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

4. Gaia: Turing-en makinak

JOSE GAINTZARAIN IBARMIA

Azken eguneraketa: 2015-9-2

GAIEN AURKIBIDEA

4. TURING-EN MAKINAK: LENGOAIA ERABAKIGARRIAK, LENGOAIA BEREIZGARRIAK ETA LENGOAIA BEREIZTEZINAK

4.1 Turing makinak

4.1.1 Turing-en makinen ezaugarriak

Automata finituen bidez definitu ezin diren lengoaiak badaudela ikusi da 3. gaian. Automata finituetan egoerak dira memoria bakarra. Horrek automata finituen kalkulu-gaitasuna asko mugatzen du, memoria finitua eta oso mugatua baita. Arazo hori gainditzeko, lehenengo proposamena piladun automatak erabiltzea izan daiteke. Printzipioz infinitua den pila bat ipintzean, lengoaia gehiago definitzea lortzen da, izan ere, azpihitzen egiturak konparatzea edo zenbaketaren bat burutzea ahalbidetzen du pilak. Hala ere pilak murriztapen batzuk dituzte erabilerari dagokionez. Piletan elementuak gailurrean ipini behar dira beti eta elementuren bat hartzerakoan, beti gailurrekoa hartu behar da. Murriztapen horiek direla eta, piladun automaten bidez definitu ezin diren lengoaiak badaude (xehetasun gehiago nahi izanez gero, 3. gaian emandako adibideak begiratu).

Piladun automatateen mugak gainditzeko, jarraian beste makina-mota bat aurkeztuko da: Turing-en makinak. Makina hauen ahalmena ordenagailuen ahalmenaren parekoa da, beraz ordenagailuen kalkulu-gaitasun bera dute. Ordenagailu bat azken batean programa bat da (sistema eragilea) eta programa hori era desberdinetako aginduak burutzeko gai da. Egoera era sinplifikatuan ikusiz, ordenagailuak (eta baita Turing-en makinak ere) sarrerako datu bezala karaktere-kateak hartzen dituzten programak direla kontsidera dezakegu. Sarrera bezala emandako karaktere-kate horien bidez aginduak (hau da programak) eta datuak pasatzen dizkiogu. Ordenagailuek (eta Turing-en makinak) karaktere-kateen bidez emandako aginduekin kalkuluak burutuko dituzte.

Turing-en makinetan Sarrera/Irteera-ko gailua memoria bezala erabiltzen da. Turing-en makinetan Sarrera/Irteerako gailu hori infinitua dela kontsideratzen da eta makinak memoria horretako edozein posiziotako informazioa eskura dezakeela inolako mugarik gabe. Beraz, gailu horretako posizioetan zehar murrizketarik gabe mugi daiteke, posizio horietan dagoena irakurriz edo posizio horietan idatziz. Beraz, Sarrera/Irteerako gailua agindu bidezko programazio-lengoaietako bektoreak edo array-ak bezalakoa dela edo Haskell-eko zerrendak bezalakoa dela kontsidera dezakegu.

Turing-en makinen kalkulu-ahalmena ordenagailuen maila berekoa denez, era guztietako kalkuluak burutu ditzakete: batuketak, biderketak eta gainontzeko eragiketa aritmetikoak; datu-baseen erabilera; eta ordenagailuen bidez egin daitezkeen beste kalkulu konplexuak.

Ikusi ditugun automata finituek eta piladun automatek sarrera bezala karaktere-kate bat jaso eta karaktere-kate horrek baldintza zehatz bat betetzen al zuen (hau da, lengoaia zehatz batekoa al zen) erabakitzen dutenez, era horretako kalkuluak burutzen dituzten Turing-en makinak aztertuko ditugu. Beraz, guk kontsideratuko ditugun Turing-en makinen kasuan, makinari karaktere-kate bat pasatuko diogu eta makinak karaktere-kate hori makinari dagokion lengoaiakoa al den aztertuko du. Automata finituekin eta piladun automatekin gertatzen den bezala, Turing-en makina bakoitzak lengoaia bat definitzen du eta makinaren lana, emandako karaktere-kateak lengoaia horretakoak al diren erabak-

itzen saiatzea da.

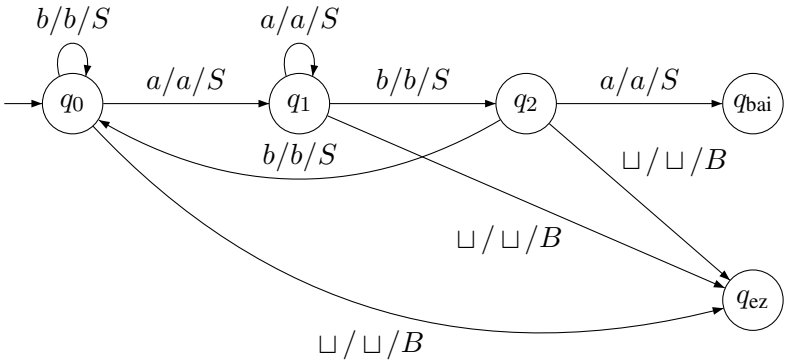
Turing-en makinak automata finituen eta piladun automaten antzekoak dira baina desberdintasun garrantzitsu batzuk ere badituzte:

