaren errepikapenez osatuta dauden hitz ez hutsak atzemateko q_0 eta q_1 egoerak erabiltzen dira eta b kopuru bikoitia duten hitz hutsak atzemateko lau egoerak behar dira (q_0, q_1, q_2 eta q_3).

Lengoaia bera definitzen duen AFED bat 3.23 irudian ikus dezakegu.

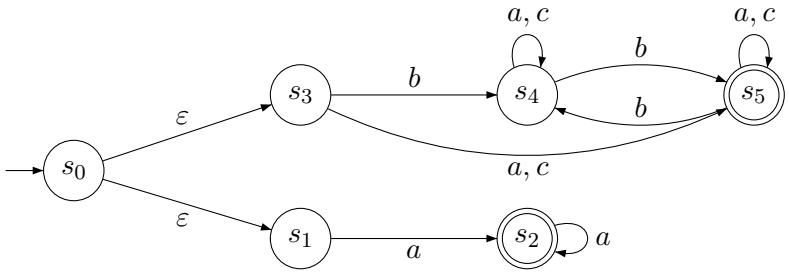
3.23 irudiko AFED-ko r_1, r_2 eta r_3 egoerek 3.22 irudiko AFD-ko q_1, q_2 eta q_3 egoeren helburu bera dute. Baina AFED honetan bakarrik a-ren errepikapenez osatutako hitza izatea (r_0 eta r_1 egoeren bidez) eta b kopurua bikoitia izatea (r_0, r_2 eta r_3 egoeren bidez) bakoitza bere aldetik aztertu daiteke. Beraz, AFED honen bidez disjuntzioaren esanahia 3.22 irudiko AFD-an baino era garbiagoan adierazten da.

3.24 irudian, adibide honetako lengoaia definitzen duen ε-AFED bat daukagu. Hor s_0 egoeratik ateratzen diren ε trantsizioei esker, baldintza biak (hitz hutsa ez izanda, bakarrik a-ren errepikapenez osatuta egotea edo b kopurua bikoitia izatea) 3.23 irudiko AFED-an baino hobeto bereizten dira. s_0-tik abiatzerakoan, zuzenean s_1 eta s_3 egoeretara igarotzen da ezer irakurri gabe. Alde batetik, s_1 eta s_2 egoeren bidez hitza hutsa ez izatea eta a-ren errepikapenez bakarrik osatuta egotea kontrolatzen da. Bestetik, s_3, s_4 eta s_5 egoeren bidez hitza hutsa ez izatea eta b kopurua bikoitia izatea kontrolatzen da.



3.23 Irudia: akarrik a sinboloaren errepikapenez osatuta dauden edo b-ren agerpen-kopuru bikoitia duten hitz ez hutsez eratutako lengoaiari dagokion AFED-a.

Datu bezala emandako hitzak bi baldintza horietakoren bat betetzen badu, erantzuna "Bai" izango da eta bi baldintza horietatik bat bera ere ez badu betetzen, erantzuna "Ez" izango da.



3.24 Irudia: akarrik a sinboloaren errepikapenez osatuta dauden edo b-ren agerpen-kopuru bikoitia duten hitz ez hutsez eratutako lengoaiari dagokion ε-AFED-a.

Adibide honetako ε-AFED-a honako boskote honen bidez defini daiteke:

$$(Q, A, \lambda, s_0, Y)$$

Hor:

- $Q = \{s_0, s_1, s_2, s_3, s_4, s_5\}$
- $A = \{a, b, c\}$
- $\lambda : Q \times (A \cup \{\varepsilon\}) \to 2^Q$ honako taula honen bidez definitutako trantsizio funtzioa da:

λ	a	b	c	ε
s_0	$\varnothing$	$\varnothing$	$\varnothing$	$\{s_1, s_3\}$
s_1	$\{s_2\}$	$\varnothing$	$\varnothing$	$\varnothing$
s_2	$\{s_2\}$	$\varnothing$	$\varnothing$	$\varnothing$
s_3	$\{s_5\}$	$\{s_4\}$	$\{s_5\}$	$\varnothing$
s_4	$\{s_4\}$	$\{s_5\}$	$\{s_4\}$	$\varnothing$
s_5	$\{s_5\}$	$\{s_4\}$	$\{s_5\}$	$\varnothing$

  Taula horretan $\lambda(s_0, a) = \varnothing$ dela, $\lambda(q_0, \varepsilon) = \{s_1, s_3\}$ dela, eta abar adierazten da.

- s_0 hasierako egoera da. Hasierako egoera bakarra izan ohi da beti.
- $Y = \{s_2, s_5\}$.

Orain $(\{q_0\}, aaa)$ konfigurazio ez deterministari dagokion konputazioa garatuko da urratsez urrats:

Paso 1: $\lambda^*(\{s_0\}, aaa) = \lambda^*(\lambda(s_0, a) \cup \lambda(s_1, a) \cup \lambda(s_3, a), aa) = \lambda^*(\{s_2, s_5\}, aa)$. Se ha de tener en cuenta que Itxidura-$\varepsilon^*(\{s_0\}, \lambda) = \{s_0, s_1, s_3\}$.

Paso 2: $\lambda^*(\{s_2, s_5\}, aa) = \lambda^*(\lambda(s_2, a) \cup \lambda(s_5, a) = \lambda^*(\{s_2, s_5\}, a)$. En este caso se ha de tenido en cuenta que Itxidura-$\varepsilon^*(\{s_2, s_5\}, \lambda) = \{s_2, s_5\}$.

Paso 3: $\lambda^*(\{s_2, s_5\}, a) = \lambda^*(\{s_2, s_5\}, a\varepsilon) = \lambda^*(\lambda(s_2, a) \cup \lambda(s_5, a), \varepsilon) = \lambda^*(\{s_2, s_5\}, \varepsilon)$. Otra vez se ha de tenido en cuenta que Itxidura-$\varepsilon^*(\{s_2, s_5\}, \lambda) = \{s_2, s_5\}$.

Paso 4: $\lambda^*(\{s_2, s_5\}, \varepsilon) = \{s_2, s_5\}$ puesto que Itxidura-$\varepsilon^*(\{s_2, s_5\}, \lambda) = \{s_2, s_5\}$.

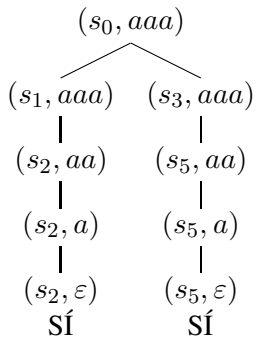
Como la computación termina en el conjunto de estados $\{s_2, s_5\}$ y como la intersección entre $\{s_2, s_5\}$ y el conjunto Y no es vacía, sabemos que la palabra es del lenguaje.

Esa computación se puede mostrar gráficamente como la siguiente secuencia de configuraciones no deterministas:

$$
\begin{array}{c}
(\{s_0\}, aaa) \equiv_{\text{Itxidura-}\varepsilon^*} (\{s_0, s_1, s_3\}, aaa) \\
| \\
(\{s_2, s_5\}, aa) \equiv_{\text{Itxidura-}\varepsilon^*} (\{s_2, s_5\}, aa) \\
| \\
(\{s_2, s_5\}, a) \equiv_{\text{Itxidura-}\varepsilon^*} (\{s_2, s_5\}, a) \\
| \\
(\{s_2, s_5\}, \varepsilon) \equiv_{\text{Itxidura-}\varepsilon^*} (\{s_2, s_5\}, \varepsilon)
\end{array}
$$

En cada nodo se indica la configuración correspondiente o equivalente teniendo en cuenta el cierre por transiciones ε, es decir, Itxidura-ε^*.

Esa computación se puede mostrar gráficamente como el siguiente árbol de configuraciones deterministas:



En este caso todas las ramas son afirmativas. La primera rama (desde la izquierda) detecta que solo hay a's y la segunda rama detecta que el número de b's es par.

A continuación vamos a desarrollar paso a paso la computación correspondiente a la configuración no determinista $(\{q_0\}, abca)$:

Paso 1: $\lambda^*(\{s_0\}, abca) = \lambda^*(\lambda(s_0, a) \cup \lambda(s_1, a) \cup \lambda(s_3, a), bca) = \lambda^*(\{s_2, s_5\}, bca)$. Se ha de tener en cuenta que Itxidura-$\varepsilon^*(\{s_0\}, \lambda) = \{s_0, s_1, s_3\}$.

Paso 2: $\lambda^*(\{s_2, s_5\}, bca) = \lambda^*(\lambda(s_2, b) \cup \lambda(s_5, b), ca) = \lambda^*(\{s_4\}, ca)$. En este caso se ha de tenido en cuenta que Itxidura-$\varepsilon^*(\{s_2, s_5\}, \lambda) = \{s_2, s_5\}$.

Paso 3: $\lambda^*(\{s_4\}, ca) = \lambda^*(\lambda(s_4, c), a) = \lambda^*(\{s_4\}, a)$. En esta ocasión se tiene que ltxidura-$\varepsilon^*(\{s_4\}, \lambda) = \{s_4\}$.

Paso 4: $\lambda^*(\{s_4\}, a) = \lambda^*(\{s_4\}, a\varepsilon) = \lambda^*(\lambda(s_4, a), \varepsilon) = \lambda^*(\{s_4\}, \varepsilon)$ puesto que ltxidura-$\varepsilon^*(\{s_4\}, \lambda) = \{s_4\}$.

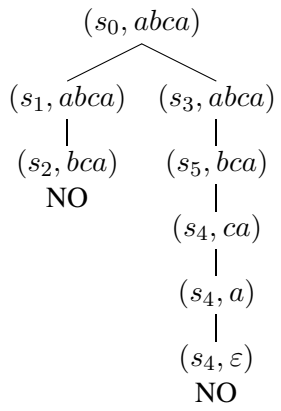
Paso 5: $\lambda^*(\{s_4\}, \varepsilon) = \{s_4\}$ puesto que ltxidura-$\varepsilon^*(\{s_4\}, \lambda) = \{s_4\}$.

Como la computación termina en el conjunto $\{s_4\}$ que no tiene ningún estado que pertenezca a Y, sabemos que la palabra no pertenece al lenguaje definido por el ε-AFND.

Esa computación se puede mostrar gráficamente como la siguiente secuencia de configuraciones no deterministas:

$$
\begin{array}{c}
(\{s_0\}, abca) \equiv_{\text{ltxidura-}\varepsilon^*} (\{s_0, s_1, s_3\}, abca) \\
| \\
(\{s_2, s_5\}, bca) \equiv_{\text{ltxidura-}\varepsilon^*} (\{s_2, s_5\}, bca) \\
| \\
(\{s_4\}, ca) \equiv_{\text{ltxidura-}\varepsilon^*} (\{s_4\}, ca) \\
| \\
(\{s_4\}, a) \equiv_{\text{ltxidura-}\varepsilon^*} (\{s_4\}, a) \\
| \\
(\{s_4\}, \varepsilon) \equiv_{\text{ltxidura-}\varepsilon^*} (\{s_4\}, \varepsilon)
\end{array}
$$

En cada nodo se indica la configuración correspondiente o equivalente teniendo en cuenta el cierre por transiciones ε, es decir, ltxidura-ε^*. Esa computación se puede mostrar gráficamente como el siguiente árbol de configuraciones deterministas:



En este caso todas las ramas son negativas.

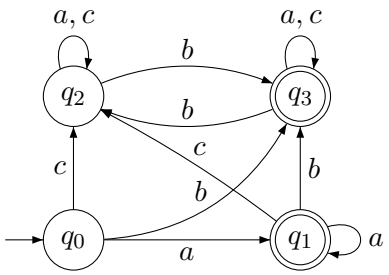
3.4.4 *Ejemplo: ε-AFND para el lenguaje formado por palabras no vacías que solo contienen a's o tienen un número impar de b's*

En este ejemplo se va a mostrar un ε-AFND que recibe como entrada una cadena de caracteres que puede contener a's, b's y c's y decide si la palabra es una secuencia no vacía que solo contienen a's o tienen un número impar de b's. La palabra vacía ε no pertenece al lenguaje. Formalmente el lenguaje es el siguiente:

$$\{w \mid w \in A^* \wedge |w| \geq 1 \wedge (|w|_a = |w| \vee |w|_b \bmod 2 \neq 0)\}$$

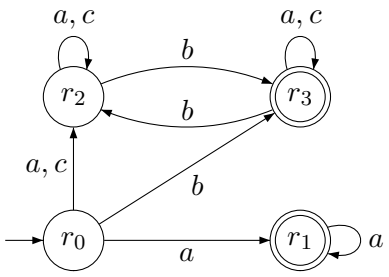
Antes de presentar dicho ε-AFND, se van a mostrar un AFD y un AFND que definen el mismo lenguaje. El objetivo es poder comparar los tres formalismos y ver que normalmente los ε-AFND's son más sencillos que los AFND's y los AFD's, de la misma forma que los AFND's son más sencillos que los AFD's. En este ejemplo es posible apreciar que con un AFND se simplifica un poco el diseño con respecto a un AFD y que, a su vez, con un ε-AFND se simplifica un poco el diseño con respecto a un AFND.

En primer lugar se muestra en la Figura 3.25 un AFD que define el lenguaje de este ejemplo. Tal como ocurría en el ejemplo de la sección anterior, en este AFD hay conexión entre el estado q_1 que detecta las cadenas que solo contienen a's y los estados q_2 y q_3 que controlan la propiedad de que el número de b's sea impar.



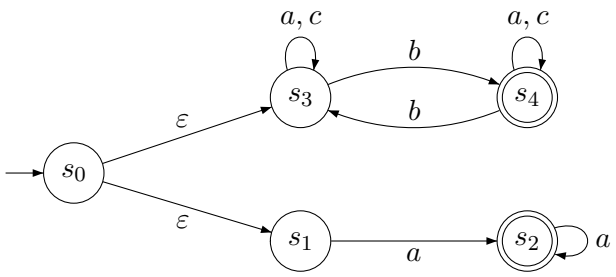
3.25 Irudia: AFD para el lenguaje formado por palabras no vacías que solo contienen a's o tienen un número impar de b's.

En segundo lugar se muestra en la Figura 3.26 un AFND que define el mismo lenguaje. En este AFND no hay transición de r_1 a r_2 y de r_1 a r_3. Por tanto se consigue separar la verificación de las dos propiedades.



3.26 Irudia: AFND para el lenguaje formado por palabras no vacías que solo contienen a's o tienen un número impar de b's.

En tercer lugar se presenta, en la Figura 3.27, el ε-AFND que define el mismo lenguaje. Las transiciones ε que salen desde el estado s_0 permiten separar por completo las dos condiciones (tener solo a's o tener un número impar de b's). Si se parte del estado s_0, se pasa directamente a los estados s_1 y s_3. Por un lado, mediante los estados s_1 y s_2 se controla la propiedad de que la palabra no tenga ni b's ni c's pero tenga al menos una a. Por otro lado, mediante los estados s_3 y s_4 se controla la propiedad de que la palabra tenga un número impar de b's. Si la palabra cumple alguna de esas propiedades, la salida será "Sí" mientras que si la palabra no cumple ninguna de esas dos propiedades la salida será "No". Estos autómatas son parecidos a los del ejemplo anterior pero cambian detalles importantes como los estados que llevan doble círculo.



3.27 Irudia: ε-AFND para el lenguaje formado por palabras no vacías que solo contienen a's o tienen un número impar de b's.

A continuación se definen los componentes del ε-AFND de la Figura 3.27. El ε-AFND de este ejemplo es una quíntupla de la forma

$$(Q, A, \lambda, s_0, Y)$$

donde:

- $Q = \{s_0, s_1, s_2, s_3, s_4\}$
- $A = \{a, b, c\}$
- $\lambda : Q \times (A \cup \{\varepsilon\}) \rightarrow 2^Q$ es la función de transición definida mediante la siguiente tabla:

λ	a	b	c	ε
s_0	$\varnothing$	$\varnothing$	$\varnothing$	$\{s_1, s_3\}$
s_1	$\{s_2\}$	$\varnothing$	$\varnothing$	$\varnothing$
s_2	$\{s_2\}$	$\varnothing$	$\varnothing$	$\varnothing$
s_3	$\{s_3\}$	$\{s_4\}$	$\{s_3\}$	$\varnothing$
s_4	$\{s_4\}$	$\{s_3\}$	$\{s_4\}$	$\varnothing$

  Mediante la tabla se indica que $\lambda(s_0, a)$ es $\varnothing$, que $\lambda(s_0, \varepsilon)$ es $\{s_1, s_3\}$, etc.

- q_0 es el estado inicial. Recordemos que solo puede haber un estado inicial.
- $Y = \{s_2, s_4\}$.

Ahora vamos a desarrollar paso a paso la computación correspondiente a la configuración no determinista $(\{q_0\}, aca)$:

Paso 1: $\lambda^*(\{s_0\}, aca) = \lambda^*(\lambda(s_0, a) \cup \lambda(s_1, a) \cup \lambda(s_3, a), ca) = \lambda^*(\{s_2, s_3\}, ca)$. Se ha de tener en cuenta que Itxidura-$\varepsilon^*(\{s_0\}, \lambda) = \{s_0, s_1, s_3\}$.

Paso 2: $\lambda^*(\{s_2, s_3\}, ca) = \lambda^*(\lambda(s_2, c) \cup \lambda(s_3, c), a) = \lambda^*(\{s_3\}, a)$. En este caso se ha de tenido en cuenta que Itxidura-$\varepsilon^*(\{s_2, s_3\}, \lambda) = \{s_2, s_3\}$.

Paso 3: $\lambda^*(\{s_3\}, a) = \lambda^*(\{s_3\}, a\varepsilon) = \lambda^*(\lambda(s_3, a), \varepsilon) = \lambda^*(\{s_3\}, \varepsilon)$. Se ha de tenido en cuenta que Itxidura-$\varepsilon^*(\{s_3\}, \lambda) = \{s_3\}$.

Paso 4: $\lambda^*(\{s_3\}, \varepsilon) = \{s_3\}$ puesto que Itxidura-$\varepsilon^*(\{s_3\}, \lambda) = \{s_3\}$.

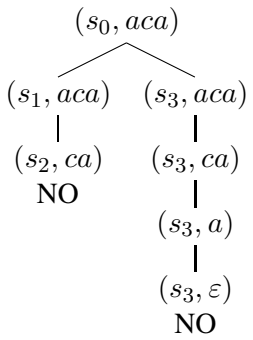
Como la computación termina en el conjunto de estados $\{s_3\}$ y como la intersección entre $\{s_3\}$ y el conjunto Y es vacía, sabemos que la palabra no es del lenguaje.

Esa computación se puede mostrar gráficamente como la siguiente secuencia de configuraciones no deterministas:

$$\begin{array}{c}(\{s_0\}, aca) \equiv_{\text{Itxidura-}\varepsilon^*} (\{s_0, s_1, s_3\}, aca)\\ \vert \\ (\{s_2, s_3\}, ca) \equiv_{\text{Itxidura-}\varepsilon^*} (\{s_2, s_3\}, ca)\\ \vert \\ (\{s_3\}, a) \equiv_{\text{Itxidura-}\varepsilon^*} (\{s_3\}, a)\\ \vert \\ (\{s_3\}, \varepsilon) \equiv_{\text{Itxidura-}\varepsilon^*} (\{s_3\}, \varepsilon)\end{array}$$

En cada nodo se indica la configuración correspondiente o equivalente teniendo en cuenta el cierre por transiciones ε, es decir, Itxidura-ε^*.

Esa computación se puede mostrar gráficamente como el siguiente árbol de configuraciones deterministas:



En este caso todas las ramas son negativas.

A continuación vamos a desarrollar paso a paso la computación correspondiente a la configuración no determinista $(\{q_0\}, abca)$:

Paso 1: $\lambda^*(\{s_0\}, abca) = \lambda^*(\lambda(s_0, a) \cup \lambda(s_1, a) \cup \lambda(s_3, a), bca) = \lambda^*(\{s_2, s_3\}, bca)$. Se ha de tener en cuenta que Itxidura-$\varepsilon^*(\{s_0\}, \lambda) = \{s_0, s_1, s_3\}$.

Paso 2: $\lambda^*(\{s_2, s_3\}, bca) = \lambda^*(\lambda(s_2, b) \cup \lambda(s_3, b), ca) = \lambda^*(\{s_4\}, a)$. En este caso se ha de tenido en cuenta que Itxidura-$\varepsilon^*(\{s_2, s_3\}, \lambda) = \{s_2, s_3\}$.

Paso 3: $\lambda^*(\{s_4\}, ca) = \lambda^*(\lambda(s_4, c), a) = \lambda^*(\{s_4\}, a)$. En esta ocasión se tiene que Itxidura-$\varepsilon^*(\{s_4\}, \lambda) = \{s_4\}$.

Paso 4: $\lambda^*(\{s_4\}, a) = \lambda^*(\{s_4\}, a\varepsilon) = \lambda^*(\lambda(s_4, a), \varepsilon) = \lambda^*(\{s_4\}, \varepsilon)$ puesto que Itxidura-$\varepsilon^*(\{s_4\}, \lambda) = \{s_4\}$.

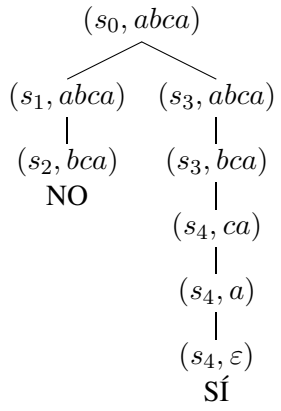
Paso 5: $\lambda^*(\{s_4\}, \varepsilon) = \{s_4\}$ puesto que Itxidura-$\varepsilon^*(\{s_4\}, \lambda) = \{s_4\}$.

Como la computación termina en el conjunto $\{s_4\}$ que contiene un estado que pertenece a Y, sabemos que la palabra pertenece al lenguaje definido por el ε-AFND.

Esa computación se puede mostrar gráficamente como la siguiente secuencia de configuraciones no deterministas:

$$
\begin{array}{c}
(\{s_0\}, abca) \equiv_{\textsf{Itxidura-}\varepsilon^*} (\{s_0, s_1, s_3\}, abca) \\
| \\
(\{s_2, s_3\}, bca) \equiv_{\textsf{Itxidura-}\varepsilon^*} (\{s_2, s_3\}, bca) \\
| \\
(\{s_4\}, ca) \equiv_{\textsf{Itxidura-}\varepsilon^*} (\{s_4\}, ca) \\
| \\
(\{s_4\}, a) \equiv_{\textsf{Itxidura-}\varepsilon^*} (\{s_4\}, a) \\
| \\
(\{s_4\}, \varepsilon) \equiv_{\textsf{Itxidura-}\varepsilon^*} (\{s_4\}, \varepsilon)
\end{array}
$$

En cada nodo se indica la configuración correspondiente o equivalente teniendo en cuenta el cierre por transiciones ε, es decir, Itxidura-ε^*. Esa computación se puede mostrar gráficamente como el siguiente árbol de configuraciones deterministas:



En este caso la segunda rama es afirmativa y detecta que el número de b's es impar.

3.4.5 Ejemplo: ε-AFND para el lenguaje formado por palabras que contienen un número par de a's, b's o c's

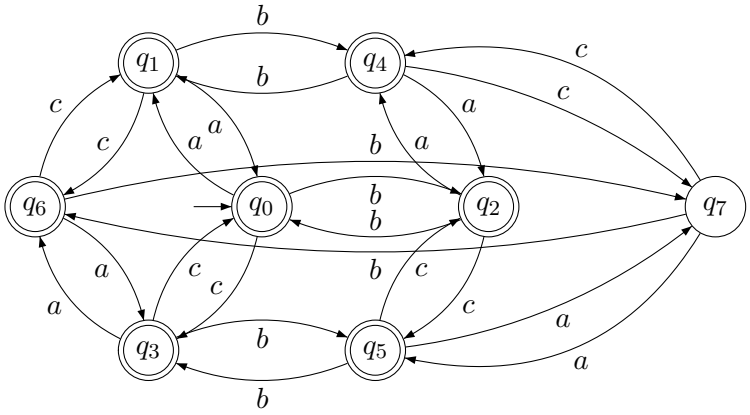
En este ejemplo se va a mostrar un ε-AFND que recibe como entrada una cadena de caracteres que puede contener a's, b's y c's y decide si la palabra es una secuencia que contiene un número par de a's, b's o c's. Basta con que un símbolo de ellos aparezca un número par de veces. La palabra vacía ε pertenece al lenguaje. Formalmente el lenguaje es el siguiente:

$$\{w \mid w \in A^* \wedge (|w|_a \bmod 2 = 0 \vee |w|_b \bmod 2 = 0 \vee |w|_c \bmod 2 = 0)\}$$

Antes de presentar dicho ε-AFND, se van a mostrar un AFD y un AFND que definen el mismo lenguaje. El objetivo es poder comparar los tres formalismos y ver que normalmente los ε-AFND's son más sencillos que los AFND's y los AFD's, de la misma forma que los AFND's son más sencillos que los AFD's. En este ejemplo es posible apreciar que con un AFND se simplifica un poco el diseño con respecto a un AFD y que, a su vez, con un ε-AFND se simplifica un poco el diseño con respecto a un AFND.

En primer lugar se muestra en la Figura 3.28 un AFD que define el lenguaje de este ejemplo. En este AFD las tres propiedades a controlar están totalmente interconectadas y aunque no es muy complicado, tampoco es trivial averiguar la propiedad recordada por cada estado. De todas formas, este ejemplo es bastante pequeño pero nos puede servir para hacernos una idea de que si las propiedades a comprobar son más complicadas, el AFD puede resultar difícil de diseñar y de entender.