- Mugarik gabeko memoria eta memoriako edozein posiziotara joateko aukera (lehenago ere aipatu dugu hau).
- Turing-en makinek ez dute zirkulu bikoitzeko egoerarik. Orokorrean bi egoera berezi izango dituzte: q_{bai} eta q_{ez}. Hitz bat karakterez karaktere irakurtzen ari garela q_{bai} egoerara iristen bagara, "Bai" erantzungo da berehala, ez dago hitzaren irakurketa bukatu beharrik edo hitza irakurtzen jarraitu beharrik. Hitz bat karakterez karaktere irakurtzen ari garela q_{ez} egoerara iristen bagara "Ez" erantzungo da berehala, ez dago hitzaren irakurketa bukatu beharrik edo hitza irakurtzen jarraitu beharrik. Bestalde, gerta daiteke hitz batekin inoiz ez iristea ez q_{bai} egoerara eta ez q_{ez} egoerara. Kasu horretan makinak ez du erantzungo eta ondorioz, ziklatzen ari dela esango da. Gainera gerta daiteke q_{bai} edo q_{ez} egoerak ez agertzea. Egoera berezi horietakoren bat ez agertzeak edo biak ez agertzeak makinaren jokabidean eragina du. Adibidez, q_{bai} egoera ez duten makinek ezingo dute inoiz "Bai" erantzun.
- Aurreko puntu horretatik Turing-en makinak "Bai" erantzun dezakeela edo "Ez" erantzun dezakeela edo erantzun gabe gera daitekeela ondoriozta dezakegu.
- Turing-en makina bakoitzak bi alfabeto izango ditu: sarrerako alfabetoa eta irteerako alfabetoa. Datu bezala emango zaizkion hitzak sarrerako alfabetoaren gainean definitutakoak izan behar dute. Irteerako alfabetoan sarrerako alfabetoan dauden sinbolo guztiez gain makinak memorian idazteko bakarrik erabiliko dituen beste sinbolo batzuk ere egongo dira. Irteerako alfabetoan badauden baina sarrerako alfabetoan ez dauden sinbolo horiek ezin dira agertu makinak sarrerako datu bezala jasoko dituen hitzetan.
- Sarrera/Irteerako gailuan hasieran sarrerako datuajasotzen denean, libre geratzen diren beste posizio denetan zuriunea adierazteko erabili ohi den $\sqcup$ sinboloa dagoela suposatu beharko da. $\sqcup$ sinboloa irteerako alfabetoan beti agertuko da baina sarrerako alfabetoan inoiz ez da agertuko.
- Trantsizioek edo geziek $\alpha/\beta/\gamma$ erako hirukoteak dituzte etiketa bezala. Hirukote horietan α osagaiak irakurritako sinboloa adierazten du (automata finituetan bezala), β osagaiak memorian idatzi behar den sinboloa adierazten du eta γ osagaiak Sarrera/Irteerako gailuan (hau da, memorian) zein zentzutan mugitu behar garen adierazten du (e<u>S</u>kuinera, e<u>Z</u>kerrera edo <u>B</u>ertan gelditu).

4.1.2 Turing-en makinen adibideak

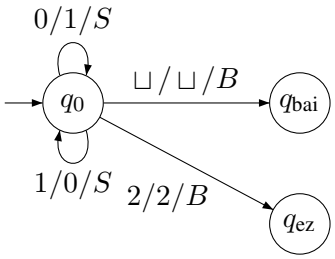
1. Adibide honetan T_1 izeneko Turing-en makina azalduko da. Sarrerako datuen alfabetoa $A_1 = \{a, b\}$ dela eta Irteerako alfabetoa $A_1' = \{a, b, \sqcup\}$ dela jakinda, T_1 makinak "Bai" erantzungo luke $bbbabaaaabaaa$ hitzarentzat eta "Ez" erantzungo luke bba hitzarentzat. Hitz bat irakurtzean $q_{\text{sí}}$ egoerara iristea ez bada lortzen, hau da, q_0, q_1 edo q_2 egoeratakoren batean gelditu bada, q_{ez} egoerara pasa daiteke $\sqcup$ sinboloa irakurriz. Beraz, T_1 makinak ez du inoiz ziklatuko.

T_1



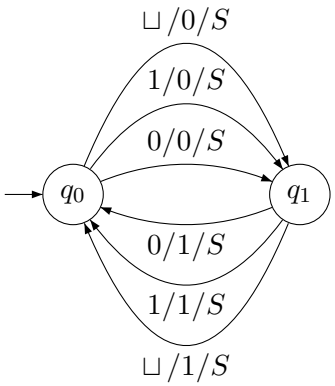
2. Adibide honetan T_2 izeneko Turing-en makina azalduko da. Sarrerako datuen alfabetoa $A_2 = \{0, 1, 2\}$ da eta Irteerako alfabetoa $A'_2 = \{0, 1, 2, \sqcup\}$ da. T_2 makinak "Bai" erantzuten die 2 sinboloa ez duten hitzei eta "Ez" erantzungo die gutxienez 2 sinboloaren agerpen bat dutenei. T_2 makinak inoiz ez du ziklatuko. Adibidez, "Bai" erantzungo luke 1101 hitzarentzat eta "Ez" erantzungo luke 11120200 hitzarentzat. Hitz bat irakurtzean 2 sinboloa ez bada agertzen q_0 egoeran bukatuko da baina gero $\sqcup$ irakurriz q_{bai} egoerara pasatuko da.

T_2



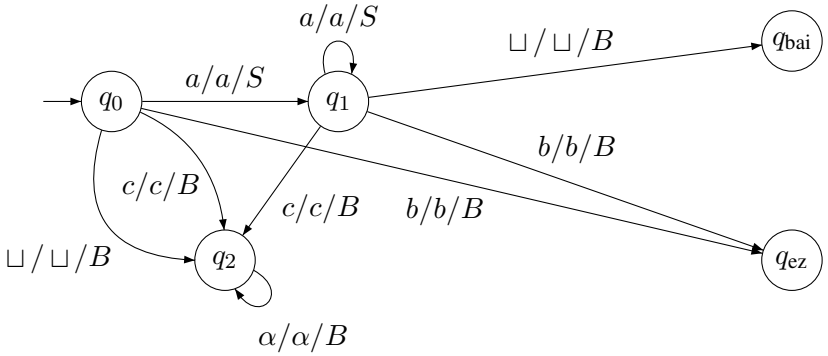
3. Adibide honetan T_3 izeneko Turing-en makina azalduko da. Sarrerako datuen alfabetoa $A_3 = \{0, 1\}$ da eta Irteerako alfabetoa $A'_3 = \{0, 1, \sqcup\}$ da. Makina honetan q_{bai} eta q_{ez} egoerak ez daudenez, makina honek ez du inoiz erantzungo. Beraz, ematen diogun hitza ematen diogula ere, beti ziklatu egingo du, baina hala ere $01010101\ldots$ sekuentzia infinitua idatziz joango da Irteerako gailuan. Hori horrela izanda, adibidez 1101 hitza ematen badiogu edo 111000 hitza ematen badiogu ere $01010101\ldots$ sekuentzia infinitua idatziz joango da Irteerako gailuan.

T_3



4. Adibide honetan T_4 izeneko Turing-en makina azalduko da. Sarrerako datuen alfabetoa $A_4 = \{a, b, c\}$ da eta Irteerako alfabetoa $A'_4 = \{a, b, c, \sqcup\}$ da. T_4 makinak "Bai" erantzuten die hutsak ez dieren eta a sinboloaren errepikapenez eratuta dauden hitzei, "Ez" erantzungo die c-rik ez duten baina gutxienez b bat duten hitzei eta, azkenik, ε hitzarekin eta gutxienez c bat duten hitzekin ziklatu egingo du. Adibidez, "Bai" erantzungo luke $aaaa$ hitzarentzat, "Ez" erantzungo luke $aabaaabb$ hitzarentzat eta ziklatu egingo luke $aabacabcb$ hitzarekin.

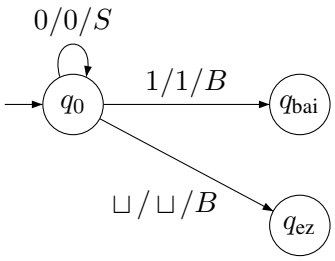
T_4



Hor α elementuak $A'_4 = \{a, b, c, \sqcup\}$ alfabetoko edozein sinbolo ordezkatzen du.

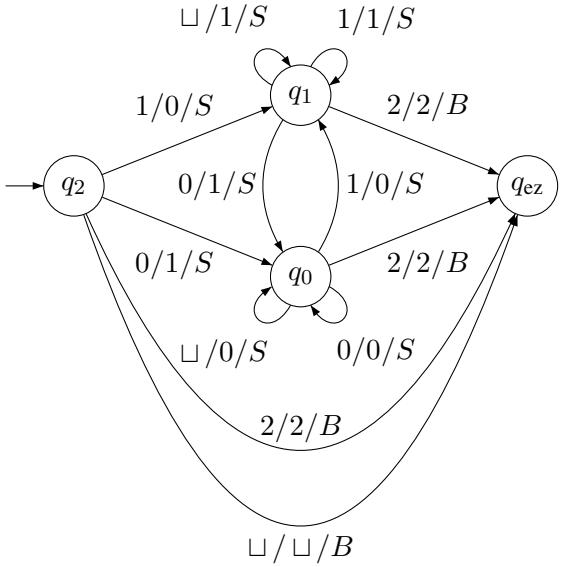
5. Adibide honetan T_5 izeneko Turing-en makina azalduko da. Sarrerako datuen alfabetoa $A_5 = \{0, 1\}$ da eta Irteerako alfabetoa $A'_5 = \{0, 1, \sqcup\}$ da. T_5 makinak "Bai" erantzungo die gutxienez 1 sinboloaren agerpen bat duten hitzei eta "Ez" erantzungo die 1 sinboloaren agerpenik ez duten hitzei. T_5 makinak inoiz ez du ziklatuko. Adibidez, "Bai" erantzungo luke 000110 hitzarentzat eta "Ez" erantzungo luke 000 hitzarentzat. Hitz bat irakurtzean 0 sinboloa ez bada agertzen q_0 egoeran bukatuko da baina gero $\sqcup$ irakurriz q_{ez} egoerara pasatuko da.

T_5



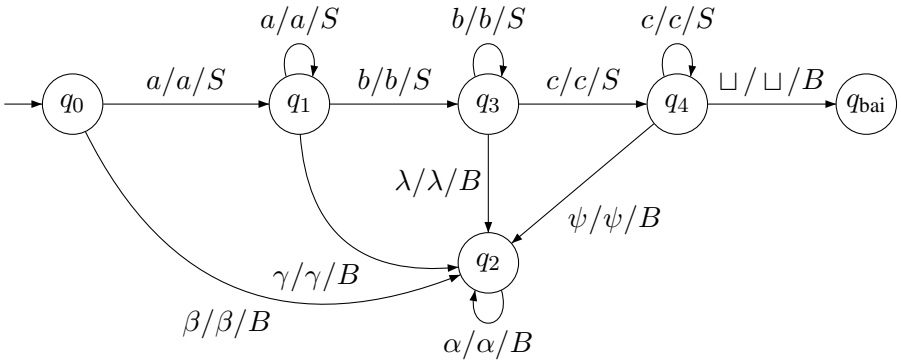
6. Adibide honetan T_6 izeneko Turing-en makina azalduko da. Sarrerako datuen alfabetoa $A_6 = \{0, 1, 2\}$ da eta Irteerako alfabetoa $A'_6 = \{0, 1, 2, \sqcup\}$ da. T_6 makinak "Ez" erantzungo die ε hitzari eta gutxienez 2 sinboloaren agerpen bat duten hitzei eta ziklatu egingo du gainontzeko hitzekin. Ez du inoiz "Bai" erantzungo, izan ere, q_{bai} egoera ez da agertzen T_6 makinan. Adibidez, "Ez" erantzungo luke 1121000 hitzarentzat eta ziklatu egingo luke 11000111 hitzarekin.

T_6



7. Adibide honetan T_7 izeneko Turing-en makina azalduko da. Sarrerako datuen alfabetoa $A_7 = \{a, b, c\}$ da eta Irteerako alfabetoa $A'_7 = \{a, b, c, \sqcup\}$ da. T_7 makinak "Bai" erantzungo die $a^i b^j c^k$ egitura duten hitzei, $i \geq 1$, $j \geq 1$ eta $k \geq 1$ izanda. T_7 makinak ziklatu egingo du gainontzeko hitzekin. Ez du inoiz "Ez" erantzungo, izan ere, q_{ez} egoera ez da agertzen T_7 makinan. Adibidez, "Bai" erantzungo lieke $aabbbcc$ eta $aaabcccc$ hitzei eta ziklatu egingo luke aa, $aabb$ eta $ccabab$ hitzekin.

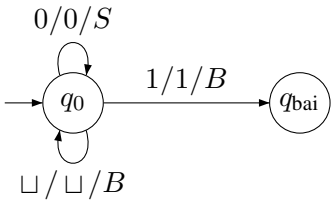
T_7



T_7 makinari dagokion irudian α elementuak $\{a, b, c, \sqcup\}$ multzoko edozein sinbolo adierazten du, β elementuak $\{b, c, \sqcup\}$ multzoko edozein sinbolo adierazten du, γ elementuak $\{c, \sqcup\}$ multzoko edozein sinbolo adierazten du, λ elementuak $\{a, \sqcup\}$ multzoko edozein sinbolo adierazten du eta ψ elementuak $\{a, b\}$ multzoko edozein sinbolo ordezkatzen du.

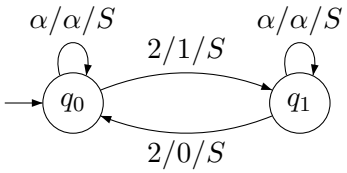
8. Adibide honetan T_8 izeneko Turing-en makina azalduko da. Sarrerako datuen alfabetoa $A_8 = \{0, 1\}$ da eta Irteerako alfabetoa $A'_8 = \{0, 1, \sqcup\}$ da. T_8 makinak "Bai" erantzungo die 1 sinboloa gutxienez behin duten hitzei. T_8 makinak ziklatu egingo du 1 sinboloaren agerpenik ez duten hitzekin. Ez du inoiz "Ez" erantzungo, izan ere, q_{ez} egoera ez da agertzen T_8 makinan. Adibidez, "Bai" erantzungo luke 000110 hitzarentzat eta ziklatu egingo luke 000 hitzarekin.