En segundo lugar se muestra en la Figura 3.29 un AFND que define el mismo lenguaje. Mediante este AFND se consigue diferenciar la verificación de las tres propiedades: Los estados r_0, r_1 y r_2



3.28 Irudia: AFD para el lenguaje formado por palabras que contienen un número par de a's, b's o c's.

comprueban la propiedad de que el número de a's sea par; los estados r_0, r_3 y r_4 comprueban la propiedad de que el número de b's sea par; por último, los estados r_0, r_1 y r_2 comprueban la propiedad de que el número de c's sea par. El diseño es ahora mucho más claro aunque todavía el hecho de que las tres propiedades incluya r_0 hace que la conexión entre r_0 y los estados r_1, r_3 y r_5 introduzca cierta confusión o falta de claridad.

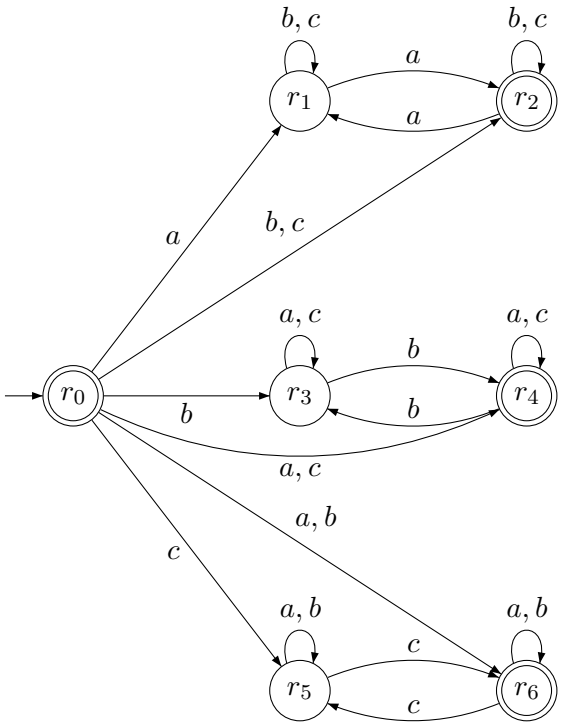
En tercer lugar se presenta, en la Figura 3.30, el ε-AFND que define el mismo lenguaje. Las transiciones ε que salen desde el estado s_0 permiten separar por completo las tres condiciones (tener un número par de a's o tener un número par de b's o tener un número par de c's). Si se parte del estado s_0, se pasa directamente a los estados s_1, s_3 y s_5. Por un lado, mediante los estados s_1 y s_2 se controla la propiedad de que la palabra tenga un número par de a's. Por otro lado, mediante los estados s_3 y s_4 se controla la propiedad de que la palabra tenga un número par de b's. Por último, mediante los estados s_5 y s_6 se controla la propiedad de que la palabra tenga un número par de c's. Si la palabra cumple alguna de esas propiedades, la salida será "Sí" mientras que si la palabra no cumple ninguna de esas tres propiedades la salida será "No". Este ejemplo muestra de manera clara la importante simplificación que se obtiene en algunas ocasiones al utilizar un ε-AFND.

A continuación se definen los componentes del ε-AFND de la Figura 3.30. Dicho ε-AFND es una quíntupla de la forma

$$(Q, A, \lambda, s_0, Y)$$

donde:

- $Q = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6\}$
- $A = \{a, b, c\}$
- $\lambda : Q \times (A \cup \{\varepsilon\}) \rightarrow 2^Q$ es la función de transición definida mediante la siguiente tabla:



3.29 Irudia: AFND para el lenguaje formado por palabras que contienen un número par de a's, b's o c's.

λ	a	b	c	ε
s_0	$\varnothing$	$\varnothing$	$\varnothing$	$\{s_1, s_3, s_5\}$
s_1	$\{s_2\}$	$\{s_1\}$	$\{s_1\}$	$\varnothing$
s_2	$\{s_1\}$	$\{s_2\}$	$\{s_2\}$	$\varnothing$
s_3	$\{s_3\}$	$\{s_4\}$	$\{s_3\}$	$\varnothing$
s_4	$\{s_4\}$	$\{s_3\}$	$\{s_4\}$	$\varnothing$
s_5	$\{s_5\}$	$\{s_5\}$	$\{s_6\}$	$\varnothing$
s_6	$\{s_6\}$	$\{s_6\}$	$\{s_5\}$	$\varnothing$

Mediante la tabla se indica que $\lambda(s_0, a)$ es $\varnothing$, que $\lambda(s_0, \varepsilon)$ es $\{s_1, s_3, s_5\}$, etc.

- s_0 es el estado inicial. Recordemos que solo puede haber un estado inicial.
- $Y = \{s_1, s_3, s_5\}$. En este caso, también s_0 podría ser incluido en Y, pero no hace falta, porque desde s_0 se puede pasar a s_1, s_3 y s_5 sin consumir ningún símbolo.

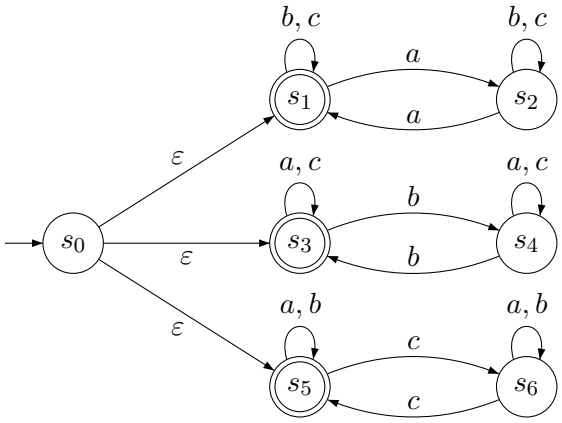
Ahora vamos a desarrollar paso a paso la computación correspondiente a la configuración no determinista $(\{q_0\}, aca)$:

Paso 1: $\lambda^*(\{s_0\}, aca) = \lambda^*(\lambda(s_0, a) \cup \lambda(s_1, a) \cup \lambda(s_3, a) \cup \lambda(s_5, a), ca) = \lambda^*(\{s_2, s_3, s_5\}, ca)$.
Se ha de tener en cuenta que Itxidura-$\varepsilon^*(\{s_0\}, \lambda) = \{s_0, s_2, s_3, s_5\}$.

Paso 2: $\lambda^*(\{s_2, s_3, s_5\}, ca) = \lambda^*(\lambda(s_2, c) \cup \lambda(s_3, c) \cup \lambda(s_5, c), a) = \lambda^*(\{s_2, s_3, s_6\}, a)$. En este caso se ha de tenido en cuenta que Itxidura-$\varepsilon^*(\{s_2, s_3, s_5\}, \lambda) = \{s_2, s_3, s_5\}$.

Paso 3: $\lambda^*(\{s_2, s_3, s_6\}, a) = \lambda^*(\{s_2, s_3, s_6\}, a\varepsilon) = \lambda^*(\lambda(s_2, a) \cup \lambda(s_3, a) \cup \lambda(s_6, a), \varepsilon) = \lambda^*(\{s_1, s_3, s_6\}, \varepsilon)$.
Se ha de tenido en cuenta que Itxidura-$\varepsilon^*(\{s_2, s_3, s_6\}, \lambda) = \{s_2, s_3, s_6\}$.

Paso 4: $\lambda^*(\{s_1, s_3, s_6\}, \varepsilon) = \{s_1, s_3, s_6\}$ puesto que Itxidura-$\varepsilon^*(\{s_1, s_3, s_6\}, \lambda) = \{s_1, s_3, s_6\}$.



3.30 Irudia: ε-AFND para el lenguaje formado por palabras que contienen un número par de a's, b's o c's.

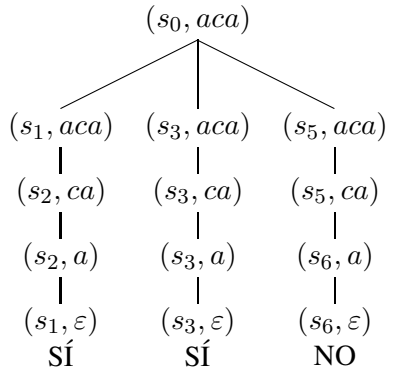
Como la computación termina en el conjunto de estados $\{s_1, s_3, s_6\}$ y como la intersección entre $\{s_1, s_3, s_6\}$ y el conjunto Y no es vacía, sabemos que la palabra es del lenguaje (a, b o c aparece un número par de veces).

Esa computación se puede mostrar gráficamente como la siguiente secuencia de configuraciones no deterministas:

$$
\begin{array}{c}
(\{s_0\}, aca) \equiv_{\mathsf{ltxidura}\text{-}\varepsilon^*} (\{s_0, s_1, s_3, s_5\}, aca) \\
| \\
(\{s_2, s_3, s_5\}, ca) \equiv_{\mathsf{ltxidura}\text{-}\varepsilon^*} (\{s_2, s_3, s_5\}, ca) \\
| \\
(\{s_2, s_3, s_6\}, a) \equiv_{\mathsf{ltxidura}\text{-}\varepsilon^*} (\{s_2, s_3, s_6\}, a) \\
| \\
(\{s_1, s_3, s_6\}, \varepsilon) \equiv_{\mathsf{ltxidura}\text{-}\varepsilon^*} (\{s_1, s_3, s_6\}, \varepsilon)
\end{array}
$$

En cada nodo se indica la configuración correspondiente o equivalente teniendo en cuenta el cierre por transiciones ε, es decir, ltxidura-ε^*.

Esa computación se puede mostrar gráficamente como el siguiente árbol de configuraciones deterministas:



La primera rama (desde la izquierda) detecta que la palabra tiene un número par de a's y la segunda rama (desde la izquierda) detecta que la palabra tiene un número par de b's. La comprobación de si el número de c's es par termina de manera negativa porque el número de c's es impar.

A continuación vamos a desarrollar paso a paso la computación correspondiente a la configuración no determinista $(\{q_0\}, abc)$:

Paso 1: $\lambda^*(\{s_0\}, abc) = \lambda^*(\lambda(s_0, a) \cup \lambda(s_1, a) \cup \lambda(s_3, a) \cup \lambda(s_5, a), bc) = \lambda^*(\{s_2, s_3, s_5\}, bc)$.
Se ha de tener en cuenta que Itxidura-$\varepsilon^*(\{s_0\}, \lambda) = \{s_0, s_2, s_3, s_5\}$.

Paso 2: $\lambda^*(\{s_2, s_3, s_5\}, bc) = \lambda^*(\lambda(s_2, b) \cup \lambda(s_3, b) \cup \lambda(s_5, b), c) = \lambda^*(\{s_2, s_4, s_5\}, c)$. En este caso se ha de tenido en cuenta que Itxidura-$\varepsilon^*(\{s_2, s_3, s_5\}, \lambda) = \{s_2, s_3, s_5\}$.

Paso 3: $\lambda^*(\{s_2, s_4, s_5\}, c) = \lambda^*(\{s_2, s_4, s_5\}, c\varepsilon) = \lambda^*(\lambda(s_2, c) \cup \lambda(s_4, c) \cup \lambda(s_5, c), \varepsilon) = \lambda^*(\{s_2, s_4, s_6\}, \varepsilon)$.
Se ha de tenido en cuenta que Itxidura-$\varepsilon^*(\{s_2, s_4, s_5\}, \lambda) = \{s_2, s_4, s_5\}$.

Paso 4: $\lambda^*(\{s_3, s_4, s_6\}, \varepsilon) = \{s_2, s_4, s_6\}$ puesto que Itxidura-$\varepsilon^*(\{s_2, s_4, s_6\}, \lambda) = \{s_2, s_4, s_6\}$.

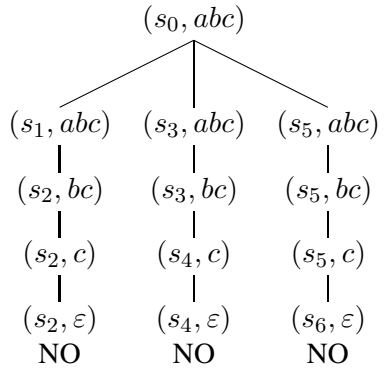
Como la computación termina en el conjunto de estados $\{s_2, s_4, s_6\}$ y como la intersección entre $\{s_2, s_4, s_6\}$ y el conjunto Y es vacía, sabemos que la palabra no es del lenguaje (ni a, ni b ni c aparece un número par de veces).

Esa computación se puede mostrar gráficamente como la siguiente secuencia de configuraciones no deterministas:

$$
\begin{array}{c}
(\{s_0\}, abc) \equiv_{\text{Itxidura-}\varepsilon} (\{s_0, s_1, s_3, s_5\}, abc) \\
| \\
(\{s_2, s_3, s_5\}, bc) \equiv_{\text{Itxidura-}\varepsilon} (\{s_2, s_3, s_5\}, bc) \\
| \\
(\{s_2, s_4, s_5\}, c) \equiv_{\text{Itxidura-}\varepsilon} (\{s_2, s_4, s_5\}, c) \\
| \\
(\{s_2, s_4, s_6\}, \varepsilon) \equiv_{\text{Itxidura-}\varepsilon} (\{s_2, s_4, s_6\}, \varepsilon)
\end{array}
$$

En cada nodo se indica la configuración correspondiente o equivalente teniendo en cuenta el cierre por transiciones ε, es decir, Itxidura-ε^*.

Esa computación se puede mostrar gráficamente como el siguiente árbol de configuraciones deterministas:



Como todas las ramas terminan en una configuración determinista formada por un estado que no pertenece a Y y la palabra vacía ε, la palabra abc no pertenece al lenguaje de este ejemplo.

3.5 Equivalencia entre AFD's, AFND's y ε-AFND's

En este tema hemos diferenciado tres tipos de autómatas finitos: AFD's, AFND's y ε-AFND's.

A la hora de diseñar un autómata finito que defina un lenguaje, habitualmente la opción más cómoda es utilizar ε-AFND's porque los ε-AFND's permiten expresar las propiedades y estructuras de las palabras de manera más fácil, permitiendo un alto grado de modularidad y la posibilidad de analizar propiedades disyuntivas por separado (por ejemplo, mediante el desarrollo de ramas distintas en árboles de configuraciones deterministas). Después de los ε-AFND's, los AFND's son los más cómodos para utilizar, teniendo en cuenta el grado de facilidad para expresar lenguajes, puesto que también permiten un cierto grado de modularidad y la posibilidad de analizar propiedades disyuntivas por separado (aunque dicha separación no es tan clara como en los ε-AFND's). Por último, como en los AFD's no es posible desarrollar árboles en los que distintas propiedades sean analizadas en distintas ramas y como se ha de controlar todas las propiedades mediante un único camino (o una única rama o secuencia de configuraciones deterministas), a veces resulta díficil diseñar un AFD para algunos lenguajes.

Como el grado de facilidad para expresar propiedades es diferente en esos tres modelos de autómatas finitos, se puede pensar que tal vez su capacidad para definir o expresar lenguajes es también distinto. Dicho de otra forma, ¿es posible que lenguajes que no son definibles mediante AFD's o AFND's sean definibles mediante ε-AFND's? De la misma forma, ¿es posible que lenguajes que no son definibles mediante AFD's sean definibles mediante AFND's?

La respuesta es que no en ambos casos. Los tres modelos de autómatas finitos son equivalentes en cuanto a expresividad o capacidad de definir lenguajes. Por tanto, cualquier lenguaje definible mediante un ε-AFND es definible mediante un AFND o AFD. En la misma línea, cualquier lenguaje definible mediante un AFND es definible mediante un AFD.

La equivalencia de los tres modelos de autómatas finitos se puede probar formalmente. Por un lado, existe un algoritmo que dado un AFND, devuelve un AFD que definide o representa el mismo lenguaje. Por otro lado, existe también un procedimiento que permite diseñar un AFND a partir de un ε-AFND. Por último, las transformaciones de AFD a AFND y de AFND a ε-AFND son muy sencillas.

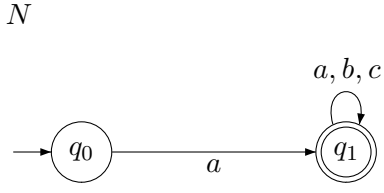
3.5.1 Equivalencia entre AFND's y ε-AFND's

Para probar que los AFND's y los ε-AFND's son equivalentes, hay que probar por una parte, que dado un AFND se puede diseñar un ε-AFND que define exactamente el mismo lenguaje y hay que probar por otra parte que, dado un ε-AFND se puede diseñar un AFND que define el mismo lenguaje.

Para todo AFND se puede diseñar un ε-AFND que define exactamente el mismo lenguaje

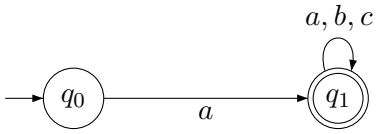
Si nos centramos en los gráficos que definen los autómatas finitos, en un ε-AFND se permite tener transiciones ε pero no es obligatorio que haya flechas etiquetadas con ε. Por ello, observando el gráfico, todo AFND es también un ε-AFND. Por ejemplo, consideremos el siguiente AFND para el lenguaje de las palabras que empiezan por a siendo $A = \{a, b, c\}$:

$$L_3 = \{w \mid w \in A^* \wedge \exists u(u \in A^* \wedge w = au)\}$$



Tal como se acaba de indicar, podríamos decir que este mismo autómata finito es también un ε-AFND aunque no tenga ninguna transición con ε.

E



Si lo planteamos desde el punto de vista de que un AFND N es una quíntupla (Q, A, ν, q_0, Y) donde $\nu : Q \times A \to 2^Q$, el ε-AFND correspondiente E será una quíntupla (Q, A, λ, q_0, Y) donde $\lambda : Q \times (A \cup \{\varepsilon\}) \to 2^Q$ se define como $\lambda(q_j, \alpha) = \nu(q_j, \alpha)$ para todo estado $q_j \in Q$ y todo símbolo α perteneciente al alfabeto A y, además, $\lambda(q_j, \varepsilon) = \varnothing$ para todo estado $q_j \in Q$. Volviendo al ejemplo de las palabras que empiezan por a, tendríamos que ν se define de la siguiente forma:

ν	a	b	c
q_0	$\{q_1\}$	$\varnothing$	$\varnothing$
q_1	$\{q_1\}$	$\{q_1\}$	$\{q_1\}$

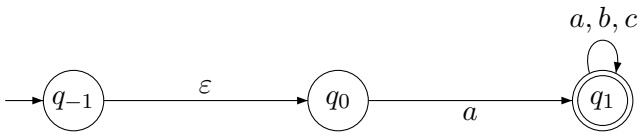
mientras que λ se define de la siguiente forma:

λ	a	b	c	ε
q_0	$\{q_1\}$	$\varnothing$	$\varnothing$	$\varnothing$
q_1	$\{q_1\}$	$\{q_1\}$	$\{q_1\}$	$\varnothing$

Aunque gráficamente el AFND N y el ε-AFND E de este ejemplo son idénticos, las tablas que definen ν y λ son distintas y ahí se ve explícitamente que ν corresponde a un AFND y que λ corresponde a un ε-AFND.

De todas formas, si dado un AFND gráficamente al transformarlo en ε-AFND queremos que en el gráfico aparezca al menos una transición ε, hay un truco para hacerlo de manera sistemática. El truco es añadir un nuevo estado q_{-1} que haga de estado inicial e insertar una transición ε desde ese estado nuevo al estado que antes era inicial (es decir, de q_{-1} a q_0). Para el AFND correspondiente a las palabras que empiezan por a, el ε-AFND E' obtenido así sería el siguiente:

E'



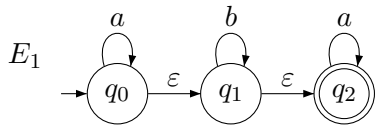
La tabla de la función λ correspondiente a E' sería la siguiente:

λ	a	b	c	ε
q_{-1}	$\varnothing$	$\varnothing$	$\varnothing$	$\{q_0\}$
q_0	$\{q_1\}$	$\varnothing$	$\varnothing$	$\varnothing$
q_1	$\{q_1\}$	$\{q_1\}$	$\{q_1\}$	$\varnothing$

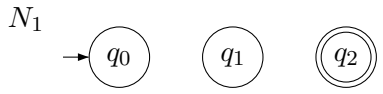
Para todo ε-AFND se puede diseñar un AFND que define exactamente el mismo lenguaje

Dado un ε-AFND de la forma (Q, A, λ, q_0, Y), el AFND correspondiente tendrá Q como conjunto de estados, A como alfabeto y q_0 como estado inicial. Además para el estado q_0, si Cierre-$\varepsilon^*(\{q_0\}, \lambda) \cap Y \neq \varnothing$, entonces q_0 ha de pertenecer al conjunto Y' de los estados que responden "Sí" en el AFND que se está construyendo. Es decir, que si desde q_0 se puede llegar a un estado de Y siguiendo solo transiciones ε, entonces q_0 ha de pertenecer a Y'. Esta peculiaridad relacionada con el estado inicial q_0 es necesaria para poder generar un AFND que responda correctamente a la palabra vacía ε. Por último, para cada estado q_j de Q y cada símbolo α de A, tendremos que $\nu(q_j, \alpha) = \lambda^*(\{q_j\}, \alpha)$. Nótese que ν es del tipo $Q \times A \to 2^Q$ mientras que λ^* es del tipo $2^Q \times A^* \to 2^Q$. Ello supone que en $\lambda^*(\{q_j\}, \alpha)$ realmente tenemos $\lambda^*(\{q_j\}, \alpha\varepsilon)$ y, por consiguiente, primero se calcula a qué estados se llega mediante Cierre-ε^* para $\{q_j\}$, luego se avanza leyendo α para cada estado de Cierre-$\varepsilon^*(\{q_j\})$ y además se vuelve a aplicar Cierre-ε^* para esos estados alcanzados al leer α. El AFND obtenido será una quíntupla de la forma (Q, A, ν, q_0, Y') donde ν se define tal como se ha indicado y el conjunto Y' puede ser igual a Y o puede ser igual a $Y \cup \{q_0\}$ teniendo en cuenta el criterio indicado anteriormente. En cualquier caso se cumple $Y \subseteq Y'$.

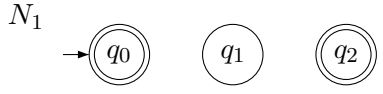
A continuación retomamos los dos ejemplos de ε-AFND's, E_1 y E_2, presentados al comienzo de la anterior sección. En primer lugar vamos a obtener el AFND correspondiente a E_1:



Para empezar sabemos que el nuevo AFND N_1 va a tener los mismos estados:



Como Cierre-$\varepsilon^*(\{q_0\}, \lambda) = \{q_0, q_1, q_2\}$ y, por tanto, Cierre-$\varepsilon^*(\{q_0\}, \lambda) \cap Y \neq \varnothing$, tenemos que q_0 pertenece a Y', es decir, va a tener doble círculo:



A continuación se han de calcular las transiciones de N_1, es decir, se ha de calcular ν.

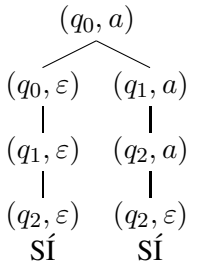
- $\nu(q_0, a) = \lambda^*(\{q_0\}, a) = \lambda^*(\lambda(q_0, a) \cup \lambda(q_1, a) \cup \lambda(q_2, a), \varepsilon) = \lambda^*(\{q_0\} \cup \{q_1\} \cup \{q_2\}, \varepsilon) = \lambda^*(\{q_0, q_1, q_2\}, \varepsilon) =$ Cierre-$\varepsilon^*(\{q_0, q_1, q_2\}, \lambda) = \{q_0, q_1, q_2\}$.

  Esto se puede realizar más gráficamente calculando las computaciones como secuencias de configuraciones no deterministas o como árboles de configuraciones deterministas.

$$
\begin{array}{c}
(\{q_0\}, a) \equiv_{\text{Cierre-}\varepsilon^*} (\{q_0, q_1, q_2\}, a) \\
| \\
(\{q_0, q_2\}, \varepsilon) \equiv_{\text{Cierre-}\varepsilon^*} (\{q_0, q_1, q_2\}, \varepsilon) \\
| \\
(\{q_0, q_1, q_2\}, \varepsilon) \equiv_{\text{Cierre-}\varepsilon^*} (\{q_0, q_1, q_2\}, \varepsilon)
\end{array}
$$

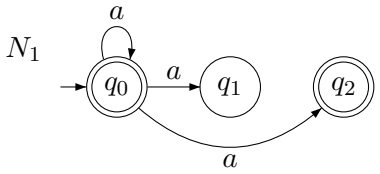
En cada nodo se indica la configuración correspondiente o equivalente teniendo en cuenta el cierre por transiciones ε, es decir, Cierre-ε^*.

Esa computación se puede mostrar gráficamente como el siguiente árbol de configuraciones deterministas:



Si se opta por realizar el árbol, hay que quedarse solo con las ramas que terminan en ε (aunque sean ramas que responden que "No") y coger los estados por los que se pasa en esas ramas (no hay que coger el estado de la raíz). En este caso obtenemos $\{q_0, q_1, q_2\}$. Ahí aparece q_0 porque en la rama de la izquierda aparece, aparte de como raíz, también dentro de la rama (como nodo interno).