T_8



9. Adibide honetan T_9 izeneko Turing-en makina azalduko da. Sarrerako datuen alfabetoa $A_9 = \{0, 1, 2\}$ da eta Irteerako alfabetoa $A'_9 = \{0, 1, 2, \sqcup\}$ da. Makina honetan q_{bai} eta q_{ez} egoerak ez daudenez, makina honek ez du inoiz erantzungo. Beraz, ematen diogun hitza ematen diogula ere, beti ziklatu egingo du, baina hala ere 2 sinboloaren agerpen bikoitiak 0 sinboloaz ordezkatuko ditu eta 2 sinboloaren agerpen bakoitiak 1 sinboloaz ordezkatuko ditu. Adibidez, 11000111 hitzean 2 sinboloa agertzen ez denez, ez litzateke ordezkaketarik burutuko, baina 1121200 hitza ematen badiogu, lehenengo 2-a 1 sinboloaz ordezkatuko du eta bigarren 2-a 0 sinboloaz ordezkatuko du, azkenean 1111000 hitza geratuz Irteerako gailuan (memorian).

T_9



T_9 makinari dagokion irdian α elementuak $\{0, 1, \sqcup\}$ multzoko edozein sinbolo ordezkatzen du.

4.2 Lengoaien sailkapena

Hiru makina mota ikusi ditugu:

1. **Automata finituak**. Automata finituen barruan deterministak (AFDak) eta ez deterministak (AFEDak eta ε-AFEDak) bereiztu ditugu, baina badakigu baliokideak direla. Izan ere, AFD baten bidez konputagarria (edo definigarria) den edozein lengoaia AFEDen bidez ere konputagarria da eta AFEDen bidez konputagarria den edozein lengoaia AFDen bidez ere konputagarria da. Automata finituek edozein hitz emanda ere beti erantzuten dute "Bai" edo "Ez" itzuliz. Jarraian automata finituen bidez konputaezinak diren bi lengoaia aipatuko ditugu:
   - $L_1 = \{w \mid w \in A^* \wedge w = a^n b^n \wedge n \geq 0\}$
   - $L_2 = \{w \mid w \in A^* \wedge \exists v(v \in A^* \wedge w = vv^R)\}$
2. **Piladun automatak**. L_1 eta L_2 piladun automaten bidez konputagarriak dira baina piladun automaten bidez konputaezinak diren lengoaiak ere badaude. Piladun automatek edozein hitz emanda ere beti erantzuten dute "Bai" edo "Ez" itzuliz. Jarraian piladun automaten bidez konputaezinak diren bi lengoaia aipatuko ditugu:
   - $L_3 = \{w \mid w \in A^* \wedge w = a^n b^n c^n \wedge n \geq 0\}$
   - $L_4 = \{w \mid w \in A^* \wedge \exists v(v \in A^* \wedge w = vv)\}$
3. **Turing-en makinak**. Turing-en makinen kasuan, Sarrera/Irteerako gailuan idazteko ahalmena edo gaitasuna edukitzeaz gain, honako hiru kasu hauek gerta daitezke: makinak "Bai" erantzutea, makinak "Ez" erantzutea, eta makinak erantzunik ez ematea (ziklatzen ari delako).

AFDa, AFEDa, piladun automata edo Turing-en makina izan daitekeen M makina bat emanda, M makinak konputatutako (edo definitutako) $L(M)$ lengoaian M makinarengandik "Bai" erantzuna jasotzen duten hitzak daude. M makinarengandik "Bai" erantzuna jasotzen ez duten hitzak ez daude $L(M)$ lengoaian. Automata finituen eta piladun automaten kasuan beti "Bai" edo "Ez" erantzuna jasoko da. Turing-en makinen kasuan aldiz, "Bai" edo "Ez" erantzuna jaso daiteke edo erantzunik jaso gabe gelditu gaitezke, makina ziklatzen hasi delako.

$$L(M) = \{w \mid w \in A^* \wedge M\text{-ren erantzuna "Bai" da } w\text{-rentzat}\}$$

Hiru makina mota horiek eta makina batek definitzen duen edo konputatzen duen lengoaiaren kontzeptuak lengoaien honako sailkapen hau eragiten dute 2^{A^*} multzoan:

1. **Lengoaia erregularrak**. Automata finituen bidez defini daitezkeenak (AFDak, AFEDak eta ε-AFEDak). Beste era batera esanda, L lengoaia bat hartzen badugu eta lengoaia hori definitzen duen automata finitu bat existitzen bada, orduan L lengoaia erregularra da.

2. **Kontesturik gabeko lengoaiak**. Piladun automaten bidez defini daitezkeen lengoaiak dira. Beste era batera esanda, L lengoaia bat hartzen badugu eta lengoaia hori definitzen duen piladun automata bat existitzen bada, orduan L kontesturik gabeko lengoaia da.

3. **Lengoaia erabakigarriak**. Edozein hitz emanda, "Bai" edo "Ez" erantzuten duten eta inoiz ziklatzen ez duten Turing-en makinen bidez defini daitezkeen lengoaiak dira hauek. Beraz, lengoaia bat erabakigarria baldin bada, edozein w hitz emanda, hitz hori lengoaiakoa al den ala ez erabakitzen duen T Turing-en makina bat existituko da:

   - $w \in L$ betetzen bada, orduan T makinak "Bai" erantzungo du.
   - $w \notin L$ betetzen bada, orduan T makinak "Ez" erantzungo du.

   Kasu honetan T makinak ez du inoiz ziklatuko, beti erantzungo du ("Bai" edo "Ez").