Esto supone que hay que añadir las siguientes transiciones:



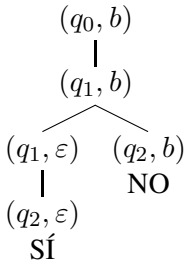
- $\nu(q_0, b) = \lambda^*(\{q_0\}, b) = \lambda^*(\lambda(q_0, b) \cup \lambda(q_1, b) \cup \lambda(q_2, b), \varepsilon) = \lambda^*(\varnothing \cup \{q_1\} \cup \varnothing, \varepsilon) = \lambda^*(\{q_1\}, \varepsilon) =$ Cierre-$\varepsilon^*(\{q_1\}, \lambda) = \{q_1, q_2\}$.

  Esto se puede realizar más gráficamente calculando las computaciones como secuencias de configuraciones no deterministas o como árboles de configuraciones deterministas.

$$
\begin{array}{c}
(\{q_0\}, b) \equiv_{\text{Cierre-}\varepsilon^*} (\{q_0, q_1, q_2\}, b) \\
| \\
(\{q_1\}, \varepsilon) \equiv_{\text{Cierre-}\varepsilon^*} (\{q_1, q_2\}, \varepsilon)
\end{array}
$$

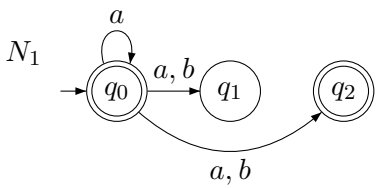
En cada nodo se indica la configuración correspondiente o equivalente teniendo en cuenta el cierre por transiciones ε, es decir, Cierre-ε^*.

Esa computación se puede mostrar gráficamente como el siguiente árbol de configuraciones deterministas:



Si se opta por realizar el árbol, hay que quedarse solo con las ramas que terminan en ε (aunque sean ramas que responden que "No") y coger los estados por los que se pasa en esas ramas (no hay que coger el estado de la raíz). En este caso obtenemos $\{q_1, q_2\}$.

Esto supone que hay que añadir las siguientes transiciones:



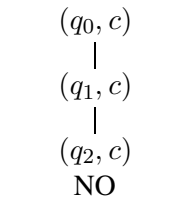
- $\nu(q_0, c) = \lambda^*(\{q_0\}, c) = \lambda^*(\lambda(q_0, c) \cup \lambda(q_1, c) \cup \lambda(q_2, c), \varepsilon) = \lambda^*(\varnothing \cup \varnothing \cup \varnothing, \varepsilon) = \lambda^*(\varnothing, \varepsilon) =$ Cierre-$\varepsilon^*(\varnothing, \lambda) = \varnothing$.

  Esto se puede realizar más gráficamente calculando las computaciones como secuencias de configuraciones no deterministas o como árboles de configuraciones deterministas.

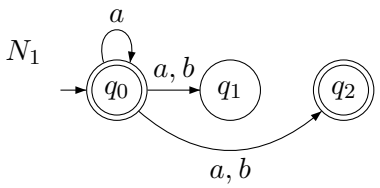
$$\begin{array}{c} (\{q_0\}, c) \equiv_{\text{Cierre-}\varepsilon^*} (\{q_0, q_1, q_2\}, c) \\ | \\ (\varnothing, \varepsilon) \equiv_{\text{Cierre-}\varepsilon^*} (\varnothing, \varepsilon) \end{array}$$

  En cada nodo se indica la configuración correspondiente o equivalente teniendo en cuenta el cierre por transiciones ε, es decir, Cierre-ε^*.

  Esa computación se puede mostrar gráficamente como el siguiente árbol de configuraciones deterministas:



  Si se opta por realizar el árbol, hay que quedarse solo con las ramas que terminan en ε (aunque sean ramas que responden que "No") y coger los estados por los que se pasa en esas ramas (no hay que coger el estado de la raíz). En este caso no obtenemos ningún estado y por tanto no hay que añadir ninguna flecha.



- $\nu(q_1, a) = \lambda^*(\{q_1\}, a) = \lambda^*(\lambda(q_1, a) \cup \lambda(q_2, a), \varepsilon) = \lambda^*(\varnothing \cup \{q_2\}, \varepsilon) = \lambda^*(\{q_2\}, \varepsilon) =$ Cierre-$\varepsilon^*(\{q_2\}, \lambda) = \{q_2\}$.

  Esto se puede realizar más gráficamente calculando las computaciones como secuencias de configuraciones no deterministas o como árboles de configuraciones deterministas.

$$\begin{array}{c}(\{q_1\}, a) \equiv_{\text{Cierre-}\varepsilon^*} (\{q_1, q_2\}, a) \\ | \\ (\{q_2\}, \varepsilon) \equiv_{\text{Cierre-}\varepsilon^*} (\{q_2\}, \varepsilon)\end{array}$$

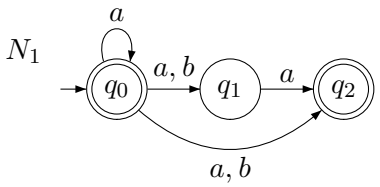
  En cada nodo se indica la configuración correspondiente o equivalente teniendo en cuenta el cierre por transiciones ε, es decir, Cierre-ε^*.

  Esa computación se puede mostrar gráficamente como el siguiente árbol de configuraciones deterministas:

$$\begin{array}{c}(q_1, a) \\ | \\ (q_2, a) \\ | \\ (q_2, \varepsilon) \\ \text{SÍ}\end{array}$$

  Si se opta por realizar el árbol, hay que quedarse solo con las ramas que terminan en ε (aunque sean ramas que responden que "No") y coger los estados por los que se pasa en esas ramas (no hay que coger el estado de la raíz). En este caso obtenemos $\{q_2\}$.

  Esto supone que hay que añadir la siguiente transición:



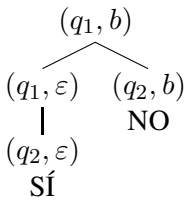
- $\nu(q_1, b) = \lambda^*(\{q_1\}, b) = \lambda^*(\lambda(q_1, b) \cup \lambda(q_2, b), \varepsilon) = \lambda^*(\{q_1\} \cup \varnothing, \varepsilon) = \lambda^*(\{q_1\}, \varepsilon) =$ Cierre-$\varepsilon^*(\{q_1\}, \lambda) = \{q_1, q_2\}$.

  Esto se puede realizar más gráficamente calculando las computaciones como secuencias de configuraciones no deterministas o como árboles de configuraciones deterministas.

$$\begin{array}{c}(\{q_1\}, b) \equiv_{\text{Cierre-}\varepsilon^*} (\{q_1, q_2\}, b) \\ | \\ (\{q_1\}, \varepsilon) \equiv_{\text{Cierre-}\varepsilon^*} (\{q_1, q_2\}, \varepsilon)\end{array}$$

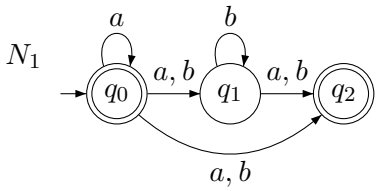
En cada nodo se indica la configuración correspondiente o equivalente teniendo en cuenta el cierre por transiciones ε, es decir, Cierre-ε^*.

Esa computación se puede mostrar gráficamente como el siguiente árbol de configuraciones deterministas:



Si se opta por realizar el árbol, hay que quedarse solo con las ramas que terminan en ε (aunque sean ramas que responden que "No") y coger los estados por los que se pasa en esas ramas (no hay que coger el estado de la raíz). En este caso obtenemos q_1 y q_2.

Esto supone que hay que añadir la siguientes transiciones:



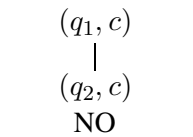
- $\nu(q_1, c) = \lambda^*(\{q_1\}, c) = \lambda^*(\lambda(q_1, c) \cup \lambda(q_2, c), \varepsilon) = \lambda^*(\varnothing \cup \varnothing, \varepsilon) = \lambda^*(\varnothing, \varepsilon) = \text{Cierre-}\varepsilon^*(\varnothing, \lambda) = \varnothing$.

  Esto se puede realizar más gráficamente calculando las computaciones como secuencias de configuraciones no deterministas o como árboles de configuraciones deterministas.

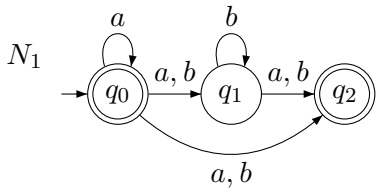
$$\begin{array}{c} (\{q_1\}, c) \equiv_{\text{Cierre-}\varepsilon^*} (\{q_1, q_2\}, c) \\ | \\ (\varnothing, \varepsilon) \equiv_{\text{Cierre-}\varepsilon^*} (\varnothing, \varepsilon) \end{array}$$

  En cada nodo se indica la configuración correspondiente o equivalente teniendo en cuenta el cierre por transiciones ε, es decir, Cierre-ε^*.

  Esa computación se puede mostrar gráficamente como el siguiente árbol de configuraciones deterministas:



  Si se opta por realizar el árbol, hay que quedarse solo con las ramas que terminan en ε (aunque sean ramas que responden que "No") y coger los estados por los que se pasa en esas ramas (no hay que coger el estado de la raíz). En este caso no obtenemos ningún estado y por tanto no hay que añadir ninguna flecha.



- $\nu(q_2, a) = \lambda^*(\{q_2\}, a) = \lambda^*(\lambda(q_2, a), \varepsilon) = \lambda^*(\{q_2\}, \varepsilon) = \mathsf{Cierre\text{-}}\varepsilon^*(\{q_2\}, \lambda) = \{q_2\}$.

  Esto se puede realizar más gráficamente calculando las computaciones como secuencias de configuraciones no deterministas o como árboles de configuraciones deterministas.

$$\begin{array}{c}(\{q_2\}, a) \equiv_{\mathsf{Cierre\text{-}}\varepsilon^*} (\{q_2\}, a) \\ | \\ (\{q_2\}, \varepsilon) \equiv_{\mathsf{Cierre\text{-}}\varepsilon^*} (\{q_2\}, \varepsilon)\end{array}$$

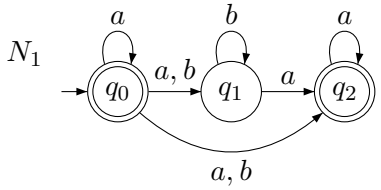
  En cada nodo se indica la configuración correspondiente o equivalente teniendo en cuenta el cierre por transiciones ε, es decir, $\mathsf{Cierre\text{-}}\varepsilon^*$.

  Esa computación se puede mostrar gráficamente como el siguiente árbol de configuraciones deterministas:

$$\begin{array}{c}(q_2, a) \\ | \\ (q_2, \varepsilon) \\ \text{SÍ}\end{array}$$

  Si se opta por realizar el árbol, hay que quedarse solo con las ramas que terminan en ε (aunque sean ramas que responden que "No") y coger los estados por los que se pasa en esas ramas (no hay que coger el estado de la raíz). En este caso obtenemos $\{q_2\}$.

  Esto supone que hay que añadir la siguiente transición:



- $\nu(q_2, b) = \lambda^*(\{q_2\}, b) = \lambda^*(\lambda(q_2, b), \varepsilon) = \lambda^*(\varnothing, \varepsilon) = \mathsf{Cierre\text{-}}\varepsilon^*(\varnothing, \lambda) = \varnothing$.

  Esto se puede realizar más gráficamente calculando las computaciones como secuencias de configuraciones no deterministas o como árboles de configuraciones deterministas.

$$\begin{array}{c}(\{q_2\}, b) \equiv_{\mathsf{Cierre\text{-}}\varepsilon^*} (\{q_2\}, b) \\ | \\ (\varnothing, \varepsilon) \equiv_{\mathsf{Cierre\text{-}}\varepsilon^*} (\varnothing, \varepsilon)\end{array}$$

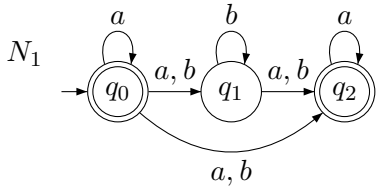
  En cada nodo se indica la configuración correspondiente o equivalente teniendo en cuenta el cierre por transiciones ε, es decir, $\mathsf{Cierre\text{-}}\varepsilon^*$.

  Esa computación se puede mostrar gráficamente como el siguiente árbol de configuraciones deterministas:

(q_2, b)
NO

Si se opta por realizar el árbol, hay que quedarse solo con las ramas que terminan en ε (aunque sean ramas que responden que "No") y coger los estados por los que se pasa en esas ramas (no hay que coger el estado de la raíz). En este caso no obtenemos ningún estado.

Esto supone que hay que añadir la siguientes transiciones:



- $\nu(q_2, c) = \lambda^*(\{q_2\}, c) = \lambda^*(\lambda(q_2, c), \varepsilon) = \lambda^*(\varnothing, \varepsilon) = \mathsf{Cierre\text{-}}\varepsilon^*(\varnothing, \lambda) = \varnothing$.

  Esto se puede realizar más gráficamente calculando las computaciones como secuencias de configuraciones no deterministas o como árboles de configuraciones deterministas.

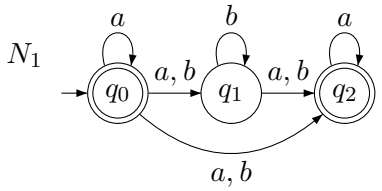
$$\begin{array}{c}(\{q_2\}, c) \equiv_{\mathsf{Cierre\text{-}}\varepsilon^*} (\{q_2\}, c) \\ | \\ (\varnothing, \varepsilon) \equiv_{\mathsf{Cierre\text{-}}\varepsilon^*} (\varnothing, \varepsilon)\end{array}$$

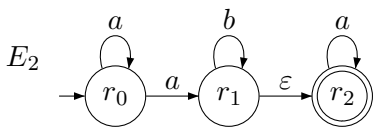
  En cada nodo se indica la configuración correspondiente o equivalente teniendo en cuenta el cierre por transiciones ε, es decir, $\mathsf{Cierre\text{-}}\varepsilon^*$.

  Esa computación se puede mostrar gráficamente como el siguiente árbol de configuraciones deterministas:

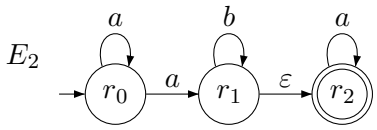
(q_2, c)
NO

Si se opta por realizar el árbol, hay que quedarse solo con las ramas que terminan en ε (aunque sean ramas que responden que "No") y coger los estados por los que se pasa en esas ramas (no hay que coger el estado de la raíz). En este caso no obtenemos ningún estado y por tanto no hay que añadir ninguna flecha.

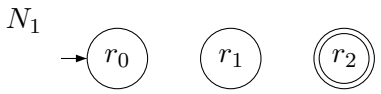




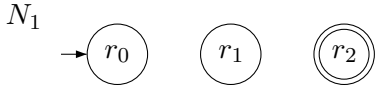
En segundo lugar vamos a obtener el AFND correspondiente a E_2:



Para empezar sabemos que el nuevo AFND N_2 va a tener los mismos estados:



Como Cierre-$\varepsilon^*(\{r_0\}, \lambda) = \{q_0\}$ y, por tanto, Cierre-$\varepsilon^*(\{q_0\}, \lambda) \cap Y = \varnothing$, tenemos que q_0 no pertenece a Y', es decir, no va a tener doble círculo:



A continuación se han de calcular las transiciones de N_2, es decir, se ha de calcular ν.

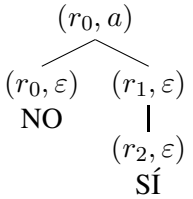
- $\nu(r_0, a) = \lambda^*(\{r_0\}, a) = \lambda^*(\lambda(q_0, a), \varepsilon) = \lambda^*(\{r_0, r_1\}, \varepsilon) = \lambda^*(\{r_0, r_1\}, \varepsilon) =$ Cierre-$\varepsilon^*(\{r_0, r_1\}, \lambda) = \{r_0, r_1\}$.

  Esto se puede realizar más gráficamente calculando las computaciones como secuencias de configuraciones no deterministas o como árboles de configuraciones deterministas.

$$
\begin{array}{c}
(\{r_0\}, a) \equiv_{\text{Cierre-}\varepsilon^*} (\{r_0\}, a) \\
\Big| \\
(\{r_0, r_1\}, \varepsilon) \equiv_{\text{Cierre-}\varepsilon^*} (\{r_0, r_1, r_2\}, \varepsilon) \\
\Big| \\
(\{r_0, r_1, r_2\}, \varepsilon) \equiv_{\text{Cierre-}\varepsilon^*} (\{r_0, r_1, r_2\}, \varepsilon)
\end{array}
$$

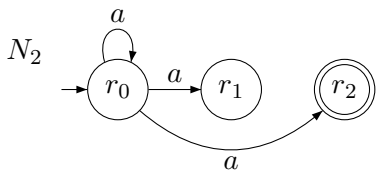
  En cada nodo se indica la configuración correspondiente o equivalente teniendo en cuenta el cierre por transiciones ε, es decir, Cierre-ε^*.

  Esa computación se puede mostrar gráficamente como el siguiente árbol de configuraciones deterministas:



Si se opta por realizar el árbol, hay que quedarse solo con las ramas que terminan en ε (aunque sean ramas que responden que "No") y coger los estados por los que se pasa en esas ramas (no hay que coger el estado de la raíz). En este caso obtenemos $\{r_0, r_1, r_2\}$. Ahí aparece r_0 porque en la rama de la izquierda aparece, aparte de como raíz, también dentro de la rama (como nodo interno).

Esto supone que hay que añadir las siguientes transiciones:



- $\nu(r_0, b) = \lambda^*(\{r_0\}, b) = \lambda^*(\lambda(r_0, b), \varepsilon) = \lambda^*(\varnothing, \varepsilon) = \mathsf{Cierre\text{-}}\varepsilon^*(\varnothing, \lambda) = \varnothing$.

  Esto se puede realizar más gráficamente calculando las computaciones como secuencias de configuraciones no deterministas o como árboles de configuraciones deterministas.

$$\begin{array}{c}(\{r_0\}, b) \equiv_{\mathsf{Cierre\text{-}}\varepsilon^*} (\{r_0\}, b) \\ | \\ (\varnothing, \varepsilon) \equiv_{\mathsf{Cierre\text{-}}\varepsilon^*} (\varnothing, \varepsilon)\end{array}$$

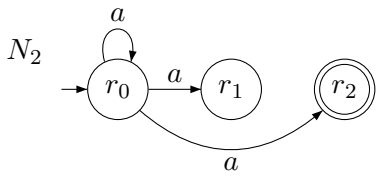
  En cada nodo se indica la configuración correspondiente o equivalente teniendo en cuenta el cierre por transiciones ε, es decir, $\mathsf{Cierre\text{-}}\varepsilon^*$.

  Esa computación se puede mostrar gráficamente como el siguiente árbol de configuraciones deterministas:

$$\begin{array}{c}(r_0, b) \\ \text{NO}\end{array}$$

  Si se opta por realizar el árbol, hay que quedarse solo con las ramas que terminan en ε (aunque sean ramas que responden que "No") y coger los estados por los que se pasa en esas ramas (no hay que coger el estado de la raíz). En este caso no obtenemos ningún estado.

  Esto supone que no hay que añadir transiciones:



- $\nu(r_0, c) = \lambda^*(\{r_0\}, c) = \lambda^*(\lambda(r_0, c), \varepsilon) = \lambda^*(\varnothing\varepsilon) = \mathsf{Cierre\text{-}}\varepsilon^*(\varnothing, \lambda) = \varnothing$.

  Esto se puede realizar más gráficamente calculando las computaciones como secuencias de configuraciones no deterministas o como árboles de configuraciones deterministas.

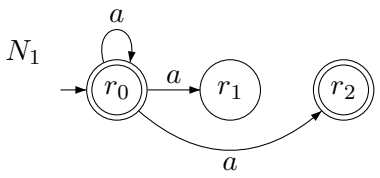
$$(\{r_0\}, c) \equiv_{\text{Cierre-}\varepsilon^*} (\{r_0\}, c)$$
$$\vert$$
$$(\varnothing, \varepsilon) \equiv_{\text{Cierre-}\varepsilon^*} (\varnothing, \varepsilon)$$

En cada nodo se indica la configuración correspondiente o equivalente teniendo en cuenta el cierre por transiciones ε, es decir, Cierre-ε^*.

Esa computación se puede mostrar gráficamente como el siguiente árbol de configuraciones deterministas:

$$(r_0, c)$$
NO

Si se opta por realizar el árbol, hay que quedarse solo con las ramas que terminan en ε (aunque sean ramas que responden que "No") y coger los estados por los que se pasa en esas ramas (no hay que coger el estado de la raíz). En este caso no obtenemos ningún estado y por tanto no hay que añadir ninguna flecha.



- $\nu(r_1, a) = \lambda^*(\{r_1\}, a) = \lambda^*(\lambda(r_1, a) \cup \lambda(r_2, a), \varepsilon) = \lambda^*(\varnothing \cup \{r_2\}, \varepsilon) = \lambda^*(\{r_2\}, \varepsilon) =$ Cierre-$\varepsilon^*(\{r_2\}, \lambda) = \{r_2\}$.

  Esto se puede realizar más gráficamente calculando las computaciones como secuencias de configuraciones no deterministas o como árboles de configuraciones deterministas.

$$(\{r_1\}, a) \equiv_{\text{Cierre-}\varepsilon^*} (\{r_1, r_2\}, a)$$
$$\vert$$
$$(\{r_2\}, \varepsilon) \equiv_{\text{Cierre-}\varepsilon^*} (\{r_2\}, \varepsilon)$$

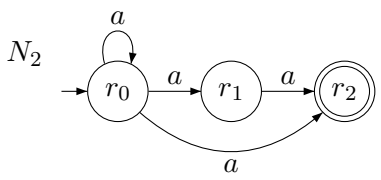
  En cada nodo se indica la configuración correspondiente o equivalente teniendo en cuenta el cierre por transiciones ε, es decir, Cierre-ε^*.

  Esa computación se puede mostrar gráficamente como el siguiente árbol de configuraciones deterministas:

$$(r_1, a)$$
$$\vert$$
$$(r_2, a)$$
$$\vert$$
$$(r_2, \varepsilon)$$
SÍ

  Si se opta por realizar el árbol, hay que quedarse solo con las ramas que terminan en ε (aunque sean ramas que responden que "No") y coger los estados por los que se pasa en esas ramas (no hay que coger el estado de la raíz). En este caso obtenemos $\{r_2\}$.

  Esto supone que hay que añadir la siguiente transición:



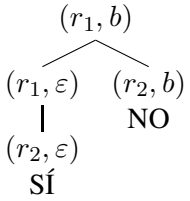
- $\nu(r_1, b) = \lambda^*(\{r_1\}, b) = \lambda^*(\lambda(r_1, b) \cup \lambda(r_2, b), \varepsilon) = \lambda^*(\{r_1\} \cup \varnothing, \varepsilon) = \lambda^*(\{r_1\}, \varepsilon) = \textsf{Cierre-}\varepsilon^*(\{r_1\}, \lambda) = \{r_1, r_2\}$.

  Esto se puede realizar más gráficamente calculando las computaciones como secuencias de configuraciones no deterministas o como árboles de configuraciones deterministas.

$$\begin{array}{c}(\{r_1\}, b) \equiv_{\textsf{Cierre-}\varepsilon^*} (\{r_1, r_2\}, b) \\ | \\ (\{r_1\}, \varepsilon) \equiv_{\textsf{Cierre-}\varepsilon^*} (\{r_1, r_2\}, \varepsilon)\end{array}$$

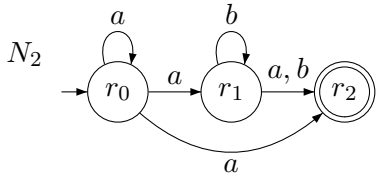
  En cada nodo se indica la configuración correspondiente o equivalente teniendo en cuenta el cierre por transiciones ε, es decir, Cierre-ε^*.

  Esa computación se puede mostrar gráficamente como el siguiente árbol de configuraciones deterministas:



  Si se opta por realizar el árbol, hay que quedarse solo con las ramas que terminan en ε y coger los estados por los que se pasa en esas ramas (no hay que coger el estado de la raíz). En este caso obtenemos r_1 y r_2.

  Esto supone que hay que añadir la siguientes transiciones:



- $\nu(r_1, c) = \lambda^*(\{r_1\}, c) = \lambda^*(\lambda(r_1, c) \cup \lambda(r_2, c), \varepsilon) = \lambda^*(\varnothing \cup \varnothing, \varepsilon) = \lambda^*(\varnothing, \varepsilon) = \textsf{Cierre-}\varepsilon^*(\varnothing, \lambda) = \varnothing$.

  Esto se puede realizar más gráficamente calculando las computaciones como secuencias de configuraciones no deterministas o como árboles de configuraciones deterministas.

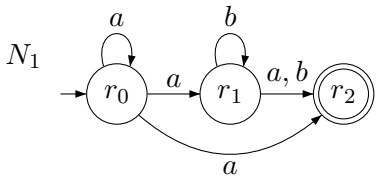
$$\begin{array}{c}(\{r_1\}, c) \equiv_{\textsf{Cierre-}\varepsilon^*} (\{r_1, r_2\}, c) \\ | \\ (\varnothing, \varepsilon) \equiv_{\textsf{Cierre-}\varepsilon^*} (\varnothing, \varepsilon)\end{array}$$

En cada nodo se indica la configuración correspondiente o equivalente teniendo en cuenta el cierre por transiciones ε, es decir, Cierre-ε^*.