4. **Lengoaia bereizgarriak**. L lengoaia bat bereizgarria dela esaten da L lengoaiakoak diren hitzentzat "Bai" erantzuten duen T Turing-en makina bat existitzen bada. Lengoaiakoak ez diren hitzentzat "Ez" erantzun dezake T makinak edo erantzun gabe gera daiteke (ziklatzen):

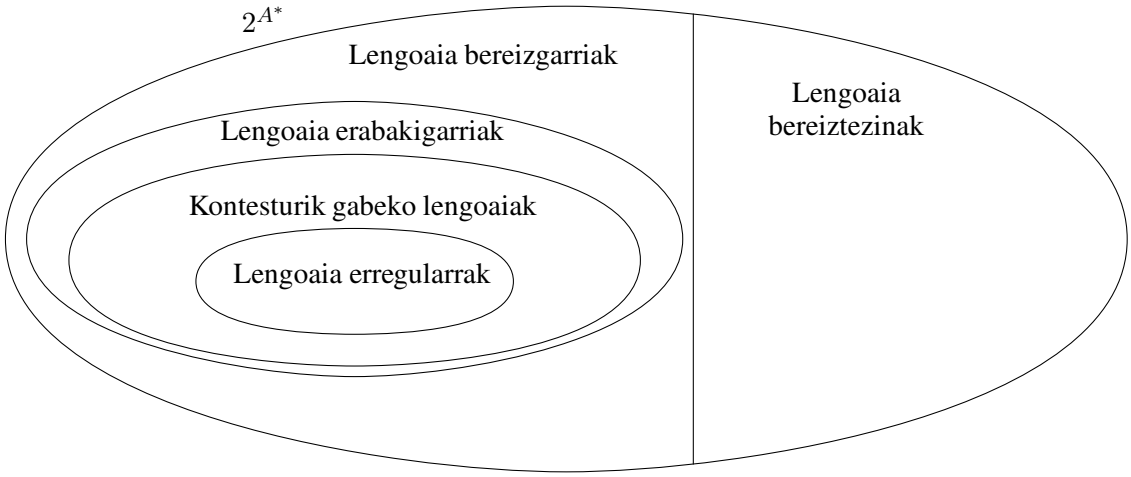
   - $w \in L$ betetzen bada, orduan T makinak "Bai" erantzungo du.
   - $w \notin L$ betetzen bada, orduan T makinak "Ez" erantzungo du edo ziklatu egingo du inoiz erantzun gabe.

   Beraz kasu honetan T makinak zikla dezake, nahiz eta L lengoaiakoak diren hitzentzat inoiz ez ziklatu.

5. **Lengoaia bereiztezinak**. L lengoaia bat bereiztezina dela esaten da L lengoaiakoak diren hitzentzat "Bai" erantzuten duen T Turing-en makinarik ez bada existitzen. Beraz, L bereiztezina baldin bada, edozein T Turing-en makina hartuz:

   - $w \in L$ betetzen bada, T makinak "Bai" erantzun dezake edo ziklatu egin dezake.
   - $w \notin L$ betetzen bada, T makinak "Ez" erantzun dezake edo ziklatu egin dezake.
   - T makinak L-koa den hitzen batentzat ziklatu egiten du.
   - T makinak L-koa ez den hitzen batentzat ziklatu egiten du.

Lengoaia erregularrek kontesturik gabekoen azpimultzoa osatzen dute. Eta kontesturik gabekoek erabakigarrien azpimultzo bat dira. Bukatzeko, lengoaia erabakigarriak bereizgarriak ere badira. Hala ere, lengoaia bat bereizgarria baldin bada baina erabakigarria ez bada, horrek esan nahi du edozein T makina hartuta ere, T makinak L-koa ez den hitzen batentzat ziklatu egiten duela.



Jarraian lengoaia erregular batzuk, kontesturik gabeko lengoaia batzuk eta lengoaia erabakigarri batzuk aipatuko dira:

1. **Lengoaia erregularrak**.
   - $L_1 = \{w \mid w \in A^* \wedge |w|_a = |w|\}$
   - $L_2 = \{w \mid w \in A^* \wedge \exists u, v(u \in A^* \wedge v \in A^* \wedge w = uabav)\}$
2. **Kontesturik gabeko lengoaiak**.
   - $L_3 = \{w \mid w \in A^* \wedge w = a^n b^n \wedge n \geq 0\}$
   - $L_4 = \{w \mid w \in A^* \wedge \exists v(v \in A^* \wedge w = vv^R)\}$
3. **Lengoaia erabakigarriak**.
   - $L_5 = \{w \mid w \in A^* \wedge w = a^n b^n c^n \wedge n \geq 0\}$
   - $L_6 = \{w \mid w \in A^* \wedge \exists v(v \in A^* \wedge w = vv)\}$

4.3 Lengoaia erabakigarriak

Lengoaia bat erabakigarria dela frogatzeko, hitz bat emanda hitz hori lengoaiakoa al den ala ez erabakitzeko balio duen algoritmo bat definitu beharko da. Aurrerago ikusiko dugun bezala, algoritmoak Turing-en makinak erabiliz diseina daitezke. Esate baterako, $L_5 = \{w \mid w \in A^* \wedge w = a^n b^n c^n \wedge n \geq 0\}$ lengoaiari dagokion algoritmoa diseinatzerakoan, a bakoitzeko b bat eta c bat ezabatu beharko dira. Horrela, a bakoitzeko b bat eta c bat ezaba badaitezke eta a sinboloekin bukatzean b-rik eta c-rik ere ez bada geratzen, orduan badakigu a, b eta c kopurua berdinak direla. a sinboloaren agerpenak bukatzean b edo c-ren bat geratzen bada, hitza ez da L_5 lengoaiakoa. Era berean, a sinboloaren agerpenak bukatu baino lehen b-rik eta c-rik gabe geratzen bagara, hitza ez da L_5 lengoaiakoa.

Hitz bat $L_6 = \{w \mid w \in A^* \wedge \exists v(v \in A^* \wedge w = vv)\}$ lengoaiakoa al den erabakitzeko, hasteko hitzaren erdiko sinboloa zein posiziotan dagoen kalkulatu beharko da. Gero, erdiko posizioa zein den kalkulatu ondoren, lehenengo erdiko lehenengo sinboloa eta bigarren erdiko lehenengo sinboloa berdinak al diren aztertu beharko da. Horrela bada, bi sinbolo horiek ezabatu eta lehenengo erdiko bigarren sinboloa eta bigarren erdiko bigarren sinboloa konparatu beharko dira. Sinbolo denak ezabatzea

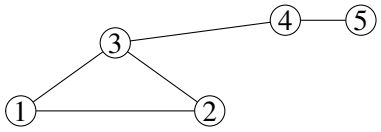
lortzen bada, w hitza vv erakoa dela ziurta dezakegu. Baina erdiko posizioa zein den erabakitzerakoan hitzak luzera bakoitia duela ikusten bada edo sinboloak konparatuz joaterakoan desberdinak diren sinboloak agertzen badira erdi bakoitzeko posizio berean, hitza ez da vv erakoa.

Orain arte lengoaia sinpleak maneiatu ditugu, baina orain norantzarik gabeko grafo konexuak adierazten dituzten hitzez osatutako L_{nggk} lengoaia hartuko dugu.

$$L_{nggk} = \{\langle G\rangle \mid \langle G\rangle \in A^* \wedge G \text{ norantzarik gabeko grafo konexua da } \}$$

Hor $\langle G\rangle$-ren bidez G grafoa kodetzeko erabiltzen den karaktere-katea adierazi nahi da.