Esa computación se puede mostrar gráficamente como el siguiente árbol de configuraciones deterministas:

$$\begin{array}{c}(r_1, c)\\ |\\ (r_2, c)\\ \text{NO}\end{array}$$

Si se opta por realizar el árbol, hay que quedarse solo con las ramas que terminan en ε y coger los estados por los que se pasa en esas ramas (no hay que coger el estado de la raíz). En este caso no obtenemos ningún estado y por tanto no hay que añadir ninguna flecha.



- $\nu(r_2, a) = \lambda^*(\{r_2\}, a) = \lambda^*(\lambda(r_2, a), \varepsilon) = \lambda^*(\{r_2\}, \varepsilon) = \text{Cierre-}\varepsilon^*(\{r_2\}, \lambda) = \{r_2\}$.

  Esto se puede realizar más gráficamente calculando las computaciones como secuencias de configuraciones no deterministas o como árboles de configuraciones deterministas.

$$\begin{array}{c}(\{r_2\}, a) \equiv_{\text{Cierre-}\varepsilon^*} (\{r_2\}, a)\\ |\\ (\{r_2\}, \varepsilon) \equiv_{\text{Cierre-}\varepsilon^*} (\{r_2\}, \varepsilon)\end{array}$$

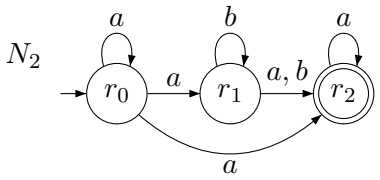
  En cada nodo se indica la configuración correspondiente o equivalente teniendo en cuenta el cierre por transiciones ε, es decir, Cierre-ε^*.

  Esa computación se puede mostrar gráficamente como el siguiente árbol de configuraciones deterministas:

$$\begin{array}{c}(r_2, a)\\ |\\ (r_2, \varepsilon)\\ \text{SÍ}\end{array}$$

  Si se opta por realizar el árbol, hay que quedarse solo con las ramas que terminan en ε y coger los estados por los que se pasa en esas ramas (no hay que coger el estado de la raíz). En este caso obtenemos $\{r_2\}$.

  Esto supone que hay que añadir la siguiente transición:



- $\nu(r_2, b) = \lambda^*(\{r_2\}, b) = \lambda^*(\lambda(r_2, b), \varepsilon) = \lambda^*(\varnothing, \varepsilon) = \mathsf{Cierre\text{-}}\varepsilon^*(\varnothing, \lambda) = \varnothing$.

  Esto se puede realizar más gráficamente calculando las computaciones como secuencias de configuraciones no deterministas o como árboles de configuraciones deterministas.

$$\begin{array}{c}(\{r_2\}, b) \equiv_{\mathsf{Cierre\text{-}}\varepsilon^*} (\{r_2\}, b)\\ \vert \\ (\varnothing, \varepsilon) \equiv_{\mathsf{Cierre\text{-}}\varepsilon^*} (\varnothing, \varepsilon)\end{array}$$

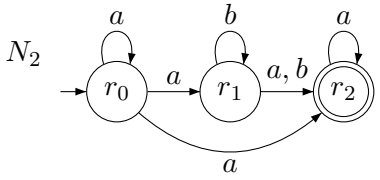
  En cada nodo se indica la configuración correspondiente o equivalente teniendo en cuenta el cierre por transiciones ε, es decir, $\mathsf{Cierre\text{-}}\varepsilon^*$.

  Esa computación se puede mostrar gráficamente como el siguiente árbol de configuraciones deterministas:

$$\begin{array}{c}(r_2, b)\\ \text{NO}\end{array}$$

  Si se opta por realizar el árbol, hay que quedarse solo con las ramas que terminan en ε y coger los estados por los que se pasa en esas ramas (no hay que coger el estado de la raíz). En este caso no obtenemos ningún estado.

  Esto supone que hay que añadir la siguientes transiciones:



- $\nu(r_2, c) = \lambda^*(\{r_2\}, c) = \lambda^*(\lambda(r_2, c), \varepsilon) = \lambda^*(\varnothing, \varepsilon) = \mathsf{Cierre\text{-}}\varepsilon^*(\varnothing, \lambda) = \varnothing$.

  Esto se puede realizar más gráficamente calculando las computaciones como secuencias de configuraciones no deterministas o como árboles de configuraciones deterministas.

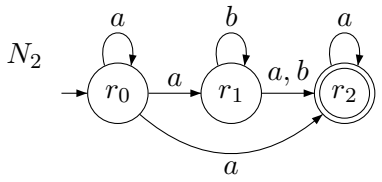
$$\begin{array}{c}(\{r_2\}, c) \equiv_{\mathsf{Cierre\text{-}}\varepsilon^*} (\{r_2\}, c)\\ \vert \\ (\varnothing, \varepsilon) \equiv_{\mathsf{Cierre\text{-}}\varepsilon^*} (\varnothing, \varepsilon)\end{array}$$

  En cada nodo se indica la configuración correspondiente o equivalente teniendo en cuenta el cierre por transiciones ε, es decir, $\mathsf{Cierre\text{-}}\varepsilon^*$.

  Esa computación se puede mostrar gráficamente como el siguiente árbol de configuraciones deterministas:

$$\begin{array}{c}(r_2, c)\\ \text{NO}\end{array}$$

  Si se opta por realizar el árbol, hay que quedarse solo con las ramas que terminan en ε (aunque sean ramas que responden que "No") y coger los estados por los que se pasa en esas ramas (no hay que coger el estado de la raíz). En este caso no obtenemos ningún estado y por tanto no hay que añadir ninguna flecha.



3.5.2 Equivalencia entre AFD's y AFND's

Para probar que los AFD's y los AFND's son equivalentes, hay que probar por una parte, que dado un AFD se puede diseñar un AFND que define exactamente el mismo lenguaje y hay que probar, por otra parte que, dado un AFND se puede diseñar un AFD que define el mismo lenguaje.

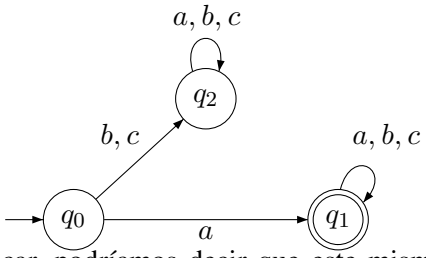
Para todo AFD se puede diseñar un AFND que define exactamente el mismo lenguaje

Si consideramos el asunto desde un punto de vista gráfico, en un AFND se permite que desde un estado salgan más de un arco para un mismo símbolo o que no haya ningún arco para un símbolo pero no es obligatorio, es decir, en un AFND puede haber exactamente un arco (una transición) por cada símbolo y cada estado. Por ello, todo AFD es, gráficamente, también un AFND. Si lo planteamos desde el punto de vista de que un AFD D es una quíntupla (Q, A, δ, q_0, Y) donde $\delta : Q \times A \to Q$, el AFND correspondiente N será una quíntupla (Q, A, ν, q_0, Y) donde $\nu : Q \times A \to 2^Q$ se define como $\nu(q_j, \alpha) = \{\delta(q_j, \alpha)\}$ para todo estado $q_j \in Q$ y todo símbolo α perteneciente al alfabeto A y, además. Por tanto, en este caso particular, para cada símbolo α de A, la función ν devuelve un conjunto con un único estado.

Si nos centramos en los gráficos que definen los autómatas finitos, en un AFND para cada estado se permite tener varias transiciones con un mismo símbolo y también para cada estado se permite no tener ninguna transición para algún símbolo, pero no es obligatorio que ello ocurra. Por tanto, observando el gráfico, todo AFD es también un AFND. Por ejemplo, consideremos el siguiente AFD para el lenguaje L de las palabras que empiezan por a siendo $A = \{a, b, c\}$:

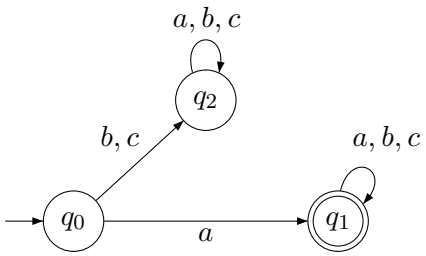
$$L = \{w \mid w \in A^* \wedge \exists u(u \in A^* \wedge w = au)\}$$

D



Tal como se acaba de indicar, podríamos decir que este mismo autómata finito es también un AFND aunque para cada estado y cada símbolo se tenga exactamente una transición.

N



Si lo planteamos desde el punto de vista de que un AFD D es una quíntupla (Q, A, δ, q_0, Y) donde $\delta : Q \times A \rightarrow Q$, el AFND correspondiente N será una quíntupla (Q, A, ν, q_0, Y) donde $\nu : Q \times A \rightarrow 2^Q$ se define como $\nu(q_j, \alpha) = \{\delta(q_j, \alpha)\}$ para todo estado $q_j \in Q$ y todo símbolo α perteneciente al alfabeto A. Volviendo al ejemplo de las palabras que empiezan por a, tendríamos que δ se define de la siguiente forma:

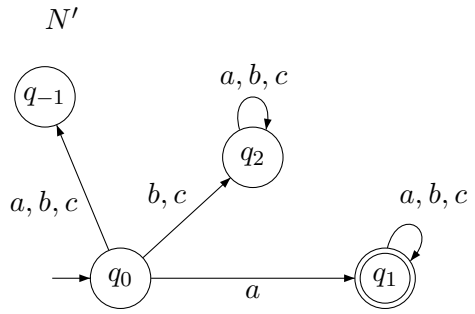
δ	a	b	c
q_0	q_1	q_2	q_2
q_1	q_1	q_1	q_1
q_2	q_2	q_2	q_2

mientras que ν se define de la siguiente forma:

ν	a	b	c
q_0	$\{q_1\}$	$\{q_2\}$	$\{q_2\}$
q_1	$\{q_1\}$	$\{q_1\}$	$\{q_1\}$
q_2	$\{q_2\}$	$\{q_2\}$	$\{q_2\}$

Aunque gráficamente el AFD D y el AFND N de este ejemplo son idénticos, las tablas que definen δ y ν son distintas y ahí se ve explícitamente que δ corresponde a un AFD y que ν corresponde a un AFND.

De todas formas, si dado un AFD gráficamente al transformarlo en AFND queremos que en el gráfico aparezca al menos un estado que tenga más de una transición para algún símbolo y que aparezca al menos un estado que no tenga ninguna transición para algún símbolo, hay un truco para hacerlo de manera sistemática. El truco es añadir un nuevo estado q_{-1} al cual se llega desde el estado inicial con todos los símbolos del alfabeto y del cual no sale ninguna transición. Para el AFD correspondiente a las palabras que empiezan por a, el AFND N' obtenido así sería el siguiente:



La tabla de la función ν correspondiente a N' sería la siguiente:

ν	a	b	c
q_{-1}	$\varnothing$	$\varnothing$	$\varnothing$
q_0	$\{q_{-1}, q_1\}$	$\{q_{-1}, q_2\}$	$\{q_{-1}, q_2\}$
q_1	$\{q_1\}$	$\{q_1\}$	$\{q_1\}$
q_2	$\{q_2\}$	$\{q_2\}$	$\{q_2\}$

Para todo AFND se puede diseñar un AFD que define exactamente el mismo lenguaje

Dado un AFND N de la forma (Q, A, ν, q_0, Y) el AFD (Q', A, δ, r_0, Y') correspondiente se calcula considerando conjuntos de estados:

- El estado inicial r_0 de Q' será el estado etiquetado como $\{q_0\}$.

- Para cada símbolo α perteneciente a A se ha de calcular $\nu^*(\{q_0\}, \alpha)$.
- Cada conjunto $\nu^*(\{q_0\}, \alpha)$ viene a ser un nuevo estado del AFD que se está calculando y habrá un arco desde $\{q_0\}$ (o r_0) a $\nu^*(\{q_0\}, \alpha)$ etiquetado con el símbolo α. Por tanto, $\delta(r_0, \alpha) = \nu^*(\{q_0\}, \alpha)$.
- A partir de los nuevos conjuntos obtenidos mediante el calculo de $\nu^*(\{q_0\}, \alpha)$ correspondiente a cada símbolo α de A, se repetirá el proceso de generación de nuevos estados mientras sea posible generar estados distintos.
- Serán estados de Y', es decir, que responden "Sí" aquellos que contengan algún estado de Y.

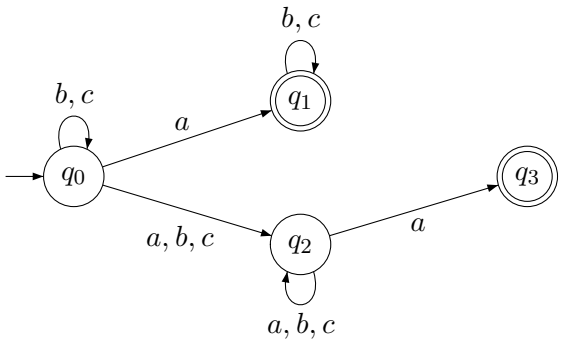
Ejemplo: de AFND a AFD, el lenguaje de las palabras que solo contienen una a y cualquier número de b's y c's

En la Figura 3.31 se muestra un AFND para el lenguaje formado por las palabras que solo contienen una a (pudiendo tener además b's y c's) o que terminan en a. La definición formal de dicho lenguaje sería la siguiente:

$$\{w \mid w \in A^* \wedge (|w|_a = 1 \vee \exists u(u \in A^* \wedge w = ua))\}$$

donde $A = \{a, b, c\}$. Otra posible definición sería:

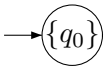
$$((\{b, c\}^*)\{a\}(\{b, c\}^*)) \cup (A^*\{a\})$$



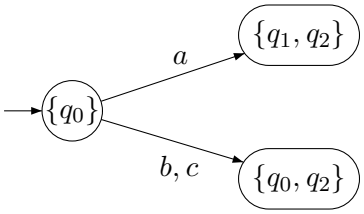
3.31 Irudia: AFND para el lenguaje de las palabras que solo contienen una a o que terminan en a.

A continuación se calculará el AFD correspondiente al AFND de la Figura 3.31. Para ello se irán mostrando los estados que se generan paso a paso hasta obtener el AFD definitivo. Como cada estado estará etiquetado por un conjunto de estados del AFND, el proceso terminará con un renombramiento. Los pasos son los siguientes:

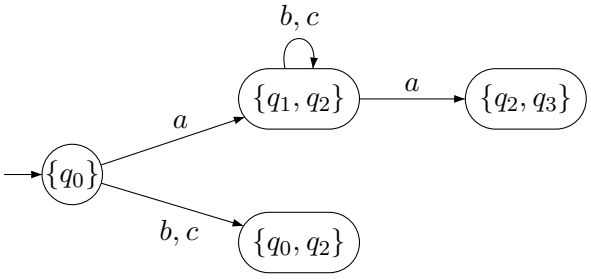
- El estado inicial es siempre $\{q_0\}$. Por tanto, primero generamos un estado con etiqueta $\{q_0\}$ y con la flecha característica de los estados iniciales.



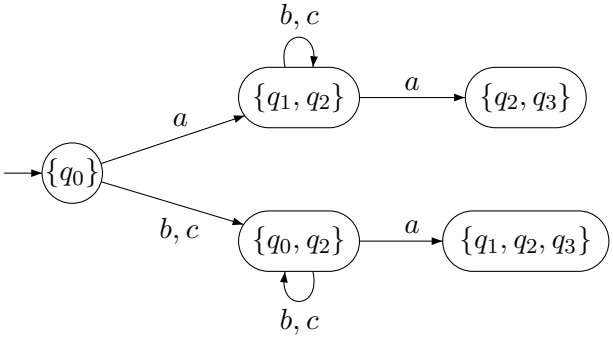
- Ahora tenemos que $\nu^*(\{q_0\}, a) = \{q_1, q_2\}$, $\nu^*(\{q_0\}, b) = \{q_0, q_2\}$ y $\nu^*(\{q_0\}, c) = \{q_0, q_2\}$



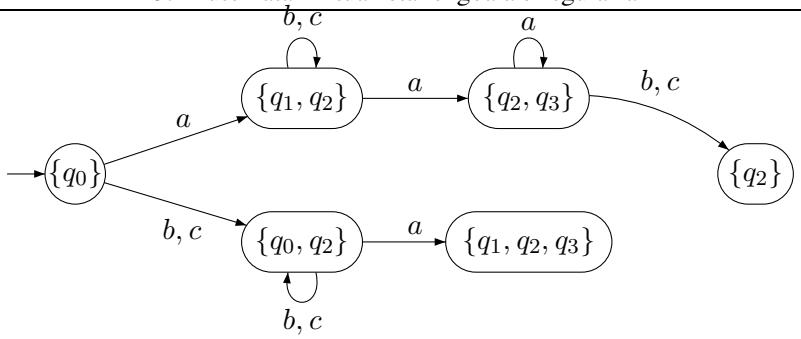
- Nos centramos primero en el estado $\{q_1, q_2\}$ y tenemos que $\nu^*(\{q_1, q_2\}, a) = \nu(q_1, a) \cup \nu(q_2, a) = \varnothing \cup \{q_2, q_3\}$, es decir, $\{q_2, q_3\}$. Por otra parte, $\nu^*(\{q_1, q_2\}, b) = \nu(q_1, b) \cup \nu(q_2, b) = \{q_1\} \cup \{q_2\}$, es decir, $\{q_1, q_2\}$. Por último, $\nu^*(\{q_1, q_2\}, c) = \nu(q_1, c) \cup \nu(q_2, c) = \{q_1\} \cup \{q_2\}$, es decir, $\{q_1, q_2\}$



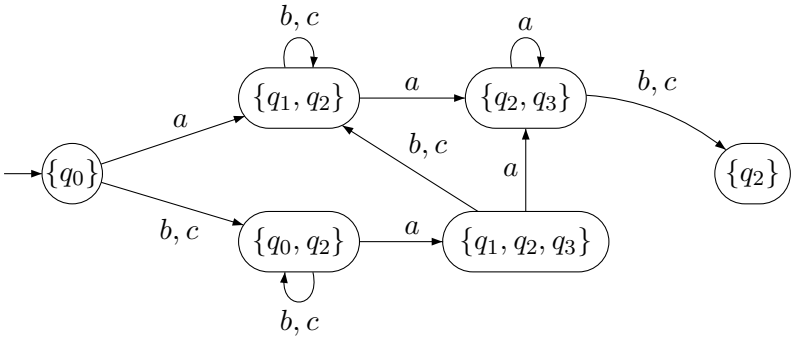
- Nos centramos ahora en el estado $\{q_0, q_2\}$ y tenemos que $\nu^*(\{q_0, q_2\}, a) = \nu(q_0, a) \cup \nu(q_2, a) = \{q_1, q_2\} \cup \{q_2, q_3\}$, es decir, $\{q_1, q_2, q_3\}$. Por otra parte, $\nu^*(\{q_0, q_2\}, b) = \nu(q_0, b) \cup \nu(q_2, b) = \{q_0, q_2\} \cup \{q_2\}$, es decir, $\{q_0, q_2\}$. También tenemos que, $\nu^*(\{q_0, q_2\}, c) = \nu(q_0, c) \cup \nu(q_2, c) = \{q_0, q_2\} \cup \{q_2\}$, es decir, $\{q_0, q_2\}$.



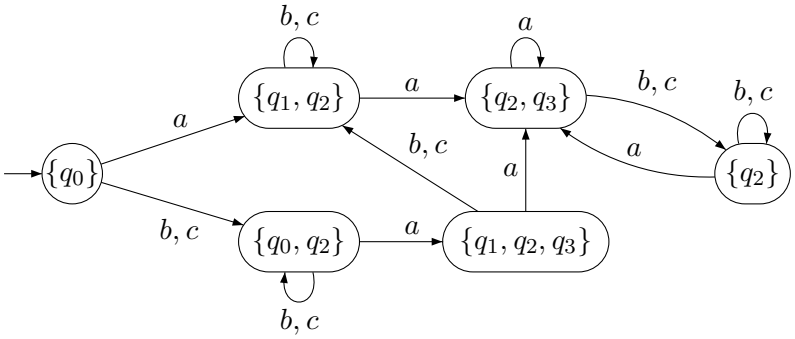
- Consideramos ahora el estado $\{q_2, q_3\}$ y tenemos que $\nu^*(\{q_2, q_3\}, a) = \nu(q_2, a) \cup \nu(q_3, a) = \{q_2, q_3\} \cup \varnothing$, es decir, $\{q_2, q_3\}$. Por otra parte, $\nu^*(\{q_2, q_3\}, b) = \nu(q_2, b) \cup \nu(q_3, b) = \{q_2\} \cup \varnothing$, es decir, $\{q_2\}$. También tenemos que, $\nu^*(\{q_2, q_3\}, c) = \nu(q_2, c) \cup \nu(q_3, c) = \{q_2\} \cup \varnothing$, es decir, $\{q_2\}$.



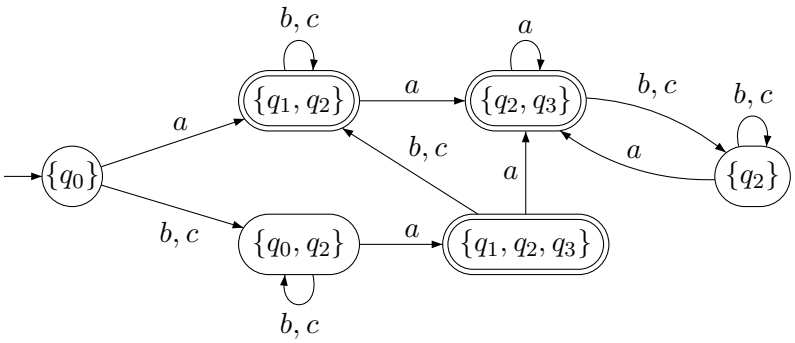
- A continuación calculamos las transiciones desde el estado $\{q_1, q_2, q_3\}$. Tenemos que $\nu^*(\{q_1, q_2, q_3\}, a) = \nu(q_1, a) \cup \nu(q_2, a) \cup \nu(q_3, a) = \varnothing \cup \{q_2, q_3\} \cup \varnothing$, es decir, $\{q_2, q_3\}$. Por otra parte, $\nu^*(\{q_1, q_2, q_3\}, b) = \nu(q_1, b) \cup \nu(q_2, b) \cup \nu(q_3, b) = \{q_1\} \cup \{q_2\} \cup \varnothing$, es decir, $\{q_1, q_2\}$. También tenemos que, $\nu^*(\{q_1, q_2, q_3\}, c) = \nu(q_1, c) \cup \nu(q_2, c) \cup \nu(q_3, c) = \{q_1\} \cup \{q_2\} \cup \varnothing$, es decir, $\{q_1, q_2\}$.



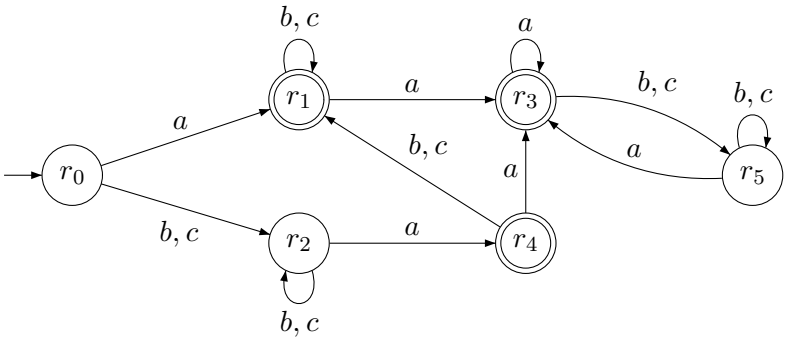
- El estado $\{q_2\}$ está sin completar. Tenemos que $\nu^*(\{q_2\}, a) = \nu(q_2, a) = \{q_2, q_3\}$. Por otra parte, $\nu^*(\{q_2\}, b) = \nu(q_2, b) = \{q_2\}$ y $\nu^*(\{q_2\}, c) = \nu(q_2, c) = \{q_2\}$.



- Como todas las transiciones están completadas, solo queda indicar qué estados tendrán doble círculo. Serán aquellos que contengan algún estado del AFND original con doble círculo. Por tanto, los estados $\{q_1, q_2\}$ por contener q_1, $\{q_2, q_3\}$ por contener q_3 y $\{q_1, q_2, q_3\}$ por contener q_1 y q_3.



- Si renombramos los estados de la siguiente forma: $r_0 = \{q_0\}$, $r_1 = \{q_1, q_2\}$, $r_2 = \{q_0, q_2\}$, $r_3 = \{q_2, q_3\}$, $r_4 = \{q_1, q_2, q_3\}$ y $r_5 = \{q_2\}$, nos queda el siguiente AFD:

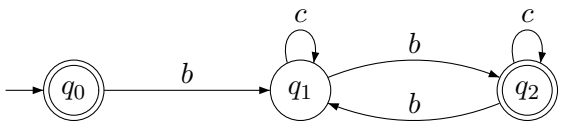


Ejemplo: de AFND a AFD, el lenguaje de las palabras que no contienen a, no empiezan por c y tienen un número par de b's

En la Figura 3.32 se muestra un AFND para el lenguaje formado por las palabras que no contienen ninguna a, no empiezan por c y tienen un número par de b's. La definición formal de dicho lenguaje sería la siguiente:

$$\{w \mid w \in A^* \wedge |w|_a = 0 \wedge |w|_b \bmod 2 = 0 \wedge \neg\exists u(u \in A^* \wedge w = cu)\}$$

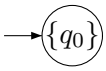
donde $A = \{a, b, c\}$.