Adibidez, G_1 honako grafo hau baldin bada:

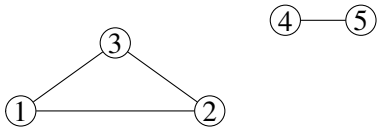


G_1 grafoari dagokion $\langle G_1\rangle$ karaktere-katea $A = \{0, 1, \ldots, 9, (,), , \}$ alfabetoko sinboloz osatuta egongo da. Alfabeto honetan 13 sinbolo ditugu: 0tik 9ra doazen digituak, parentesiak eta koma. $\langle G_1\rangle$ honako hitz hau izango litzateke:

$$(1, 2, 3, 4, 5)(1, 2)(1, 3)(2, 3)(3, 4)(3, 5)$$

Hor $(1, 2, 3, 4, 5)$ zatiak grafoan zein nodo ditugun adierazten du eta gainontzeko bikoteek nodoen arteko arkuak adierazten dituzte.

Horrela, G_2 honako grafo hau baldin bada:

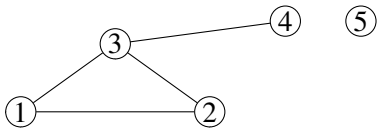


G_2-ri dagokion $\langle G_2\rangle$ karaktere-katea $A = \{0, 1, \ldots, 9, (,), , \}$ alfabetoko sinboloekin osatuta egongo da. $\langle G_2\rangle$ honako hitz hau izango litzateke

$$(1, 2, 3, 4, 5)(1, 2)(1, 3)(2, 3)(4, 5)$$

Hor $(1, 2, 3, 4, 5)$ azpikateak grafoan zein nodo ditugun adierazten du eta gainontzeko bikoteek nodoen arteko arkuak adierazten dituzte.

Adibidez, G_3 honako grafo hau baldin bada:



G_3-ri dagokion $\langle G_3 \rangle$ hitza edo karaktere-katea $A = \{0, 1, \ldots, 9, (,), , \}$ alfabetoko sinboloez osatuta egongo litzateke. $\langle G_3 \rangle$ hitza honako hau izango litzateke

$$(1, 2, 3, 4, 5)(1, 2)(1, 3)(2, 3)(3, 4)$$

Hor $(1, 2, 3, 4, 5)$ azpikateak zein nodo ditugun adierazten du eta gainerako bikoteek nodoen arteko arkuak adierazten dituzte. Adibide honek $(1, 2, 3, 4, 5)$ azpikatearen bidez grafoan zein nodo ditugun adieraztea garrantzitsua dela erakusten du, izan ere, arkuetan bakarrik begiratuz ezin baita jakin beti zein nodo ditugun. Kasu honetan 5 nodoa ez da agertzen bikoteetan, 5 nodoarentzat ez baitago arkurik.

4.4 Turing-en makinen eta sarrerako datuen kodeketa

Aurreko atalean grafoak karaktere-kate bezala adieraz ditzakegula ikusi dugu. Grafoekin egin dugun bezala beste edozein informazio karaktere-kate bezala adieraz daiteke. Turing-en makinak ere karaktere-kate bezala adieraz daitezke. Oinarrizko alfabetoa $\{0, 1\}$ da. Informazio guztia 0 eta 1 sinboloak bakarrik erabiliz adieraz daiteke. Aurreko adibideetan erabili diren alfabetoak gauzak era errazagoan adierazteko erabiltzen dira, baina karaktere-kate horiek 0 eta 1 sinboloak erabiliz kode daitezke. Beraz bai Turing-en makinak eta baita Turing-en makinei pasatzen dizkiegun datuak ere $\{0, 1\}$ alfabetoaren gaineko hitzak bezala ikus ditzakegu. T Turing-en makina bati dagokion karaktere-katea $\langle T \rangle$ bezala adieraziko dugu. Gogoratu edozein programazio-lengoaiatan (Ada, Java, Haskell, eta abar) programa bat idazten dugunean, programa programazio-lengoaia horretan onartzen diren sinboloak erabiliz osatutako karaktere-kate bat dela. D datu bati dagokion karaktere-katea adierazteko $\langle D \rangle$ idatziko dugu.

4.5 Bereizgarriak bai baina erabakigarriak ez diren lengoaiak

Lengoaia bat erabakigarria ez dela frogatzeko, hitz bakoitza lengoaiakoa al den ala ez erabakitzeko gai den algoritmorik (edo Turing-en makinarik) ez dagoela erakutsi behar da. Hau kontraesanaren teknika erabiliz frogatu ohi da. Hiru adibide aztertuko ditugu. Lehenengoan ez da froga formala egingo, oso konplikatua delako. Beste bi kasuetan froga formala emango da.

4.5.1 x aldagaiaren gainean definituta dauden eta erro osoa duten polinomioen lengoaia

x aldagaiaren gainean definituta polinomio batek honako egitura izaten du:

$$a_n x^n + a_{n-1} x^{n-1} + \ldots + a_2 x^2 + a_1 x^1 + a_0 x^0$$

Hor $a_n, a_{n-1}, \ldots, a_2, a_1, a_0$ zenbaki osoak dira (konstanteak).

Adibidez $2x^2 + 2x^1 + 1$ polinomio bat da, $1x^2 + 2x^1 + 1$ beste polinomio bat da, $2x^2 + 5x^1 + 2x^0$ beste polinomio bat da eta $1x^3 + 0x^2 + 0x^1 + (-8)x^0$ beste polinomio bat da.

Polinomio baten erro osoak x aldagaiaren lekuan ipintzean polinomioaren balioa 0 izatea eragiten duten zenbaki osoak dira.

$2x^2 + 2x^1 + 1$ polinomioak ez du erro osorik. $1x^2 + 2x^1 + 1x^0$ polinomioaren kasuan bere erro oso bakarra -1 da. $2x^2 + 5x^1 + 2$ polinomioak ere erro oso bat du: -2. $1x^3 + 0x^2 + 0x^1 + (-8)x^0$ polinomioarentzat bere erro bakarra 2 da. Bukatzeko, $x^2 + (-4)$ polinomioak bi erro oso ditu: 2 eta -2.