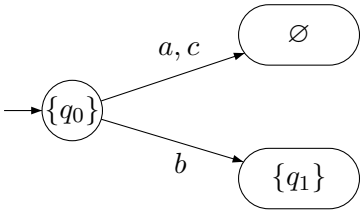
3.32 Irudia: AFND para el lenguaje de las palabras que no contienen a, no empiezan por c y tienen un número par de b's.

A continuación se calculará el AFD correspondiente al AFND de la Figura 3.32. Para ello se irán mostrando los estados que se generan paso a paso hasta obtener el AFD definitivo. Como cada estado estará etiquetado por un conjunto de estados del AFND, el proceso terminará con un renombramiento. Los pasos son los siguientes:

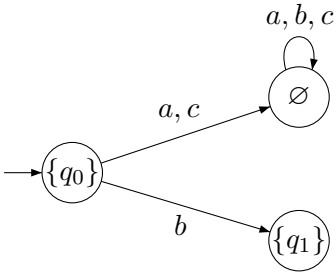
- El estado inicial es siempre $\{q_0\}$. Por tanto, primero generamos un estado con etiqueta $\{q_0\}$ y con la flecha característica de los estados iniciales.



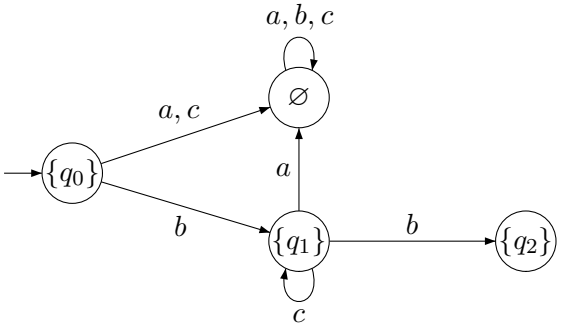
- Ahora tenemos que $\nu^*(\{q_0\}, a) = \nu(q_0, a) = \varnothing$, $\nu^*(\{q_0\}, b) = \nu(q_0, b) = \{q_1\}$ y $\nu^*(\{q_0\}, c) = \nu(q_0, c) = \varnothing$



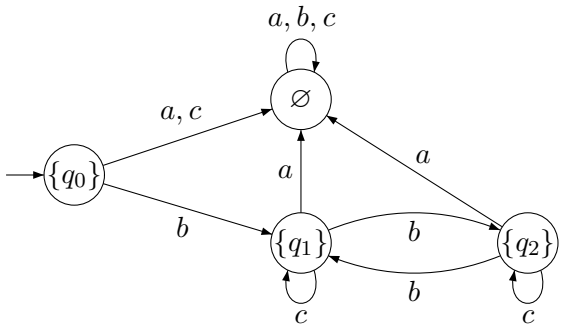
- Nos centramos primero en el estado $\varnothing$ y tenemos que $\nu^*(\varnothing, a) = \varnothing$. Por otra parte, $\nu^*(\varnothing, b) = \varnothing$. Por último, $\nu^*(\varnothing, c) = \varnothing$



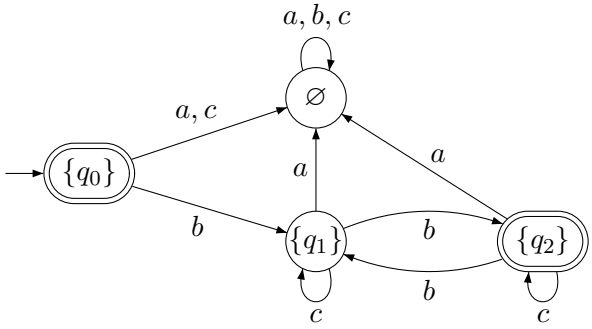
- Nos centramos ahora en el estado $\{q_1\}$ y tenemos que $\nu^*(\{q_1\}, a) = \nu(q_1, a) = \varnothing$. Por otra parte, $\nu^*(\{q_1\}, b) = \nu(q_1, b) = \{q_2\}$. También tenemos que, $\nu^*(\{q_1\}, c) = \nu(q_1, c) = \{q_1\}$.



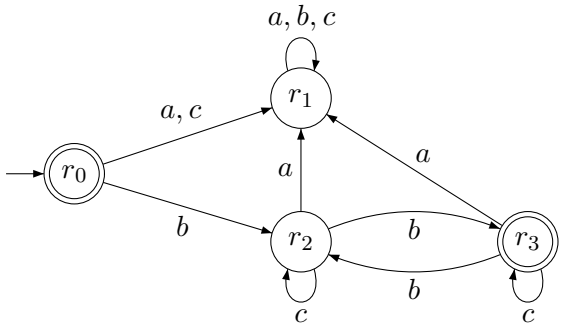
- Consideramos ahora el estado $\{q_2\}$ y tenemos que $\nu^*(\{q_2\}, a) = \nu(q_2, a) = \varnothing$. Por otra parte, $\nu^*(\{q_2\}, b) = \nu(q_2, b) = \{q_1\}$. También tenemos que, $\nu^*(\{q_2\}, c) = \nu(q_2, c) = \{q_2\}$.



- Como todas las transiciones están completadas, solo queda indicar qué estados tendrán doble círculo. Serán aquellos que contengan algún estado del AFND original con doble círculo. Por tanto, el estado $\{q_0\}$ por contener q_0 y el estado $\{q_2\}$ por contener q_2.



- Si renombramos los estados de la siguiente forma: $r_0 = \{q_0\}$, $r_1 = \varepsilon$, $r_2 = \{q_1\}$ y $r_3 = \{q_2\}$, nos queda el siguiente AFD:



3.6 Cálculo del lenguaje correspondiente a un autómata finito (AFD, AFND o ε-AFND)

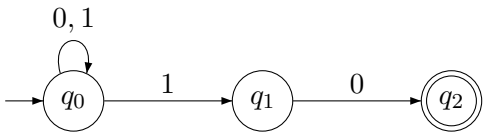
En este apartado se presentará un método que, dado un autómata finito (que abreviaremos como AF) permite obtener de manera sistemática el lenguaje definido por ese AF.

Cuando hablamos de autómatas finitos o de AF's, nos estamos refiriendo tanto a AFD's como a AFND's como a ε-AFND's. Por tanto, el método sirve para los tres tipos de autómatas finitos que hemos visto.

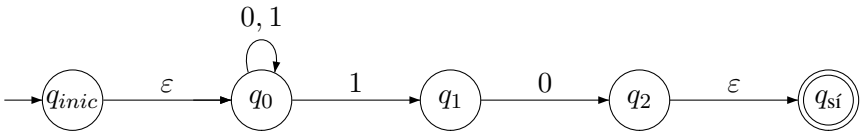
Tal como se verá al aplicar el método, la idea es trabajar con los lenguajes más elementales posibles. Si el alfabeto es $A = \{a, b, c\}$ los lenguajes básicos serán el lenguaje vacío $\varnothing$, el lenguaje que solo contiene la palabra vacía $\{\varepsilon\}$ y los lenguajes $\{a\}$, $\{b\}$ y $\{c\}$. Además para evitar el uso de las llaves, que dificulta bastante la legibilidad, prescindiremos de ellas. Por ejemplo, en vez de escribir $\{a\}$ escribiremos solo a o en vez de escribir $\{\varepsilon\}$ solo escribiremos ε. Con el objetivo de mejorar aun más la legibilidad, utilizaremos el símbolo $+$ para representar la unión de lenguajes $\cup$. A modo de ejemplo, en vez de escribir $\{a\} \cup \{b\}$ escribiremos $a+b$, en vez de escribir $\{a\}\{b\}$ escribiremos ab, en vez de escribir $\{a\}^*$ escribiremos a^* y en vez de escribir $(\{a\}\{a\}(\{b\}^*)) \cup ((\{b\}\{b\})^*)$ escribiremos $(aa(b^*)) + ((bb)^*)$. Además, podemos evitar el uso de paréntesis en muchos casos teniendo en cuenta que $*$ es el que mayor prioridad tiene, luego viene la concatenación y por último $+$, es decir, la unión. Por tanto, en vez de escribir $(aa(b^*)) + ((bb)^*)$ escribiremos $aab^* + (bb)^*$.

3.6.1 Método de eliminación de estados y cálculo del lenguaje correspondiente

Utilizaremos un ejemplo para ir mostrando los pasos que se han de aplicar el método. Supongamos que tenemos el siguiente AF:

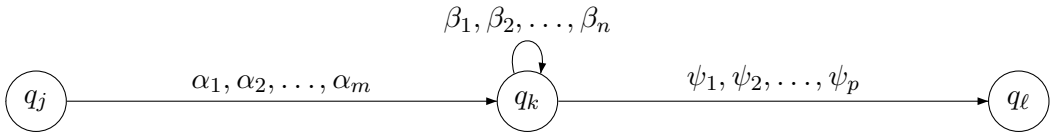


El primer paso consiste en añadir dos estados nuevos q_{inic} y $q_{\text{sí}}$. El estado q_{inic} será el nuevo estado inicial y el el estado $q_{\text{sí}}$ será el único que lleve doble círculo. Por tanto habrá que añadir una transición vacía desde q_{inic} a q_0 y también una transición vacía desde cada estado del AF inicial con doble círculo a $q_{\text{sí}}$.

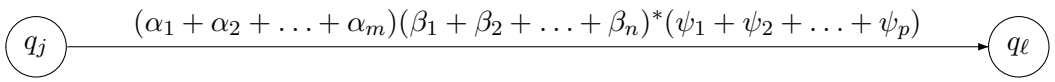


A continuación hay que ir eliminando los estados q_0, q_1 y q_2 de uno en uno. En principio da igual el orden elegido para eliminar dichos estados pero a veces algunos estados son más fáciles de eliminar y conviene ir eliminando primero los más fáciles. La idea fundamental es que al eliminar el estado q_k, hay que mantener todos los caminos que pasan por q_k.

- Consideremos un caso general:

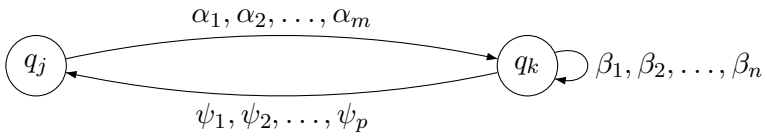


Para eliminar el estado q_k, hay que mantener los caminos que pasan por q_k. Nos quedaría lo siguiente:

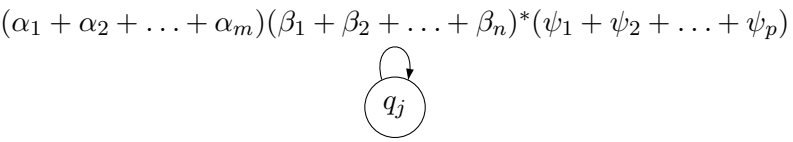


Habría que aplicar ese esquema a cada camino que pasa por q_k y conecta a otros dos estados q_j y q_ℓ.

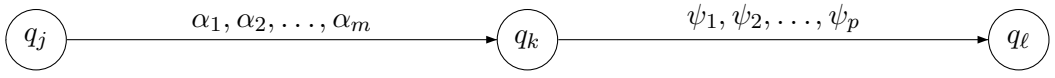
- Si q_j y q_ℓ son el mismo estado, es decir, si la situación es la siguiente:



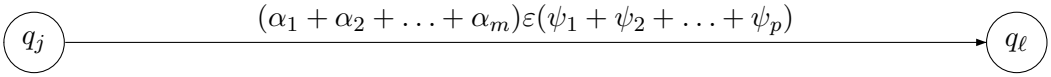
Al eliminar q_k se obtendría lo siguiente:



- Si no hay β's, es decir si $n = 0$,

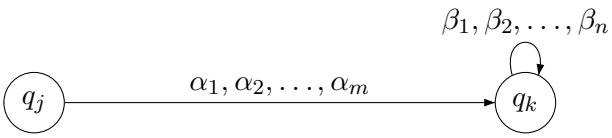


Entonces como $\beta_1 + \beta_2 + \ldots + \beta_n$ es una suma vacía o una unión vacía, es igual a $\varnothing$ y nos queda $\varnothing^*$ que es $\{\varepsilon\}$, es decir, ε:



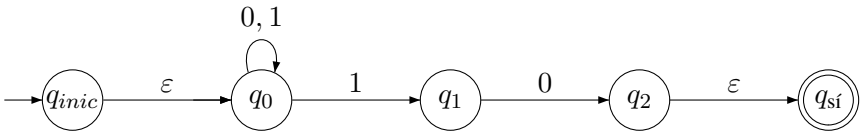
Pero $(\alpha_1 + \alpha_2 + \ldots + \alpha_m)\varepsilon(\psi_1 + \psi_2 + \ldots + \psi_p)$ es lo mismo que $(\alpha_1 + \alpha_2 + \ldots + \alpha_m)(\psi_1 + \psi_2 + \ldots + \psi_p)$. Es decir, se puede quitar ε.

- Por último, si no hay ψ's, es decir si $p = 0$, es decir, de q_k no se puede ir a ningún otro estado

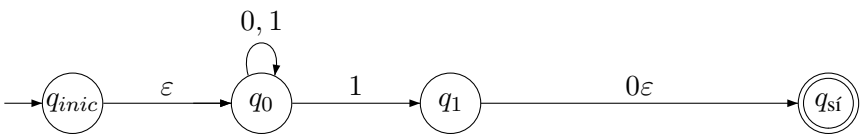


Entonces como $\psi_1 + \psi_2 + \ldots + \psi_p$ es una suma vacía o una unión vacía, es igual a $\varnothing$ y nos queda $(\alpha_1 + \alpha_2 + \ldots + \alpha_m)(\beta_1 + \beta_2 + \ldots + \beta_p)^*\varnothing$ que es igual a $\varnothing$. Esto supone que se elimina el estado q_k sin más:

Retomamos ahora el ejemplo que habíamos iniciado:

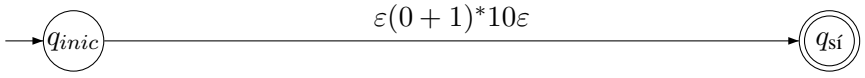


Primero procedemos a eliminar q_2. En este caso tenemos solo un α y un ψ ($\alpha_1 = 0$, $\psi_1 = \varepsilon$) y no hay β's. Nos queda lo siguiente:



A continuación procedemos a eliminar q_1. En este caso tenemos también solo un α y un ψ ($\alpha_1 = 1$, $\psi_1 = 0\varepsilon$) y no hay β's. Nos queda lo siguiente:

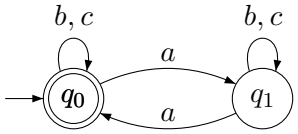
Por último procedemos a eliminar q_0. En este caso tenemos también solo un α y un ψ ($\alpha_1 = \varepsilon$, $\psi_1 = 10\varepsilon$) y además hay dos β's ($\beta_1 = 0$, $\beta_2 = 1$). Nos queda lo siguiente:



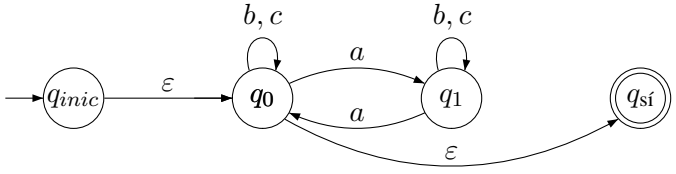
Por tanto el lenguaje definido por este autómata es $\varepsilon(0 + 1)^*10\varepsilon$. si lo simplificamos eliminando los ε concatenados, nos queda $(0 + 1)^*10$. Recordemos que esto representa el lenguaje $(\{0\} \cup \{1\})^*\{1\}\{0\}$, que viene a ser el lenguaje de las palabras que terminan con la cadena 10.

3.6.2 Ejemplo: AF correspondiente al lenguaje de las palabras que tienen un número par de a's

Se quiere calcular el lenguaje definido por el siguiente AF:

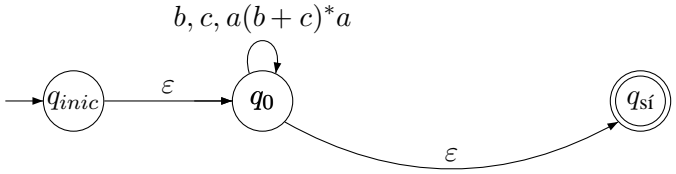


El primer paso consiste en añadir dos estados nuevos q_{inic} y $q_{sí}$. El estado q_{inic} será el nuevo estado inicial y el el estado $q_{sí}$ será el único que lleve doble círculo. Por tanto habrá que añadir una transición vacía desde q_{inic} a q_0 y también una transición vacía desde cada estado del AF inicial con doble círculo a $q_{sí}$.

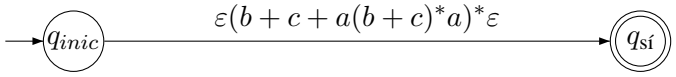


A continuación hay que ir eliminando los estados q_0 y q_1 de uno en uno. En principio da igual el orden elegido para eliminar dichos estados pero a veces algunos estados son más fáciles de eliminar y conviene ir eliminando primero los más fáciles.

Primero procedemos a eliminar q_1. En este caso tenemos solo un α y un ψ ($\alpha_1 = a$, $\psi_1 = a$) y hay, además, dos β's ($\beta_1 = b$, $\beta_2 = c$). Nos queda lo siguiente:



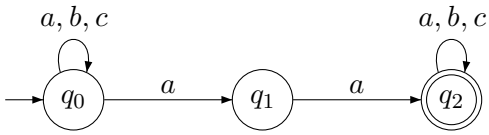
A continuación procedemos a eliminar q_0. En este caso tenemos solo un α y un ψ ($\alpha_1 = \varepsilon$, $\psi_1 = \varepsilon$) y hay, además, tres β's ($\beta_1 = b$, $\beta_2 = c$, $\beta_3 = a(b + c)^*a$). Nos queda lo siguiente:



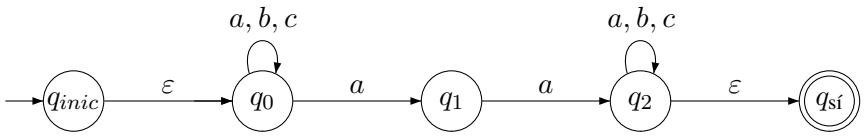
La expresión $\varepsilon(b+c+a(b+c)^*a)^*\varepsilon$ se puede simplificar eliminando ε. Es decir, $(b+c+a(b+c)*a)^*$ es la definición correspondiente al lenguaje en este caso.

3.6.3 Ejemplo: AF correspondiente al lenguaje de las palabras que contienen aa

Supongamos que tenemos el siguiente AF:

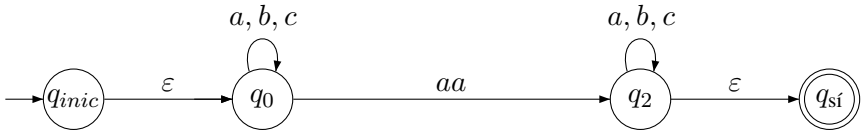


El primer paso consiste en añadir dos estados nuevos q_{inic} y $q_{\text{sí}}$. El estado q_{inic} será el nuevo estado inicial y el estado $q_{\text{sí}}$ será el único que lleve doble círculo. Por tanto habrá que añadir una transición vacía desde q_{inic} a q_0 y también una transición vacía desde cada estado del AF inicial con doble círculo a $q_{\text{sí}}$.

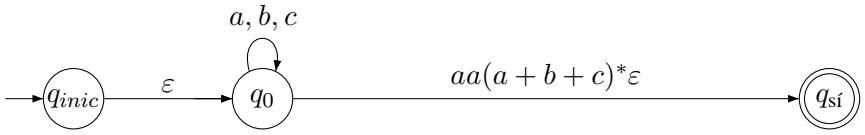


A continuación hay que ir eliminando los estados q_0, q_1 y q_2 de uno en uno. En principio da igual el orden elegido para eliminar dichos estados pero a veces algunos estados son más fáciles de eliminar y conviene ir eliminando primero los más fáciles. La idea fundamental es que al eliminar un estado, hay que mantener todos los caminos que pasan por ese estado.

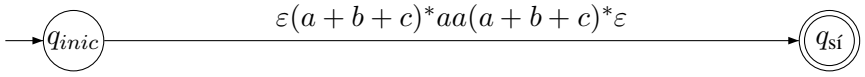
Primero procedemos a eliminar q_1. En este caso tenemos solo un α y un ψ ($\alpha_1 = a$, $\psi_1 = a$) y no hay β's. Nos queda lo siguiente:



A continuación procedemos a eliminar q_2. En este caso tenemos también solo un α y un ψ ($\alpha_1 = aa$, $\psi_1 = \varepsilon$) y además hay tres β's ($\beta_1 = a$, $\beta_2 = b$, $\beta_3 = c$). Nos queda lo siguiente:



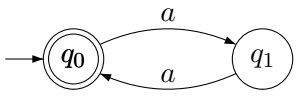
Por último procedemos a eliminar q_0. En este caso tenemos también solo un α y un ψ ($\alpha_1 = \varepsilon$, $\psi_1 = aa(a+b+c)^*\varepsilon$) y además hay tres β's ($\beta_1 = a$, $\beta_2 = b$, $\beta_3 = c$). Nos queda lo siguiente:



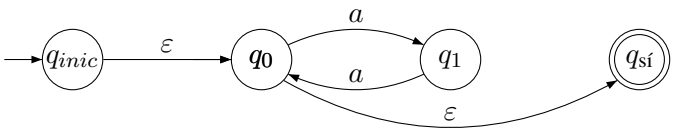
Por tanto el lenguaje definido por este autómata es $\varepsilon(a+b+c)^*aa(a+b+c)^*\varepsilon$. si lo simplificamos eliminando los ε concatenados, nos queda $(a+b+c)^*aa(a+b+c)^*$. Recordemos que esto representa el lenguaje $(\{a\} \cup \{b\} \cup \{c\})^*\{a\}\{a\}(\{a\} \cup \{b\} \cup \{c\})^*$, que viene a ser el lenguaje de las palabras que contienen la cadena aa.

3.6.4 Ejemplo: AF correspondiente al lenguaje de las palabras que solo tienen un número par de a's (no hay ni b's ni c's)

Se quiere calcular el lenguaje definido por el siguiente AF:

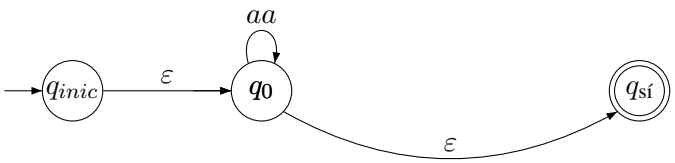


El primer paso consiste en añadir dos estados nuevos q_{inic} y $q_{sí}$. El estado q_{inic} será el nuevo estado inicial y el el estado $q_{sí}$ será el único que lleve doble círculo. Por tanto habrá que añadir una transición vacía desde q_{inic} a q_0 y también una transición vacía desde cada estado del AF inicial con doble círculo a $q_{sí}$.

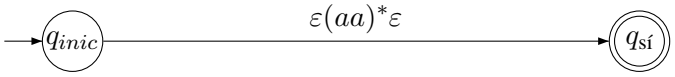


A continuación hay que ir eliminando los estados q_0 y q_1 de uno en uno. En principio da igual el orden elegido para eliminar dichos estados pero a veces algunos estados son más fáciles de eliminar y conviene ir eliminando primero los más fáciles.

Primero procedemos a eliminar q_1. En este caso tenemos solo un α y un ψ ($\alpha_1 = a$, $\psi_1 = a$) y no hay β's. Nos queda lo siguiente:



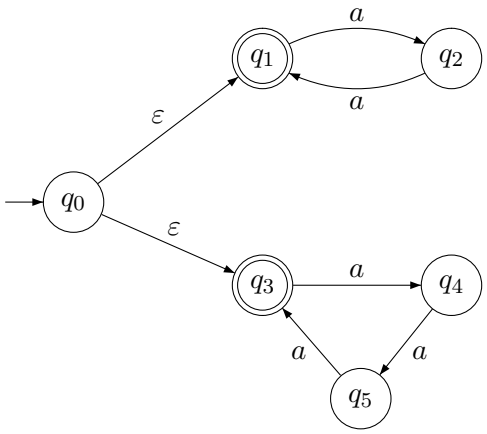
A continuación procedemos a eliminar q_0. En este caso tenemos solo un α y un ψ ($\alpha_1 = \varepsilon$, $\psi_1 = \varepsilon$) y un β ($\beta_1 = aa$). Nos queda lo siguiente:



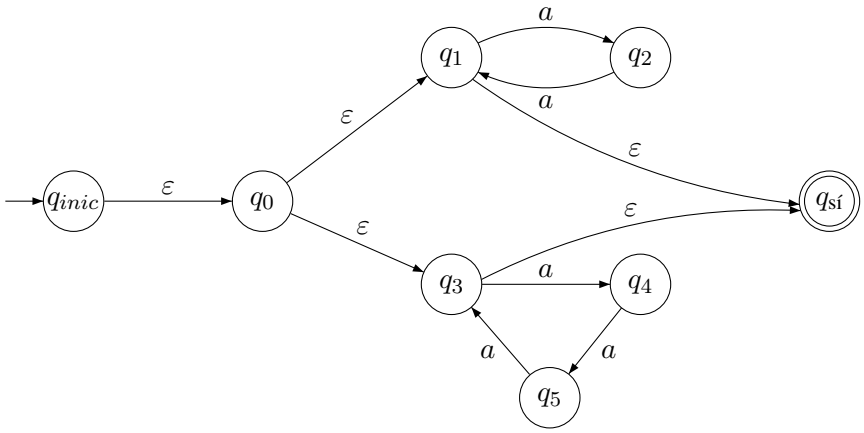
La expresión $\varepsilon(aa)^*\varepsilon$ se puede simplificar eliminando ε. Es decir, $(aa)^*$ es la definición correspondiente al lenguaje en este caso. Recordemos que $(aa)^*$ representa $(\{a\}\{a\})^*$.