$1x^3 + 0x^2 + 0x^1 + (-8)x^0$ polinomioa karaktere-kate bezala honela adieraziko genuke:

$$1xxx + 0xx + 0x + (-8)$$

Karaktere-kate hori $A = \{0, 1, \ldots, 9, +, -, (,)\}$ alfabetoaren gaineko hitz bat da. Bera $1x^3 + 0x^2 + 0x^1 + (-8)x^0$ polinomioa P baldin bada, bere adierazpena $\langle P \rangle$ honako karaktere-kate hau izango da: $1xxx + 0xx + 0x + (-8)$

Orain A alfabetoaren gainean definitutakoak diren eta erro osoa duten polinomioak adierazten dituzten hitzez osatutako lengoia hartuko dugu::

$$L_{eopol} = \{\langle P \rangle \mid \langle P \rangle \in A^* \wedge P \text{ gutxienez erro oso bat duen polinomio bat da } \}$$

w polinomio batek erro osorik ba al duen jakiteko metodoa x aldagaiari balioak emanez joatean datza. Horrela, $x = 0$, $x = 1$, $x = -1$, $x = 2$, $x = -2$, $x = 3$, $x = -3$, eta abarrekin frogatuz joango ginateke polinomioaren balioa 0 izatea eragiten duen x aldagaiaren balio balio aurkitu arte. w polinomioarentzat erro osorik existen bada, lehenago edo geroago balio horretara iritsiko gara eta orduan w polinomioak erro osoa baduela esan ahal izango dugu. Baina w polinomioak erro osorik ez badu, bilaketa infinitua izango da, izan ere x aldagaiari balio desberdinak emanez jarraituko dugu amaierarik gabe noizbait erro osoren bat aurkituko dugulakoan. Beraz polinomio batek erro osorik ez badu, algoritmoak ez du inoiz "Ez" erantzungo, bilatzen eta bilatzen jarraituko baitu. Beraz L_{eopol} lengoaia ez da erabakigarria. L_{eopol} ez den polinomio bat ematen digutenean inoiz ez baitugu jakingo ez dela multzo horretakoa eta hori jakin nahian ziklo infinitu batean sartuko gara. Baina L_{eopol} lengoaiakoa den polinomio bat ematen digutenean bilaketa finitu baten ondoren "Bai" erantzun ahal izango dugu. Horregatik L_{eopol} bereizgarria da baina ez da erabakigarria.

4.5.2 "T makina w hitzarentzat "Bai" erantzuten du" baldintza betetzen duten $\langle T, w \rangle$ erako hitzez osatutako lengoaia

Har dezagun honako lengoaia hau:

$$L_{bai} = \{\langle T, w \rangle \mid \langle T, w \rangle \in A^* \wedge T \text{ Turing-en makinak "Bai" erantzuten du } w\text{-rentzat}\}$$

$\langle T, w \rangle$ karaktere kate bat da eta kate horren bidez T Turing-en makina eta w hitza adierazten dira.

Hasteko L_{bai} lengoaia bereizgarria dela frogatuko dugu. Hori frogatzeko, $\langle T, w \rangle$ karaktere-katea emanda, T makinak w hitzarentzat "Bai" erantzuten badu, "Bai" erantzuten duen $\mathcal{U}$ Turing-en makina bat existitzen dela frogatuko dugu. Ada, Java, Haskell edo beste edozein programazio-lengoaiaz idatzitako programa bat kalkulu zehatz bat egiteko gai den Turing-en makina bat bezala da. Aldi berean ordenagailua beste Turing-en makina bat bezala da eta programa bat exekuta dezake, beraz beste Turing-en makina bat simulatzeko ahalmena du.

$\mathcal{U}$ makinak $\langle T, w \rangle$ egitura duten hitzak jasoko ditu sarrera bezala. $\langle T, w \rangle$ hitzean T Turing-en makina baten deskribapena da eta w hitz bat da. $\mathcal{U}$ makinak jarraian zehazten den algoritmoa jarraituz, L_{bai} lengoaiakoak diren hitzen kasuan "Bai" erantzungo du: $\mathcal{U}$ makinari $\langle T, w \rangle$ erako karaktere-kate bat emandakoan, T-ren exekuzioa simulatuko du T-ri sarrerako datu bezala w emanez. T makinak w hitzarentzat "Bai" erantzuten badu, orduan $\mathcal{U}$ makinak "Bai" erantzungo du $\langle T, w \rangle$ hitzarentzat. T makinak w hitzarentzat "Ez" erantzuten badu, orduan $\mathcal{U}$ makinak "Ez" erantzungo du $\langle T, w \rangle$ hitzarentzat. T makinak w hitzarentzat ziklatu egiten badu (hau da, ez badu ezer erantzuten), orduan $\mathcal{U}$ makinak ere ziklatu egingo du (ez du ezer erantzungo) $\langle T, w \rangle$ hitzarentzat. Beraz L_{bai} lengoaia bereizgarria da $\mathcal{U}$ makinak "Bai" erantzuten duelako justu L_{bai} lengoaiakoak diren $\langle T, w \rangle$ erako hitzentzat.

Orain L_{bai} ez dela erabakigarria frogatuko dugu. Horretarako kontraesanaren teknika erabiliko dugu. Demagun $\langle T, w \rangle$ egitura duen edozein hitzentzat, $\langle T, w \rangle$ hitza L_{bai} lengoaiakoa baldin bada "Bai" eta bestela "Ez" erantzuten duen V makina bat ezistitzen dela. Beraz V makinak inoiz ez du ziklatuko.

- V makina existitzen bada, honako hau egiten duen Y makina eraiki dezakegu:
  - sarrera bezala $\langle T \rangle$ erako hitz bat jasoko du Y makinak.
  - V makinak $\langle T, \langle T \rangle\rangle$ hitzarentzat "Bai" erantzuten badu, Y makinak "Ez" erantzungo du $\langle T \rangle$ hitzarentzat eta V makinak $\langle T, \langle T \rangle\rangle$ hitzarentzat "Ez" erantzuten badu, Y makinak "Bai" erantzungo du $\langle T \rangle$ hitzarentzat.
- Beraz Y makinak $\langle T \rangle$ hitzarentzat V makinak $\langle T, \langle T \rangle\rangle$ hitzarentzat erantzuten duenaren kontrakoa erantzungo du.
- Orain kontraesana sortuko dugu. Y makinari bere deskribapena ematen badiogu sarrera bezala, hau da, $\langle Y \rangle$ ematen badiogu sarrera bezala, zer gertatuko da? Y-ren definizioaren arabera, Y makinak "Ez" erantzungo du $\langle Y \rangle$ hitzarentzat V makinak "Bai" erantzuten badu $\langle Y, \langle Y \rangle\rangle$ hitzarentzat eta Y makinak "Bai" erantzungo du $\langle Y \rangle$ hitzarentzat V makinak $\langle Y, \langle Y \rangle\rangle$ hitzarentzat "Ez" erantzuten badu.
- Horrek esan nahi du Y makinak $\langle Y \rangle$ hitzarentzat "Ez" erantzungo duela Y makinak $\langle Y \rangle$ hitzarentzat "Bai" erantzuten badu eta Y makinak $\langle Y \rangle$ hitzarentzat "Bai" erantzungo duela Y makinak $\langle Y \rangle$ hitzarentzat "Ez" erantzuten badu.
- Hori kontraesana da, Y makinak ezin baitu "Bai" eta "Ez" erantzun hitz berarentzat ($\langle Y \rangle$ hitzarentzat).
- L_{bai} lengoaia erabakigarria suposatzeak kontraesan batera eraman gaituenez, L_{bai} ez dela erabakigarria ondoriozta dezakegu.