3.6.5 Ejemplo: AF correspondiente al lenguaje de las palabras que están formadas por repeticiones de aa o repeticiones de aaa

Supongamos que tenemos el siguiente AF:

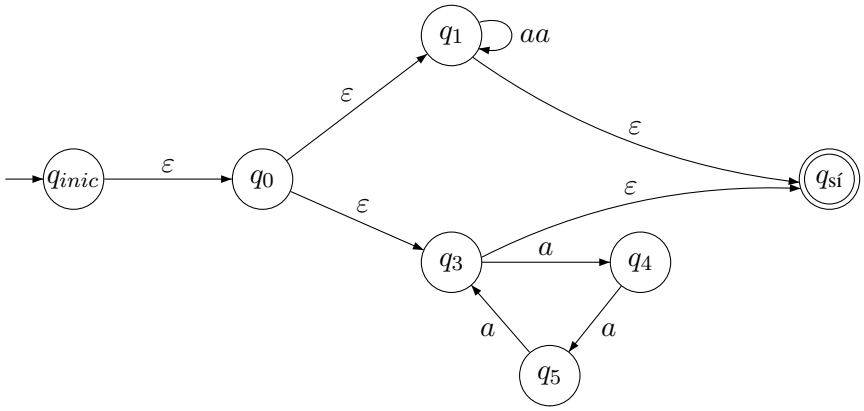


El primer paso consiste en añadir dos estados nuevos q_{inic} y $q_{\text{sí}}$. El estado q_{inic} será el nuevo estado inicial y el el estado $q_{\text{sí}}$ será el único que lleve doble círculo. Por tanto habrá que añadir una transición vacía desde q_{inic} a q_0 y también una transición vacía desde cada estado del AF inicial con doble círculo a $q_{\text{sí}}$.

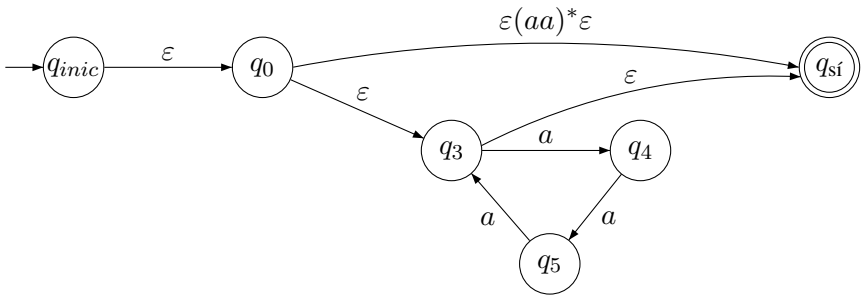


A continuación hay que ir eliminando los estados q_0, q_1 ,. . ., q_5 de uno en uno. En principio da igual el orden elegido para eliminar dichos estados pero a veces algunos estados son más fáciles de eliminar y conviene ir eliminando primero los más fáciles. La idea fundamental es que al eliminar un estado, hay que mantener todos los caminos que pasan por ese estado.

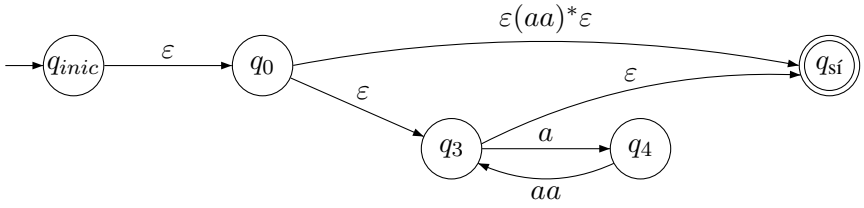
Primero procedemos a eliminar q_2. En este caso tenemos solo un α y un ψ ($\alpha_1 = a$, $\psi_1 = a$) y no hay β's. Nos queda lo siguiente:



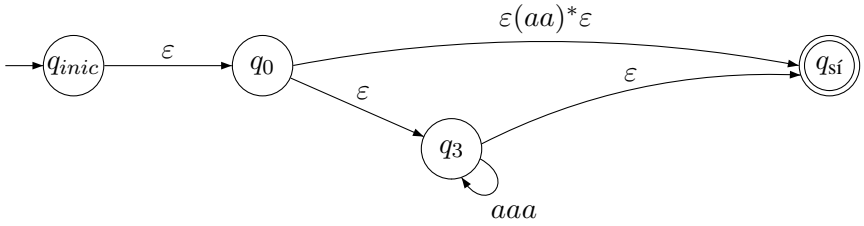
A continuación procedemos a eliminar q_1. En este caso tenemos también solo un α y un ψ ($\alpha_1 = \varepsilon$, $\psi_1 = \varepsilon$) y además hay un β ($\beta_1 = aa$). Nos queda lo siguiente:



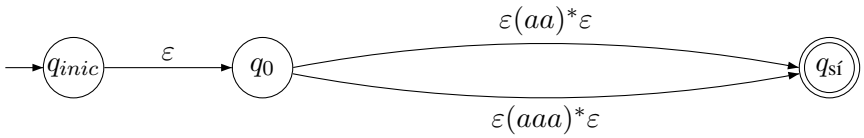
A continuación procedemos a eliminar q_5. En este caso tenemos también solo un α y un ψ ($\alpha_1 = a$, $\psi_1 = a$) y no hay β's. Nos queda lo siguiente:



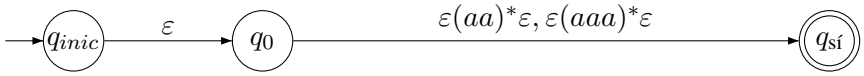
A continuación procedemos a eliminar q_4. En este caso tenemos también solo un α y un ψ ($\alpha_1 = a$, $\psi_1 = aa$) y no hay β's. Nos queda lo siguiente:



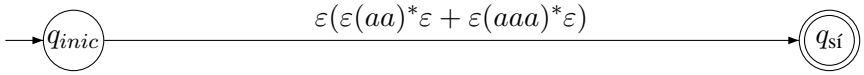
A continuación procedemos a eliminar q_3. En este caso tenemos también solo un α y un ψ ($\alpha_1 = \varepsilon$, $\psi_1 = \varepsilon$) y una β ($\beta_1 = aaa$). Nos queda lo siguiente:



En vez de dos flechas que van de q_0 a $q_{\text{sí}}$, escribiríamos solo una con dos elementos separados por una coma:



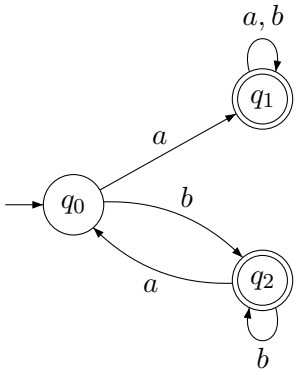
Por último procedemos a eliminar q_0. En este caso tenemos una α y dos ψ's ($\alpha_1 = \varepsilon$, $\psi_1 = \varepsilon(aa)^*\varepsilon$, $\psi_2 = \varepsilon(aaa)^*\varepsilon$) y no hay β's. Nos queda lo siguiente:



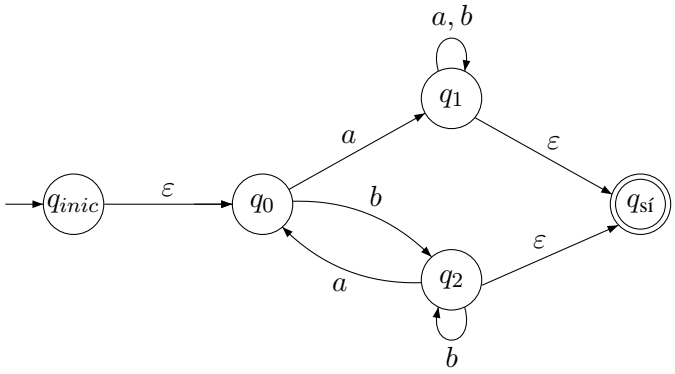
Por tanto el lenguaje definido por este autómata es $\varepsilon(\varepsilon(aa)^*\varepsilon + \varepsilon(aaa)^*\varepsilon)$. Si lo simplificamos eliminando los ε concatenados, nos queda $(aa)^* + (aaa)^*$. Recordemos que esto representa el lenguaje $(\{a\}\{a\})^* \cup (\{a\}\{a\}\{a\})^*$, que viene a ser el lenguaje de las palabras formadas exclusivamente por repeticiones de aa o formadas exclusivamente por repeticiones de aaa.

3.6.6 Ejemplo: Otro ejemplo de cálculo del lenguaje correspondiente a un AF

Supongamos que tenemos el siguiente AF:

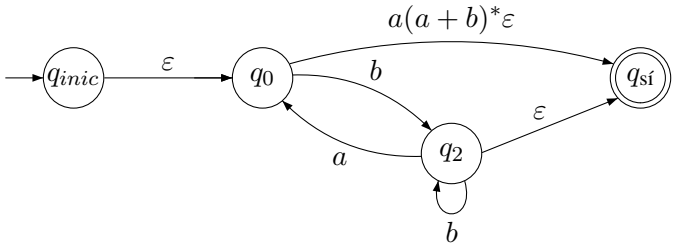


El primer paso consiste en añadir dos estados nuevos q_{inic} y $q_{sí}$. El estado q_{inic} será el nuevo estado inicial y el estado $q_{sí}$ será el único que lleve doble círculo. Por tanto habrá que añadir una transición vacía desde q_{inic} a q_0 y también una transición vacía desde cada estado del AF inicial con doble círculo a $q_{sí}$.

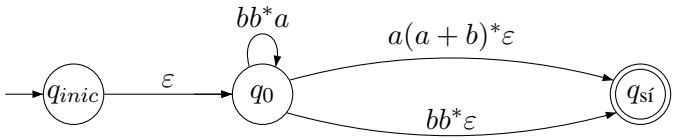


A continuación hay que ir eliminando los estados q_0, q_1 y q_2 de uno en uno. En principio da igual el orden elegido para eliminar dichos estados pero a veces algunos estados son más fáciles de eliminar y conviene ir eliminando primero los más fáciles. La idea fundamental es que al eliminar un estado, hay que mantener todos los caminos que pasan por ese estado.

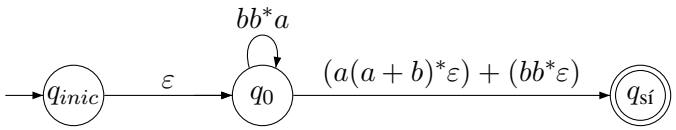
Primero procedemos a eliminar q_1. En este caso tenemos solo un α y un ψ ($\alpha_1 = a$, $\psi_1 = \varepsilon$) y dos β's ($\beta_1 = a$, $\beta_2 = b$). Nos queda lo siguiente:



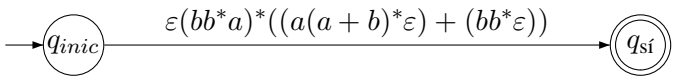
A continuación procedemos a eliminar q_2. En este caso tenemos dos caminos distintos que pasan por q_2: el camino que va de q_0 a q_0 pasando por q_2 y el camino que va de q_0 a $q_{\text{sí}}$ pasando por q_2. Hay que analizar cada camino por separado. En el primer camino tenemos un α, un β y un ψ ($\alpha_1 = b$, $\beta_1 = b$ y $\psi_1 = a$). En el segundo camino tenemos también un α, un β y un ψ ($\alpha_1 = b$, $\beta_1 = b$ y $\psi_1 = \varepsilon$).Nos queda lo siguiente:



Como tenemos dos arcos que van de q_0 a $q_{\text{sí}}$, primero ponemos un único arco con ambas expresiones separadas por una coma:



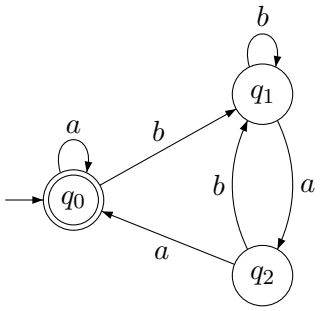
Por último procedemos a eliminar q_0. En este caso tenemos un único camino que pasa por q_0. En ese camino tenemos una α, una β y una ψ's ($\alpha_1 = \varepsilon$, $\beta_1 = bb^*a$ y $\psi_1 = (a(a+b)^*\varepsilon) + (bb^*\varepsilon)$). Nos queda lo siguiente:



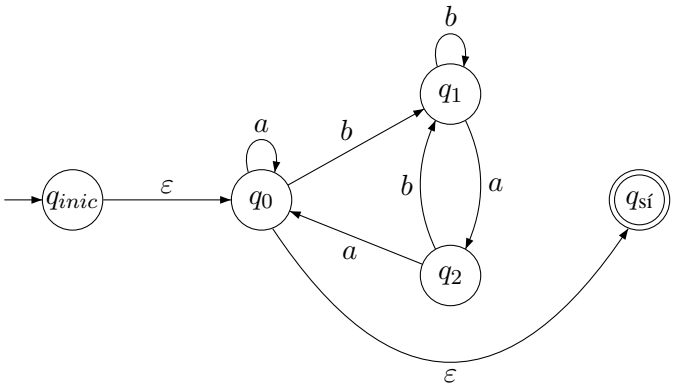
Por tanto el lenguaje definido por este autómata es $\varepsilon(bb^*a)^*((a(a+b)^*\varepsilon)+(bb^*\varepsilon))$. Si lo simplificamos eliminando los ε concatenados, nos queda $(bb^*a)^*((a(a+b)^*)+(bb^*))$. Recordemos que esto representa el lenguaje $(\{b\}\{b\}^*\{a\})^*((\{a\}(\{a\}+\{b\})^*)+(\{b\}\{b\}^*))$.

3.6.7 Ejemplo: Otro ejemplo más de cálculo del lenguaje correspondiente a un AF

Supongamos que tenemos el siguiente AF:

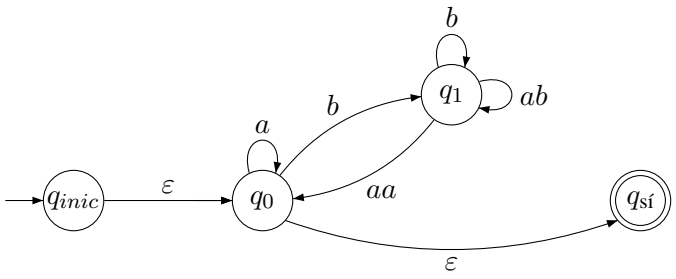


El primer paso consiste en añadir dos estados nuevos q_{inic} y $q_{\text{sí}}$. El estado q_{inic} será el nuevo estado inicial y el estado $q_{\text{sí}}$ será el único que lleve doble círculo. Por tanto habrá que añadir una transición vacía desde q_{inic} a q_0 y también una transición vacía desde cada estado del AF inicial con doble círculo a $q_{\text{sí}}$.

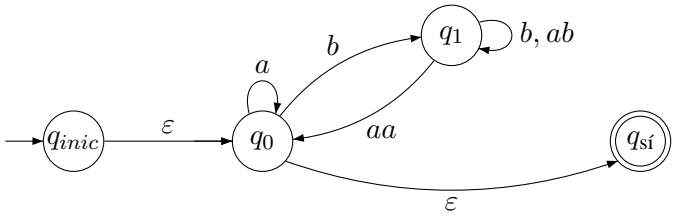


A continuación hay que ir eliminando los estados q_0, q_1 y q_2 de uno en uno. En principio da igual el orden elegido para eliminar dichos estados pero a veces algunos estados son más fáciles de eliminar y conviene ir eliminando primero los más fáciles. La idea fundamental es que al eliminar un estado, hay que mantener todos los caminos que pasan por ese estado.

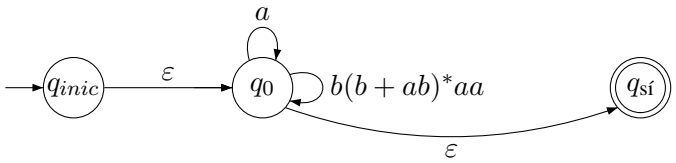
Primero procedemos a eliminar q_2. En este caso tenemos dos caminos distintos que pasan por q_2: el camino que va de q_1 a q_0 pasando por q_2 y el camino que va de q_1 a q_1 pasando por q_2. Hay que analizar cada camino por separado. En el primer camino tenemos un α, ningún β y un ψ ($\alpha_1 = a$ y $\psi_1 = a$). En el segundo camino tenemos también un α y un ψ ($\alpha_1 = a$ y $\psi_1 = \varepsilon$). Nos queda lo siguiente:



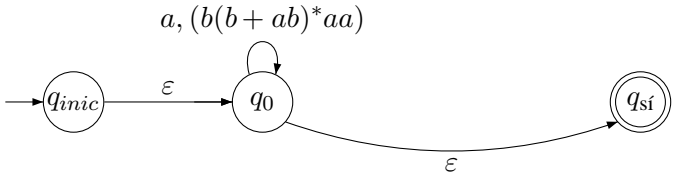
Como q_1 tiene dos bucles, vamos a representarlo en el formato habitual, que es con un único bucle y separando las dos expresiones con coma:



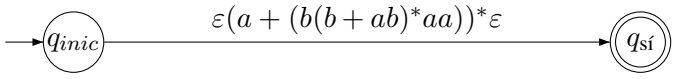
A continuación procedemos a eliminar q_1. En este caso tenemos un único camino que va de q_0 a q_0 pasando por q_1. En ese camino tenemos un α, dos β's y un ψ ($\alpha_1 = b$, $\beta_1 = b$, $\beta_2 = ab$ y $\psi_1 = aa$). Nos queda lo siguiente:



Como tenemos dos arcos que van de q_0 a q_0, primero ponemos un único arco con ambas expresiones separadas por una coma:



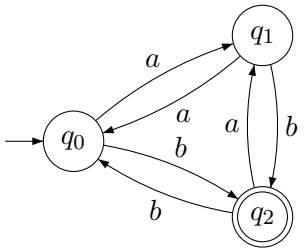
Por último procedemos a eliminar q_0. En este caso tenemos un único camino que pasa por q_0. En ese camino tenemos una α, dos β y una ψ ($\alpha_1 = \varepsilon$, $\beta_1 = bb^*a$ y $\psi_1 = (a(a+b)^*\varepsilon) + (bb^*\varepsilon)$). Nos queda lo siguiente:



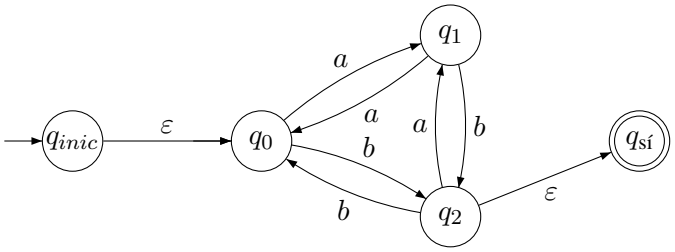
Por tanto el lenguaje definido por este autómata es $\varepsilon(a + (b(b+ab)^*aa))^*\varepsilon$. Si lo simplificamos eliminando los ε concatenados, nos queda $(a + (b(b+ab)^*aa))^*$. Recordemos que esto representa el lenguaje $(\{a\} \cup (\{b\}(\{b\} \cup \{a\}\{b\})^*\{a\}\{a\}))^*$.

3.6.8 Ejemplo: Último ejemplo de cálculo del lenguaje correspondiente a un AF

Supongamos que tenemos el siguiente AF:

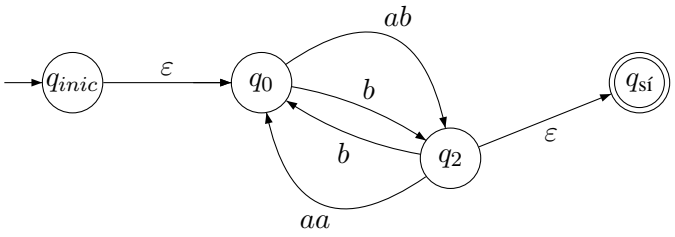


El primer paso consiste en añadir dos estados nuevos q_{inic} y $q_{sí}$. El estado q_{inic} será el nuevo estado inicial y el estado $q_{sí}$ será el único que lleve doble círculo. Por tanto habrá que añadir una transición vacía desde q_{inic} a q_0 y también una transición vacía desde cada estado del AF inicial con doble círculo a $q_{sí}$.

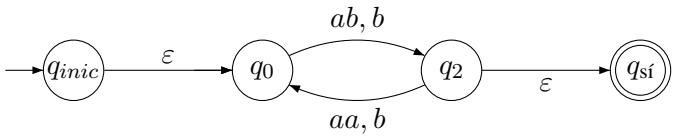


A continuación hay que ir eliminando los estados q_0, q_1 y q_2 de uno en uno. En principio da igual el orden elegido para eliminar dichos estados pero a veces algunos estados son más fáciles de eliminar y conviene ir eliminando primero los más fáciles. La idea fundamental es que al eliminar un estado, hay que mantener todos los caminos que pasan por ese estado.

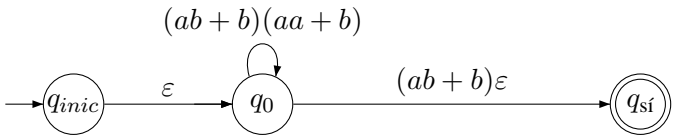
Primero procedemos a eliminar q_1. En este caso tenemos dos caminos distintos que pasan por q_1: el camino que va de q_0 a q_2 pasando por q_1 y el camino que va de q_2 a q_0 pasando por q_1. Hay que analizar cada camino por separado. En el primer camino tenemos un α, ningún β y un ψ ($\alpha_1 = a$ y $\psi_1 = b$). En el segundo camino tenemos también un α y un ψ ($\alpha_1 = a$ y $\psi_1 = a$). Nos queda lo siguiente:



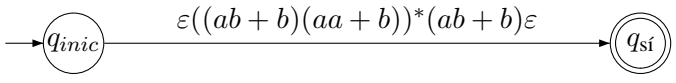
Como entre q_0 y q_2 hay dos arcos en ambas direcciones, vamos a representarlo en el formato habitual, que es con un único arco en cada dirección y separando las dos expresiones de cada arco con coma:



A continuación procedemos a eliminar q_2. En este caso tenemos también dos caminos distintos que pasan por q_2: el camino que va de q_0 a q_0 pasando por q_2 y el camino que va de q_0 a $q_{\text{sí}}$ pasando por q_2. Hay que analizar cada camino por separado. En el primer camino tenemos dos α's, ningún β y dos ψ's ($\alpha_1 = ab$, $\alpha_2 = b$, $\psi_1 = aa$ y $\psi_2 = b$). En el segundo camino tenemos dos α's, ningún β y un ψ ($\alpha_1 = ab$, $\alpha_2 = b$ y $\psi_1 = \varepsilon$). Nos queda lo siguiente:



Por último procedemos a eliminar q_0. En este caso tenemos un único camino que pasa por q_0. En ese camino tenemos una α, una β y una ψ ($\alpha_1 = \varepsilon$, $\beta_1 = (ab + b)(aa + b)$ y $\psi_1 = (ab + b)\varepsilon$. Nos queda lo siguiente:



Por tanto el lenguaje definido por este autómata es $\varepsilon((ab{+}b)(aa{+}b))^*(ab{+}b)\varepsilon$. Si lo simplificamos eliminando los ε concatenados, nos queda $((ab{+}b)(aa{+}b))^*(ab{+}b)$. Recordemos que esto representa el lenguaje $((\{a\}\{b\} \cup \{b\})(\{a\}\{a\} \cup \{b\}))^*(\{a\}\{b\} \cup \{b\})$.

3.7 Lenguajes regulares

En la sección anterior se ha visto que al obtener la expresión correspondiente al lenguaje definido por un autómata finito, solo se utilizan las siguientes tres operaciones sobre lenguajes: la unión (representada mediante $\cup$ o $+$), la clausura (representada mediante $*$) y la concatenación (que no tiene ningún símbolo). Esta observación conduce a considerar aquellos lenguajes que son definibles utilizando únicamente esas tres operaciones. Dichos lenguajes son conocidos como lenguajes regulares.

3.7.1 Definición de lenguajes regulares

Dado un alfabeto A, la noción de lenguaje regular sobre el alfabeto A se define recursivamente de la siguiente forma:

- $\varnothing$ y $\{\varepsilon\}$ son lenguajes regulares definidos sobre A.
- Para cada símbolo α que pertenece al alfabeto A, el lenguaje $\{\alpha\}$ es un lenguaje regular definido sobre A.
- Si L es un lenguaje regular definido sobre A, entonces el lenguaje L^* (clausura de L), es un lenguaje regular definido sobre A.
- Si L_1 y L_2 son lenguajes regulares definidos sobre A, entonces el lenguaje L_1L_2 (concatenación) y el lenguaje $L_1 \cup L_2$ (unión) son lenguajes regulares definidos sobre A.

Por tanto, por un lado $\varnothing$, $\{\varepsilon\}$ y $\{\alpha\}$ (donde α es un símbolo de A) son lenguajes regulares. Por otro lado los lenguajes que se obtienen como clausura de otros lenguajes regulares son tambíen lenguajes regulares. Por último, los lenguajes construibles a partir de otros lenguajes regulares utilizando las operaciones de concatenación y unión son también lenguajes regulares. Conviene recordar que la clausura L^* de un lenguaje L se define de la siguiente forma: $L^* = L^0 \cup L^1 \cup L^2 \cup L^3 \ldots$

Es decir, L^* se obtiene concatenando las palabras de L todas las veces posibles. Por ejemplo, para $L = \{ab, cc\}$, L^* contendría

- las palabras obtenidas mediante cero concatenaciones (L^0): ε
- las palabras obtenidas mediante una concatenación (L^1): ab, cc
- las palabras obtenidas mediante dos concatenaciones (L^2) $abab$, $abcc$, $ccab$, $cccc$
- las palabras obtenidas mediante tres concatenaciones (L^3) $ababab$, $ababcc$, $abccab$, $abcccc$, $ccabab$, $ccabcc$, $ccccab$, $cccccc$
- Etc.

Tal como se ve en las palabras dadas a modo de ejemplo, el orden de los símbolos de las palabras de L no se puede cambiar y por tanto, palabras como ba, $baba$ y $bacc$ no pertenecen al lenguaje L^* porque ba no pertenece al lenguaje L.

3.7.2 Ejemplos de lenguajes regulares

Consideramos el alfabeto $A = \{a, b, c\}$.

1. El lenguaje L_1 formado por todas las palabras que no contienen ni b-s ni c-s es un lenguaje regular. Para probar esto tenemos que dar una manera de definir L_1 utilizando los lenguajes regulares básicos $\varnothing$, $\{\varepsilon\}$, $\{a\}$, $\{b\}$ y $\{c\}$ y las operaciones de concatenación, unión y clausura para lenguajes. El lenguaje L_1 se define a partir del lenguaje regular $\{a\}$ y utilizando la clausura: $L_1 = \{a\}^*$. Otra manera de expresar lo mismo sería definiendo primero el lenguaje $H = \{a\}$. El lenguaje H es regular por definición. Utilizando el lenguaje regular H definiríamos L_1 de la siguiente forma: $L_1 = H^*$. Otra manera de definir L_1 de manera abreviada sería $L_1 = a^*$.

2. El lenguaje L_2 formado por todas las palabras no vacías que sólo contienen a-s es un lenguaje regular. L_2 puede definirse como $L_1 \setminus \{\varepsilon\}$. Pero esta definición no nos vale para probar que L_2 es regular, ya que la operación de diferencia $\setminus$ no está permitida a la hora de probar que un lenguaje es regular. Por otro lado L_2 no puede ser definido como $\{a\}^*$ porque en el lenguaje $\{a\}^*$ se incluye también la palabra ε pero ε no pertenece a L_2. Como las palabras de L_2 han de tener al menos una a, L_2 se define como la concatenación del lenguaje regular $\{a\}$ y el lenguaje regular $\{a\}^*$. Es decir, $L_2 = \{a\}\{a\}^*$, o utilizando los lenguajes regulares del punto 1, $L_2 = HL_1$. Otra manera de definir L_2 de manera abreviada sería $L_1 = aa^*$.

3. El lenguaje $L_3 = \{a, b, c\}$, que contiene los tres símbolos del alfabeto A, es regular, ya que L_3 puede ser definido como la unión de los lenguajes regulares $\{a\}$, $\{b\}$ y $\{c\}$. Es decir, $L_3 = \{a\} \cup \{b\} \cup \{c\}$. Otra manera de expresar lo mismo sería definiendo los lenguajes $H_1 = \{a\}$, $H_2 = \{b\}$ y $H_3 = \{c\}$, que son regulares por definición, y definiendo L_3 de la siguiente forma: $L_3 = H_1 \cup H_2 \cup H_3$. Otra manera de definir L_3 de manera abreviada sería $L_3 = a + b + c$.

4. El lenguaje universal A^*, formado por todas las palabras que se pueden generar a partir del alfabeto A, es regular, ya que A^* puede ser definido como la clausura del lenguaje regular L_3. Es decir, $A^* = L_3^*$. Otra manera de expresar lo mismo sería $A^* = (\{a\} \cup \{b\} \cup \{c\})^*$. Una manera abreviada sería la siguiente: $A^* = (a + b + c)^*$.

5. El lenguaje L_4 formado por todas las palabras que no contienen ninguna b ni ninguna c y en las que el número de a-s es par (ε, aa, $aaaa$, $aaaaaa$, etc.) es regular. Definimos primero el lenguaje $L_5 = \{aa\}$. El lenguaje L_5 es regular porque se puede definir como la concatenación del lenguaje regular $\{a\}$ con el lenguaje regular $\{a\}$. Es decir, $L_5 = \{a\}\{a\}$. O dicho de otra forma, $L_5 = HH$ donde el H es el lenguaje regular $\{a\}$. El lenguaje L_4 se define como la clausura del lenguaje L_5. Por tanto, $L_4 = (L_5)^*$. Otra manera de definir L_4 sería $L_4 = (\{a\}\{a\})^*$. Otra manera de definir L_4 de manera abreviada sería $L_4 = (aa)^*$.

6. A continuación vamos a probar que el lenguaje L_6 formado por todas las palabras que empiezan por a es regular. Una manera de definir formalmente este lenguaje es la siguiente:

$$L_6 = \{w | w \in A^* \wedge \exists u (u \in A^* \wedge w = au)\}$$

   Pero esa definición ni sirve para probar que L_6 es regular. Para probar que es regular tenemos que dar una manera de definir L_6 utilizando los lenguajes regulares básicos $\varnothing$, $\{\varepsilon\}$, $\{a\}$, $\{b\}$ y $\{c\}$ y las operaciones de concatenación, unión y clausura para lenguajes. Una manera de definir el lenguaje L_6 utilizando esos elementos sería la siguiente: $L_6 = \{a\}(\{a\} \cup \{b\} \cup \{c\})^*$. Otra manera de expresar lo mismo sería utilizando los lenguajes regulares $H = \{a\}$ y $L_3 = \{a, b, c\}$: $L_6 = H(L_3)^*$. Otra posibilidad para definir L_6 de manera abreviada sería $L_6 = a(a + b + c)^*$.

7. A continuación vamos a probar que el lenguaje L_7 formado por todas las palabras que terminan en b es regular. Una manera de definir formalmente este lenguaje es la siguiente:

$$L_7 = \{w | w \in A^* \wedge \exists u (u \in A^* \wedge w = ub)\}$$

Pero esa definición ni sirve para probar que L_7 es regular. Para probar que es regular tenemos que dar una manera de definir L_7 utilizando los lenguajes regulares básicos $\varnothing$, $\{\varepsilon\}$, $\{a\}$, $\{b\}$ y $\{c\}$ y las operaciones de concatenación, unión y clausura para lenguajes. Una definición del lenguaje L_7 utilizando esos elementos sería la siguiente: $L_7 = (\{a\} \cup \{b\} \cup \{c\})^*\{b\}$. Otra manera de expresar lo mismo sería utilizando los lenguajes regulares $H = \{b\}$ y $L_3 = \{a, b, c\}$: $L_7 = (L_3)^*H$. Otra posibilidad para definir L_7 de manera abreviada sería $L_7 = (a + b + c)^*b$.

8. A continuación vamos a probar que el lenguaje L_8 formado por todas las palabras que empiezan por a y terminan en b es regular. Una manera de definir formalmente este lenguaje es la siguiente:

$$L_8 = \{w | w \in A^* \wedge \exists u (u \in A^* \wedge w = aub)\}$$

Pero esa definición ni sirve para probar que L_8 es regular. Para probar que es regular tenemos que dar una manera de definir L_8 utilizando los lenguajes regulares básicos $\varnothing$, $\{\varepsilon\}$, $\{a\}$, $\{b\}$ y $\{c\}$ y las operaciones de concatenación, unión y clausura para lenguajes. Una posible definición del lenguaje L_8 utilizando esos elementos sería la siguiente: $L_8 = \{a\}(\{a\} \cup \{b\} \cup \{c\})^*\{b\}$. Otra manera de expresar lo mismo sería utilizando los lenguajes regulares $H_1 = \{a\}$, $H_2 = \{b\}$ y $L_3 = \{a, b, c\}$: $L_8 = H_1(L_3)^*H_2$. Otra posibilidad para definir L_8 de manera abreviada sería $L_8 = a(a + b + c)^*b$. Como anteriormente hemos probado que L_6 y L_7 son regulares, definir L_8 como L_6L_7 sirve también para probar que L_8 es regular.

9. A continuación vamos a probar que el lenguaje L_9 formado por todas las palabras que no contienen ninguna c y en las que todas las a-s (si hay a-s) aparecen en el lado izquierdo y todas las b-s (si hay b-s) aparecen en el lado derecho (ε, $aaaa$, bbb, $aabbb$, etc.) es regular. Una manera de definir formalmente este lenguaje es la siguiente:

$$L_9 = \{w | w \in A^* \wedge \exists u, v (u \in A^* \wedge v \in A^* \wedge |u| = |u|_a \wedge |v| = |v|_b \wedge w = uv)\}$$

Pero esa definición ni sirve para probar que L_9 es regular. Para probar que es regular tenemos que dar una manera de definir L_9 utilizando los lenguajes regulares básicos $\varnothing$, $\{\varepsilon\}$, $\{a\}$, $\{b\}$ y $\{c\}$ y las operaciones de concatenación, unión y clausura para lenguajes. Una posible definición del lenguaje L_9 utilizando esos elementos sería la siguiente: $L_9 = \{a\}^*\{b\}^*$. Otra manera de expresar lo mismo sería utilizando los lenguajes regulares $H_1 = \{a\}$ y $H_2 = \{b\}$: $L_9 = H_1H_2$. Otra posibilidad para definir L_9 de manera abreviada sería $L_9 = a^*b^*$.

10. A continuación vamos a probar que el lenguaje L_{10} formado por todas las palabras que no contienen ninguna c y en las que las a-s y las b-s no están mezcladas (ε, $aaaa$, bbb, $aabbb$, $bbbaaaa$, etc.) es regular. Una manera de definir formalmente este lenguaje es la siguiente:

$$L_{10} = \{w | w \in A^* \wedge \exists u, v (u \in A^* \wedge v \in A^* \wedge |u| = |u|_a \wedge |v| = |v|_b \wedge (w = uv \vee w = vu))\}$$

Pero esa definición ni sirve para probar que L_{10} es regular. Para probar que es regular tenemos que dar una manera de definir L_{10} utilizando los lenguajes regulares básicos $\varnothing$, $\{\varepsilon\}$, $\{a\}$, $\{b\}$ y $\{c\}$ y las operaciones de concatenación, unión y clausura para lenguajes. Una posible definición del lenguaje L_{10} utilizando esos elementos sería la siguiente: $L_{10} = (\{a\}^*\{b\}^*) \cup (\{b\}^*\{a\}^*)$. Otra manera de expresar lo mismo sería utilizando los lenguajes regulares $H_1 = \{a\}$ y $H_2 = \{b\}$: $L_{10} = H_1H_2 \cup H_2H_1$. Otra posibilidad para definir L_{10} de manera abreviada sería $L_{10} = (a^*b^*) + (b^*a^*)$.

11. A continuación vamos a probar que el lenguaje L_{11} formado por todas las palabras que contienen exactamente dos a-s y cualquier cantidad de b-s y c-s (ε, aa, $abacc$, $bbbaabbb$, $cabbbabb$, $babbbca$, etc.) es regular. Una manera de definir formalmente este lenguaje es la siguiente:

$$L_{11} = \{w | w \in A^* \wedge |w|_a = 2\}$$

Pero esa definición ni sirve para probar que L_{11} es regular. Para probar que es regular tenemos que dar una manera de definir L_{11} utilizando los lenguajes regulares básicos $\varnothing$, $\{\varepsilon\}$, $\{a\}$, $\{b\}$ y $\{c\}$ y las operaciones de concatenación, unión y clausura para lenguajes. Una posible definición del lenguaje L_{11} utilizando esos elementos sería la siguiente:

$$L_{11} = (\{b\} \cup \{c\})^* \{a\} (\{b\} \cup \{c\})^* \{a\} (\{b\} \cup \{c\})^*$$

Otra posibilidad para definir L_{11} de manera abreviada sería $L_{11} = (b + c)^* a (b + c)^* a (b + c)^*$.

12. A continuación vamos a probar que el lenguaje L_{12} formado por todas las palabras en las que el número de a-s es par (ε, aa, $abacc$, $aaaa$, $ababaabbbcaa$, etc.) es regular. Una manera de definir formalmente este lenguaje es la siguiente:

$$L_{12} = \{w | w \in A^* \wedge |w|_a \bmod 2 = 0\}$$

Pero esa definición ni sirve para probar que L_{12} es regular. Para probar que es regular tenemos que dar una manera de definir L_{12} utilizando los lenguajes regulares básicos $\varnothing$, $\{\varepsilon\}$, $\{a\}$, $\{b\}$ y $\{c\}$ y las operaciones de concatenación, unión y clausura para lenguajes. Una posible definición del lenguaje L_{12} utilizando esos elementos y teniendo en cuenta que el lenguaje L_{11} formado por las palabras que tienen dos a's sería la siguiente: $L_{12} = (L_{11})^*$. Se ha utilizado la operación $*$ aplicada a un lenguaje regular, el lenguaje regular L_{11}.

3.8 Cálculo del autómata finito (AFD, AFND o ε-AFND) correspondiente a un lenguaje regular

En este apartado se presentará un método que, dado un lenguaje regular permite obtener de manera sistemática el autómata finito (o AF) correspondiente.

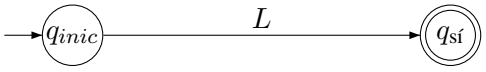
Cuando hablamos de autómatas finitos o de AF's, nos estamos refiriendo tanto a AFD's como a AFND's como a ε-AFND's. Por tanto, el método obtiene un AF que puede ser de cualquiera de los tres tipos de autómata finito que hemos visto (AFD, AFND o ε-AFND). Normalmente obtiene un AFND o un ε-AFND.

Tal como se verá al aplicar el método, nos darán un lenguaje regular expresado utilizando los lenguajes más elementales posibles y las operaciones de concatenación, clausura ($*$) y unión ($+$). Si el alfabeto es $A = \{a, b, c\}$ los lenguajes básicos serán el lenguaje vacío $\varnothing$, el lenguaje que solo contiene la palabra vacía $\{\varepsilon\}$ y los lenguajes $\{a\}$, $\{b\}$ y $\{c\}$. Tal como se ha dicho en la sección anterior, se prescindirá del uso de las llaves. Por ejemplo, los lenguajes regulares vendrán escritos de la forma $(aa(b^*)) + ((bb)^*)$ en vez de $(\{a\}\{a\}(\{b\}^*)) \cup ((\{b\}\{b\})^*)$. Se intentará evitar el uso de paréntesis en muchos casos teniendo en cuenta que $*$ es el que mayor prioridad tiene, luego viene la concatenación y por último $+$, es decir, la unión. Por tanto, en vez de escribir $(aa(b^*)) + ((bb)^*)$ escribiremos $aab^* + (bb)^*$.

3.8.1 Método de generación de estados y cálculo del AF correspondiente

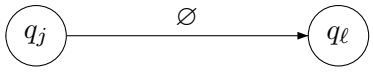
A continuación se indica el procedimiento a seguir para generar un AF a partir de un lenguaje regular:

Dado un lenguaje regular L expresado en el formato indicado anteriormente, el primer paso es generak un AF de la forma:



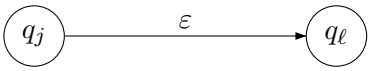
A continuación habrá que ir aplicando los siguientes esquemas generales hasta que cada transición o arco este etiquecado por un único símbolo del alfabeto A o por ε.

- Si tenemos que L es el lenguaje vacío, es decir, $L = \varnothing$



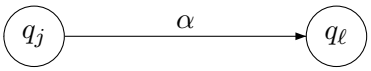
entonces se ha de eliminar el arco entre q_j y q_ℓ

- Si tenemos que L es el lenguaje formado por la palabra vacía, es decir, $L = \varepsilon$



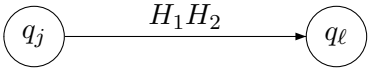
entonces se deja tal cual.

- Si tenemos que L es un lenguaje básico formado por un símbolo α de A, es decir, $L = \alpha$

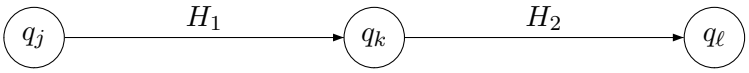


entonces se deja tal cual.

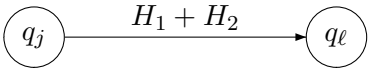
- Si tenemos que L está definido como la concatenación de otros dos lenguajes regulares, es decir, $L = H_1 H_2$



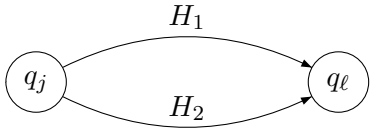
entonces se ha de generar un nuevo estado



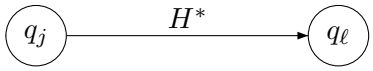
- Si tenemos que L está definido como la unión de otros dos lenguajes regulares, es decir, $L = H_1 + H_2$



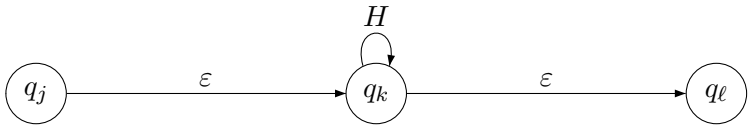
entonces se ha de generar un nuevo arco



- Si tenemos que L está definido como la clausura de otro lenguaje regular, es decir, $L = H^*$



entonces se ha de generar un nuevo estado

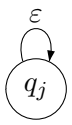


A continuación se indica cómo se han de aplicar esos esquemas generales cuando q_j y q_ℓ son el mismo estado.

- Si tenemos que L es el lenguaje vacío, es decir, $L = \varnothing$

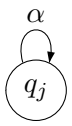
entonces se ha de eliminar el bucle de q_j

- Si tenemos que L es el lenguaje formado por la palabra vacía, es decir, $L = \varepsilon$



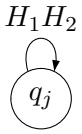
entonces se deja tal cual.

- Si tenemos que L es un lenguaje básico formado por un símbolo α de A, es decir, $L = \alpha$

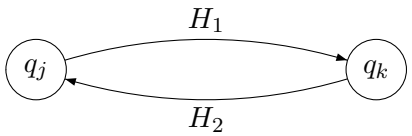


entonces se deja tal cual.

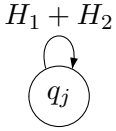
- Si tenemos que L está definido como la concatenación de otros dos lenguajes regulares, es decir, $L = H_1 H_2$



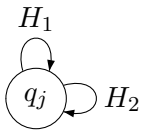
entonces se ha de generar un nuevo estado



- Si tenemos que L está definido como la unión de otros dos lenguajes regulares, es decir, $L = H_1 + H_2$



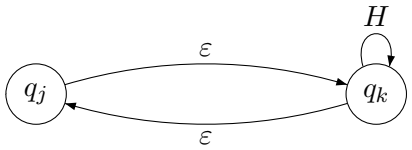
entonces se ha de generar un nuevo bucle



Habitualmente representaríamos eso con un único arco y separando H_1 y H_2 con una coma, pero en este método eso sólo se puede hacer si H_1 y H_2 son dos elementos de $A \cup \{\varepsilon\}$. En general hay que que generar dos arcos para que H_1 y H_2 puedan ser desarrollados de manera separada.

- Si tenemos que L está definido como la clausura de otro lenguaje regular, es decir, $L = H^*$

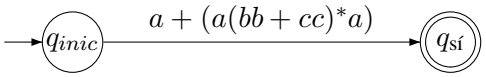
entonces se ha de generar un nuevo estado



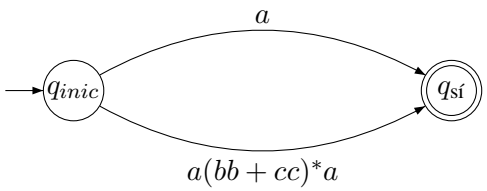
3.8.2 Ejemplo: Cálculo del AF correspondiente al lenguaje regular $a + (a(bb + cc)^*a)$

Se desea diseñar un AF para el siguiente lenguaje regular: $L = a + (a(bb + cc)^*a)$

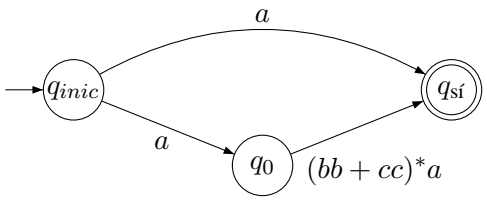
El primer paso consiste en generar los estados q_{inic} y $q_{\text{sí}}$ con un arco que va del primero al segundo y etiquetado con L:



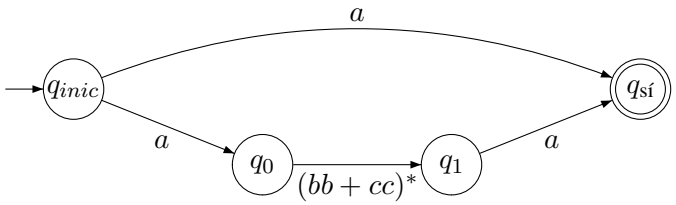
En primer lugar tenemos una unión entre $H_1 = a$ y $H_2 = a(bb + cc)^*a$. Por tanto, se pondrán dos arcos de q_{inic} a $q_{\text{sí}}$.



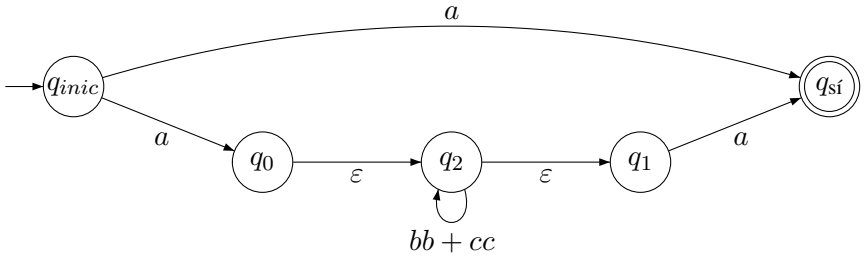
El arco correspondiente a H_1 no hay que desarrollarlo más porque tenemos solo un símbolo de A. Por tanto nos centramos en el arco correspondiente a H_2. En H_2 tenemos la concatenación de los lenguajes regulares $G_1 = a$ y $G_2 = (bb + cc)^*a$. Por tanto creamos un estado nuevo.



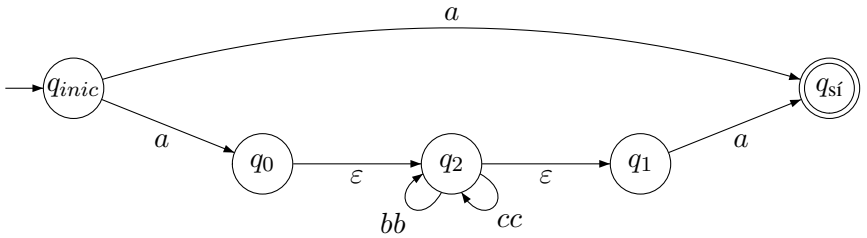
El arco correspondiente a G_1 no hay que desarrollarlo más porque tenemos solo un símbolo de A. Por tanto nos centramos en el arco correspondiente a G_2. En G_2 tenemos la concatenación de los lenguajes regulares $F_1 = (bb + cc)^*$ y $F_2 = a$. Por tanto creamos un estado nuevo.



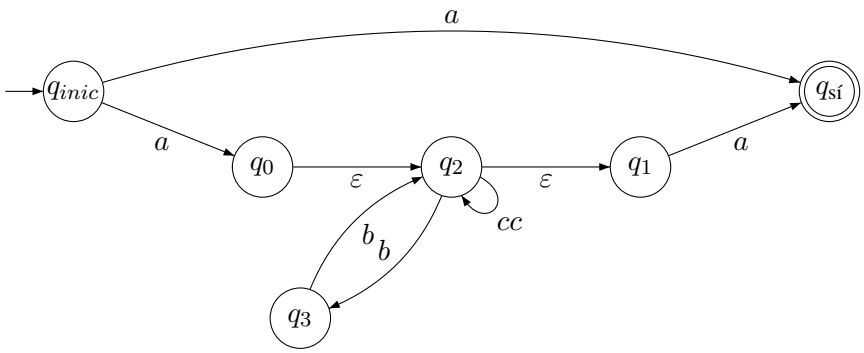
El arco correspondiente a F_2 no hay que desarrollarlo más porque tenemos solo un símbolo de A. Por tanto nos centramos en el arco correspondiente a F_1. En F_1 tenemos la clausura del lenguaje regular $I = (bb + cc)$. Por tanto creamos un estado nuevo.



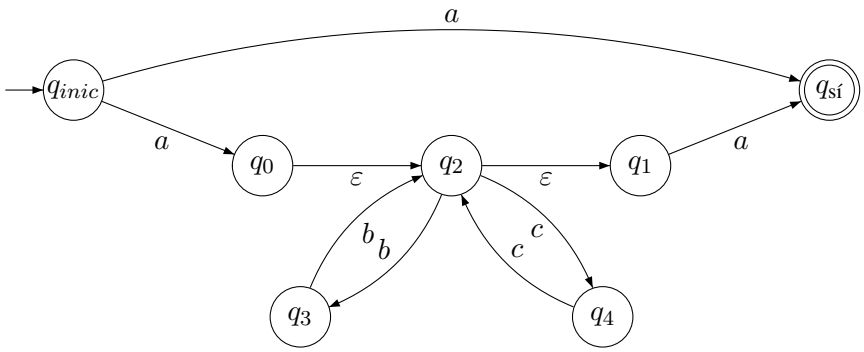
Ahora toca desarrollar el bucle de q_2. El lenguaje $I = bb + cc$ es la unión de los lenguajes $Z_1 = bb$ y $Z_2 = cc$. Por tanto se necesitan dos bucles en q_2.



El lenguaje Z_1 es la concatenación de $Y_1 = b$ e $Y_2 = b$. Por tanto añadimos un estado nuevo q_3:



El lenguaje Z_2 es la concatenación de $X_1 = c$ e $X_2 = c$. Por tanto añadimos un estado nuevo q_4:



Como en todos los arcos se tiene un símbolo de A o ε, se ha terminado la construcción del AF correspondiente al lenguaje regular $a + (a(bb + cc)^*a)$.

3.9 Automata finituak eta lengoaia erregularren arteko baliokidetasuna

Gai honetako aurreko ataletan esan den bezala, automata finitu (determinista edo ez determinista) bakoitzak lengoaia bat definitzen du. Hitz baten aurrean automatak "Bai" erantzuten badu, orduan hitz hori lengoaiako da eta "Ez" erantzuten badu, ez da lengoaiakoa.

Edozein automata finitu hartuz, automata horri dagokion lengoaia erregularra izango da. Eta edozein lengoaia erregular hartuz, lengoaia hori onartzen duen automata finitua beti existituko da. Beraz lengoaia erregularrak automata finituen bidez defini daitezkeen lengoaiak dira.

3.10 Lengoaia ez erregularrak

Automata finitu eta lengoaia erregularren arteko baliokidetasunaren aurrean honako galdera hau sortzen da: Lengoaia denak erregularrak al dira? edo beste era batera esanda, 2^{A^*} multzoko edozein L lengoaia hartzen badugu, L onartzen duen automata finitua kasu denetan existitzen al da?

Erantzuna ezezkoa da.

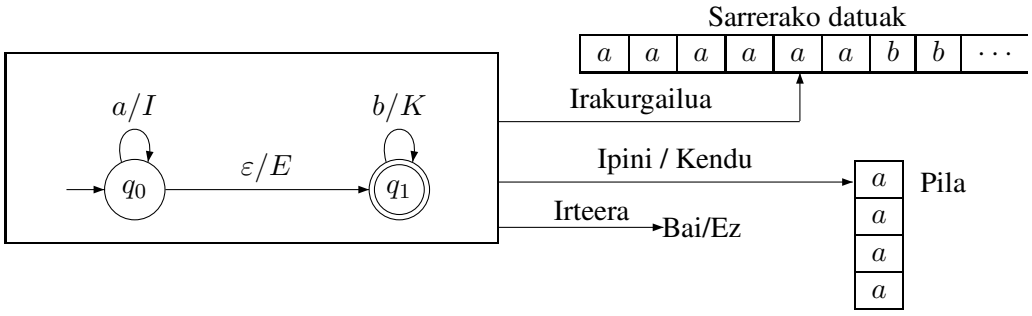
3.10.1 Kontesturik gabeko lengoaiak eta piladun automatak

Jarraian datorren lengoaiarentzat ez dago automata finiturik:

$$G_1 = \{w \in A^* \mid w = a^n b^n \wedge n \geq 0\}$$

a eta b-kopuru bera duten eta gainera a-ren eta b-ren agerpenak nahastu gabe dituzten hitzez osatuta dago G_1 lengoaia. Beraz $G_1 = \{\varepsilon, ab, aabb, aaabbb, aaaabbbb, \ldots\}$. G_1 lengoaia L_9 lengoaiaren azpilengoaia da. L_9 lengoaia erregularra izan arren, G_1 ez da erregularra. Kontuan hartu a-kopurua eta b-kopurua edozein izan daitekeela (0, 1, 2, 3, eta abar). Automata finituetan memoria egoeren bidez lortzen da eta automata finituak gauza batzuk gogoratzeko gai dira. Adibidez, a-kopurua bikoitia al den gogoratzeko gai den automata defini dezakegu. Baina G_1 lengoaiaren kasuan lehenengo a-kopurua zenbatu behar da eta gero kopuru hori b kopuruarekin alderatu behar da, baina automata finitu batek ezin dezake egin hori, ezin baitezake zenbatu hasieratik ez bada ezagutzen zenbatu beharreko kopurua.

Beraz lengoaia batzuk ez dira erregularrak eta automata finituen bidez ezin dira lengoaia denak definitu. Esate baterako G_1 lengoaiarentzat jarraian erakusten den piladun automata ez determinista beharko genuke.



Automata hau pila hutsarekin abiatzen da eta hasierako a denak pilan gordez joaten da b lehenengo aldiz agertu arte. Gero, irakurtzen den b bakoitzeko a bat kentzen da pilatik. Hitza bukatzean pila hutsik badago, badakigu a eta b kopurua berdina izan dela. Hitza bukatzean pila hutsik ez badago, orduan badakigu a-kopurua b-kopurua baino handiagoa izan dela. Bestalde, b-ren agerpen denak irakurri baino lehen pila hutsik gelditzen bada, badakigu b-kopurua a-kopurua baino handiagoa dela. Gainera uneren batean c agertzen bada, hitza ez da onartuko. Era berean, b agertu ondoren berriro a agertzen bada, hitza ez da onartuko. Piladun automata hau determinista ez denez, egoeratatik era desberdinetan mugi gaitezke hitza irakurtzean. Hitza irakurriz goazenean, egoeratan zehar mugitzeko era horietako bat ondo bukatuko da makina gelditzean bukaerako egoeran baldin bagaude, hitz dena irakurri bada, pilatik elementu bat kendu behar zen bakoitzean kendu ahal izan bada eta azkenean pila hutsik gelditu bada. Pilatik a bat kendu behar denean pila hutsik baldin badago, automata blokeatu egingo da. Gainera pilatik a bat kendu ahal izateko sarrerako hitzean b bat irakurri behar da. Hitz bat irakurtzean egoerak zeharkatzeko era bat baino gehiago daudenean, nahikoa da era bat ondo bukatzearekin hitza onartzeko. Era denak gaizki bukatzen badira, orduan hitza ez da onartuko.

Trantsizioetan α/β erako etiketak ditugu. Etiketa horietan α elementua alfabetoko sinbolo bat edo hitz hutsa da (beraz, α elementua $A \cup \{\varepsilon\}$ multzoko sinbolo bat da) eta β elementua $\{E, I, K\}$

multzoko sinbolo bat da. E sinboloak pilan ez dela ezer egin behar adierazten du, I sinboloak pilaren gainean (gailurrean) a sinboloa ipini behar dela adierazten du eta K sinboloak pilaren gailurrean dagoen a sinboloa kendu behar dela adierazten du. Adibidez, $aabb$ hitzarentzat egoeratatik zehar mugitzeko era bat baino gehiago daude:

1. $(aabb, q_0, [\,]) \rightarrow (abb, q_0, [a]) \rightarrow (bb, q_0, [a, a]) \rightarrow (bb, q_1, [a, a]) \rightarrow (b, q_1, [a]) \rightarrow (\varepsilon, q_1, [\,])$. Bide hau ondo bukatu da.

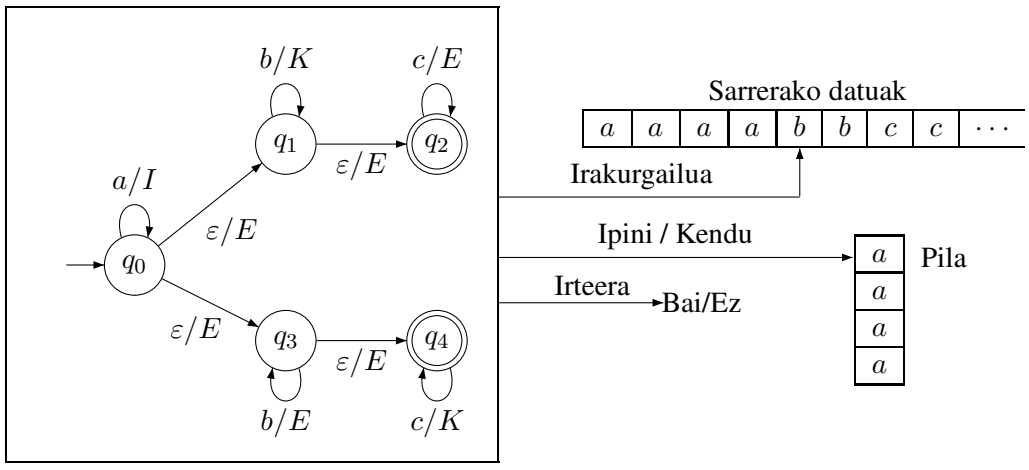
2. $(aabb, q_0, []) \rightarrow (abb, q_0, [a]) \rightarrow (abb, q_1, [a])$. Kasu honetan ez da ondo bukatu. Azkarregi pasatu da q_1 egoerara eta jarraian irakurri beharreko sinboloa a da eta pilan ere a dago eta ondorioz ezin da pilako a hori ezabatu eta automata blokeatu egingo da eta hitza onartzeko baldintzak ez dira beteko. Izan ere pilatik a bat ezabatzeko, irakurtzeko gelditzen den hitz zatiko lehenengo sinboloak b izan behar du q_1 egoeran gaudenean. Gainera q_1 egoeran a-rik ezin da irakurri.

3. $(aabb, q_0, []) \rightarrow (aabb, q_1, [])$. Kasu honetan ere ez da ondo bukatu, izan ere azkarregi pasatu da q_1 egoerara eta jarraian irakurri beharreko sinboloa a da, baina q_1 egoeran a agertzen bada, automata blokeatu egiten da eta hitza onartzeko baldintzak ez dira betetzen.

Kasu batean ondo bukatzen denez, badakigu $aabb$ hitza G_1 lengoaiakoa dela. Esate baterako $aaab$ hitza hartzen badugu, kasu denak gaizki bukatuko dira eta ondorioz badakigu hitz hori ez dela G_1 lengoaiakoa.

Piladun automata ez deterministen bidez definigarraiak diren lengoaiak **kontesturik gabeko lengoaiak dira**. Jarraian piladun automata ez determinista behar duten kontesturik gabeko beste bi lengoaia emango dira: G_2 eta G_3.

- $G_2 = \{w \in A^* \mid w = a^i b^j c^k \wedge i \geq 0 \wedge j \geq 0 \wedge k \geq 0 \wedge (i = j \vee i = k)\}$. Beraz, G_2 lengoaian ε, $aaabbb$, $aaabbbc$, $aaaccc$, $aaabccc$ eta $aabbcc$ erako hitzak daude. Baina, adibidez, $abbccc$ eta aab ez dira G_2 lengoaiakoak.



Hasieran pila hutsik egongo da eta automata sarrerako hitzeko a-ren agerpenak pilan ipiniz joango da b edo c agertu arte. b edo c agertzen denean bi aukera daude:

(a) q_1, q_2 bidea jarraitu eta hitzean irakurtzen den b bakoitzeko pilatik a bat kendu. Irakurtzen ari den hitzeko b denak bukatzean pilako a denak ere kentzea lortu bada, ez dago c-kopurua kontrolatu beharrik, bai baitakigu a-kopurua eta b-kopurua berdinak direla.

(b) q_3, q_4 bidea jarraitu eta hitzeko b-ak irakurri pilatik a-rik kendu gabe eta c lehenengo aldiz agertzen denean, hitzean irakurtzen den c bakoitzeko pilatik a bat kendu. Irakurtzen ari den hitza bukatzean (hau da, c denak bukatzean) pilako a denak ere kentzea lortu bada, badakigu a-kopurua eta c-kopurua berdinak direla.

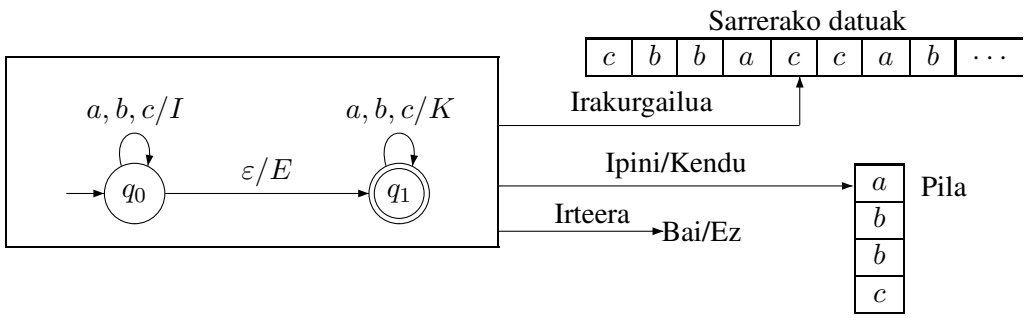
Piladun automata hau determinista ez denez, egoeratatik era desberdinetan mugi gaitezke hitza irakurtzean. Hitza irakurriz goazenean, egoeratan zehar mugitzeko era horietako bat ondo bukatuko da makina gelditzean bukaerako egoeran baldin bagaude, hitz dena irakurri bada, pilatik elementu bat kendu behar zen bakoitzean kendu ahal izan bada eta azkenean pila hutsik gelditu bada. Pilatik a bat kendu behar denean pila hutsik baldin badago, automata blokeatu egingo da eta hitza ez da onartuko. Gainera pilatik a bat kendu ahal izateko sarrerako hitzean b bat irakurri behar da q_1 egoeran baldin bagaude eta c bat irakurri behar da q_4 egoeran baldin bagaude. Hitz bat irakurtzean egoerak zeharkatzeko era bat baino gehiago baldin badaude, nahikoa da era bat ondo bukatzearekin hitza onartzeko. Era denak gaizki bukatzen badira, orduan hitza ez da onartuko. E sinboloak pilan ez dela ezer egin behar adierazten du, I sinboloak pilaren gainean (gailurrean) a sinboloa ipini behar dela adierazten du eta K sinboloak pilaren gailurrean dagoen a sinboloa kendu behar dela adierazten du.

Adibidez, $aabcc$ hitzarentzat aukera bat baino gehiago daude:

1. $(aabcc, q_0, [\,]) \to (abcc, q_0, [a]) \to (bcc, q_0, [a, a]) \to (bcc, q_3, [a, a]) \to (cc, q_3, [a, a]) \to (cc, q_4, [a, a]) \to (c, q_4, [a]) \to (\varepsilon, q_4, [\,])$. Kasu honetan ondo bukatu da. a-kopurua eta c-kopurua konparatu dira eta berdinak dira.
2. $(aabcc, q_0, [\,]) \to (abcc, q_0, [a]) \to (bcc, q_0, [a, a]) \to (bcc, q_1, [a, a]) \to (cc, q_1, [a]) \to (cc, q_2, [a]) \to (c, q_2, [a]) \to (\varepsilon, q_2, [a])$. Kasu honetan gaizki joan da. Automata q_1 egoerara pasatu da a-kopurua eta b-kopurua konparatzeko asmoarekin, baina bi kopuru horiek ez dira berdinak. b-ren agerpenak bukatu direnean pila ez dago hutsik, a sinbolo bat dago. Jarraian c-ak irakurtzen dira pilan ezer aldatu gabe eta hitza bukatzean pila ez dago hutsik. Ondo bukatzeko baldintzak ez dira betetzen.
3. $(aabcc, q_0, [\,]) \to (abcc, q_0, [a]) \to (abcc, q_3, [a]) \to (abcc, q_4, [a])$. Kasu honetan gaizki joan da q_3 egoerara azkarregi pasatzeagatik. q_3 egoeran a-rik ez denez onartzen, aukera bakarra ezer irakurri gabe q_4-ra pasatzea da. Baina q_4 egoeran a-rik ez denez onartzen automata blokeatu egin da. Ondo bukatzeko baldintzak ez dira betetzen.

Gaizki bukatzen diren kasu gehiago ere badaude, baina kasu batean ondo bukatzen denez, $aabcc$ hitza G_2 lengoaiakoa da. Baina $aaab$ hitza hartzen badugu, kasu denak gaizki bukatuko dira eta hitza ez da onartuko.

- $G_3 = \{w \in A^* \mid \exists v(v \in A^* \wedge w = vv^R)\}$. G_3 lengoaiako hitz batzuk ε, $\underline{aa}aa$, $\underline{abcc}ba$, $\underline{ab}ba$ eta $\underline{ccbaba}ababcc$ dira. Bestalde, adibidez $aaca$ eta aab ez dira G_3 lengoaiakoak.



vv^R egitura duen hitz bat onartzeko, ideia v-ren sinboloak pilan gordez joatea eta v^R irakurtzean pilako sinboloak kenduz joatea da. Piladun automata hau determinista ez denez, egoeratatik era desberdinetan mugi gaitezke hitza irakurtzean. Hitza irakurriz goazenean, egoeratan zehar mugitzeko era horietako bat ondo bukatuko da makina gelditzean bukaerako egoeran baldin bagaude, hitz dena irakurri bada, pilatik elementu bat kendu behar zen bakoitzean kendu ahal izan bada eta azkenean pila hutsik gelditu bada. Pilatik sinbolo bat kendu behar denean pila hutsik baldin badago, automata blokeatu egingo da eta hitza ez da onartuko. Gainera pilatik sinbolo bat kendu ahal izateko, sinbolo horrek irakurtzeko gelditzen den sarrerako hitzaren zatiko lehenengo sinboloaren berdina izan behar du. Hitz bat irakurtzean egoerak zeharkatzeko era bat baino gehiago baldin badaude, nahikoa da era bat ondo bukatzearekin hitza onartzeko. Era denak gaizki bukatzen badira, orduan hitza ez da onartuko. E sinboloak pilan ez dela ezer egin behar adierazten du, I sinboloak pilaren gainean (gailurrean) azkena irakurri den sinboloa ipini behar dela adierazten du eta K sinboloak pilaren gailurrean dagoen sinboloa kendu behar dela adierazten du, sinbolo hori eta irakurtzeko gelditzen den hitz zatiko lehenengo sinboloa berdinak badira.

Adibidez, $acca$ hitzarentzat aukera bat baino gehiago daude:

1. $(acca, q_0, [\,]) \rightarrow (cca, q_0, [a]) \rightarrow (ca, q_0, [c, a]) \rightarrow (ca, q_1, [c, a]) \rightarrow (a, q_1, [a]) \rightarrow (\varepsilon, q_1, [\,])$. Kasu honetan ondo bukatu da.
2. $(acca, q_0, []) \rightarrow (cca, q_0, [a]) \rightarrow (cca, q_1, [a])$. Kasu honetan gaizki joan da q_1 egoerara azkarregi pasatu delako. Pilatik sinbolo bat kentzeko hitzeko lehenengo sinboloaren berdina izan behar duenez, automata blokeatu egingo da hitzean c eta pilan a baitago. Beraz, ondo bukatzeko baldintzak ez dira betetzen.
3. $(acca, q_0, []) \rightarrow (cca, q_0, [a]) \rightarrow (ca, q_0, [c, a]) \rightarrow (a, q_0, [c, c, a]) \rightarrow (a, q_1, [c, c, a])$. Kasu honetan gaizki joan da q_1 egoerara beranduegi pasatu delako. Jarraian hitzean a dago eta pilan c. Pilatik elementu bat kentzeko elementu horrek hitzeko lehenengo elementuaren berdina izan behar duenez, automata blokeatu egingo da. Beraz, ondo bukatzeko baldintzak ez dira betetzen.
4. ...

Gaizki bukatzen diren kasu gehiago ere badaude, baina kasu batean ondo bukatzen denez, $acca$ hitza G_3 lengoaiakoa da. Baina $bbba$ hitza hartzen badugu, kasu denak gaizki bukatuko dira eta hitza ez da onartuko.

3.10.2 Piladun automaten bidez definigarriak ez diren lengoaiak

Piladun automatetan bakarrik bi eragiketa buru daitezke pilan: elementu bat ipini gailurrean eta gailurreko elementua kendu. Beraz, gailurrean ez dauden elementuak ezin dira hartu. Pilako gailurrean ez dagoen x elementu bat hartzeko era bakarra lehenengo bere gainean daudenak kentzea da. Pilak duten muga hori dela eta, piladun automatak ere kalkulu-gaitasun mugatua dute.

Jarraian erakusten diren lengoaiak ezin dira definitu piladun automaten bidez eta ondorioz ez dira kontesturik gabeko lengoaiak:

- $F_1 = \{w \in A^* \mid w = a^n b^n c^n \wedge n \geq 0\}$. Adibidez, ε, abc, $aabbcc$ eta $aaabbbccc$ hitzak F_1 lengoaiakoak dira. Baina, esate baterako, $abbccc$ eta aab ez dira F_1 lengoaiakoak. Lengoaia hau G_2 lengoaiaren antzekoa da: G_2 lengoaian a-kopurua b-kopuruarekin edo c-kopuruarekin konparatu behar zen, baina ez biekin. Baina F_1 lengoaian a-kopurua b-kopuruarekin eta c-kopuruarekin konparatu behar zen, biekin. Beraz, a sinboloaren agerpenak pilan gordetzen baditugu eta gero b-ak irakurtzean pilako a horiek kenduz joaten bagara, c-ak iristean a denak ezabatuta egongo dira pilatik eta a-kopurua zein zen ez dugu jakingo.

- $F_2 = \{w \in A^* \mid w = a^i b^j c^k \wedge 0 \leq i \leq j \leq k\}$. Adibidez, ε, abc, $aabbcc$, $bbccc$ eta $abbccc$ hitzak F_2 lengoaiakoak dira, baina $aaabbccc$ eta aab ez. Lengoaia honen arazoa F_1 lengoaiako arazoaren antzekoa da.

- $F_3 = \{w \in A^* \mid \exists v(v \in A^* \wedge w = vv)\}$. Adibidez,$\varepsilon$, $\underline{aa}aa$, $\underline{abc}abc$, $\underline{ab}ab$ eta $\underline{ccbaba}ccbaba$ hitzak F_3 lengoaiakoak dira baina $aaca$, $acca$ eta aab ez. Pilan sinboloak gordetzeko ordenagatik eta pilako gailurreko elementua bakarrik irakur daitekeenez, hitz bat bi azpihitz berdinez osatuta al dagoen ala ez jakiteko erarik ez dago piladun automaten bidez.

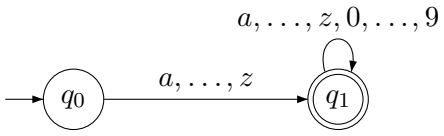
3.11 Lengoaia erregularren eta kontesturik gabeko lengoaien aplikazioak

Lengoaia erregularrak eta kontesturik gabeko lengoaiak, eta ondorioz, automata finituak eta piladun automatak, konputagarritasuna aztertzeko erabil daitezke teoria mailan, baina beste arlo batzuetan ere erabil daitezke.

Lengoaia erregularrak (eta automata finituak) programazio-lengoaietan erabiltzen diren aldagaien eta izenen egitura kontrolatzeko erabil daitezke (analisi lexikoa). Adibidez, programazio-lengoaia bateko aldagaien izenek letra minuskula batez hasi behar badute eta gero letra minuskulez eta digituz osatuta egon badaitezke, aldagaien izenek $A = \{a, b, \ldots, z, 0, 1 \ldots, 9\}$ alfabetoaren gainean definitutako honako V lengoaia hau osatzen dute:

$$H_1 = \{w \in A^* \mid w = a \vee w = b \vee \ldots \vee w = z\}$$
$$H_2 = A^*$$
$$V = H_1 H_2$$

V lengoaiarentzat honako AFEDa izango genuke:

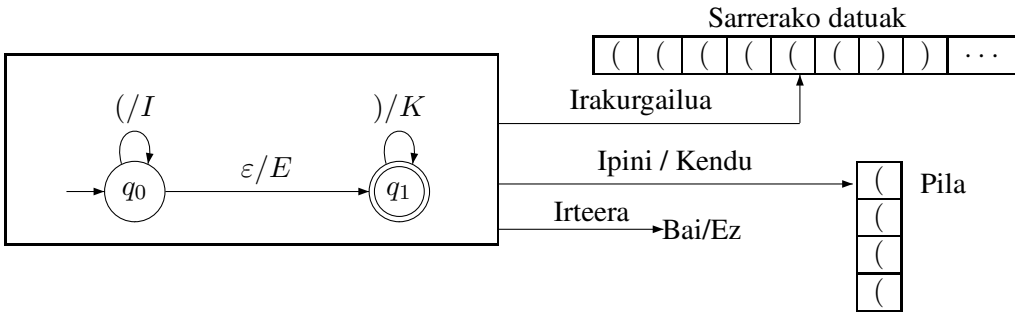


Bestalde, kontesturik gabeko lengoaiak konpiladoreak eraikitzeko erabiltzen dira (analisi sintaktikoa). Adibide bat parentesiak ondo al dauden kontrolatzen duen piladun automatarena izango litzateke. Alfabetoa $A = \{(,)\}$ izango litzateke eta kontesturik gabeko lengoaia honako hau izango litzateke:

$$L = \{w \in A^* \mid w = (^n)^n \wedge n \geq 0\}$$

Beraz, adibidez $\varepsilon, (), (()), ((()))$ eta $((((()))))$ hitzak lengoaia horretakoak dira eta $((,)(, ()()$ eta $(()))$ ez dira.

Lengoaia horri dagokion piladun automata honako hau izango litzateke:



‘(’ sinboloak pilan ipini edo gordeko dira eta gero ‘)’ sinbolo bakoitzeko ‘(’ bat kenduko da pilatik.