4.5.3 "T makina w hitzarentzat gelditu egiten da ("Bai" edo "Ez" erantzunez)" baldintza betetzen duten $\langle T, w \rangle$ erako hitzez osatutako lengoaia

Lengoaia hau **gelditzearen problema** (*halt problem* ingelesez) bezala ezagutzen da.

Kontsideratuko dugun lengoaia honako hau da beraz:

$$L_{halt} = \{\langle T, w \rangle \mid \langle T, w \rangle \in A^* \wedge \\ T \text{ Turing-en makina gelditu egiten da } w \text{ hitza ematen zaionean, "Bai" edo "Ez" erantzunez } \}$$

$\langle T, w \rangle$ espresioa T Turing-en makina eta w hitza kodetzen dituen karaktere-kate bat da.

Hasteko L_{halt} lengoaia **bereizgarria** dela frogatuko dugu, hau da, $\langle T, w \rangle$ erako hitz bat emandakoan honako jokaera duen R izeneko Turing-en makina bat existitzen dela frogatuko dugu:

- T makinari w hitza emandakoan T makinak "Bai" edo "Ez" erantzuten badu (hau da, T makinak w hitza ematen zaionean ez badu ziklatzen), orduan R makinak "Bai" erantzungo du $\langle T, w \rangle$ hitzarentzat.
- T makinari w hitza emandakoan T makinak ez badu ezer erantzuten (hau da, T makinak w hitza ematen zaionean ziklatu egiten badu), R makinak ez du ezer erantzungo (ziklatu egingo du) $\langle T, w \rangle$ hitzarentzat.

R izeneko Turing-en makinak L_{halt} lengoaia bereizteko honako urratsak jarraituko ditu:

1. R makinak T makina exekutatuko du T makinari sarrera bezala w hitza emanez.

2. T makinak "Bai" edo "Ez" erantzuten badu w hitza ematen zaionean, orduan R makinak "Bai" erantzungo du $\langle T, w\rangle$ hitzarentzat.

3. T makinak ziklatu egiten badu w hitza ematen zaionean, orduan R makinak ere ziklatu egingo du $\langle T, w\rangle$ hitzarentzat.

Beraz L_{halt} lengoaia bereizgarria da R erabiliz, izan ere justu L_{halt} lengoaiakoak diren $\langle T, w\rangle$ hitzentzat R makinak "Bai" erantzuten du.

Orain L_{halt} **erabakigarria ez dela** frogatuko dugu. Horretarako kontraesanaren teknika erabiliko dugu. Demagun L_{halt} erabakigarria dela eta beraz $\langle T, w\rangle$ erako hitz bat emandakoan honako jokaera duen H makina bat existitzen dela: $\langle T, w\rangle$ hitza L_{halt} lengoaiakoa baldin bada, H makinak "Bai" erantzungo du eta $\langle T, w\rangle$ hitza L_{halt} lengoaiakoa ez bada, H makinak "Ez" erantzungo du. Beste era batera esanda:

- T makinari w hitza emandakoan T makinak "Bai" edo "Ez" erantzuten badu, H makinak "Bai" erantzungo du $\langle T, w\rangle$ hitzarentzat (hau da, T makinak ez badu ziklatzen w hitzarentzat, H makinak "Bai" erantzungo du $\langle T, w\rangle$ hitzarentzat).

- T makinari w hitza emandakoan T makinak ziklatu egiten badu w hitzarentzat, H makinak "Ez" erantzungo du $\langle T, w\rangle$ hitzarentzat.

Definizioz, H makinak ez du inoiz ziklatzen, beti "Bai" edo "Ez" erantzungo du.

Jarraian kontraesana sortuko dugu, izan ere, H makina hori existitzen bada, $\langle T, w\rangle$ erako edozein hitz L_{bai} lengoaiakoa al den ala ez erabakitzeko gai den G makina defini dezakegu. G makinari $\langle T, w\rangle$ erako hitz bat emandakoan G makinak honako urratsak jarraituko lituzke $\langle T, w\rangle$ hitza L_{bai} lengoaiakoa al den ala ez erabakitzeko:

1. G makinak H makina exekutatu edo simulatuko du H-ri sarrera bezala $\langle T, w\rangle$ emanez.

2. H makinak "Bai" erantzuten badu $\langle T, w\rangle$ hitzarentzat, horrek esan nahi du T makinari w hitza ematen badiogu T gelditu egingo dela eta "Bai" edo "Ez" erantzungo duela. Horregatik, H makinak "Bai" erantzuten badu $\langle T, w\rangle$ hitzarentzat, jarraian G makinak T makina simulatu edo exekutatuko du datu bezala w hitza emanez. T makinak "Bai" erantzuten badu w hitzarentzat, orduan G makinak ere "Bai" erantzungo du $\langle T, w\rangle$ hitzarentzat. Baina T makinak "Ez" erantzuten badu w hitzarentzat, orduan G makinak "Ez" erantzungo du $\langle T, w\rangle$ hitzarentzat.

3. Bestalde, H makinak "Ez" erantzuten badu $\langle T, w\rangle$ hitzarentzat, horrek esan nahi du T makinak ziklatu egiten duela w hitzarentzat eta beraz T makinak ez duela "Bai" erantzuten w hitzarentzat. Kasu honetan G makinak zuzenean "Ez" erantzungo luke $\langle T, w\rangle$ hitzarentzat (T makina exekutatu gabe).

Beraz $\langle T, w\rangle$ erako edozein hitz emanda, G makinak "Bai" edo "Ez" erantzungo luke eta ondorioz L_{bai} erabakigarria izango litzateke. Baina hau kontraesana da, izan ere badakigu L_{bai} ez dela erabakigarria.

L_{halt} erabakigarria dela suposatzeak kontraesanera garamatzanez, L_{halt} ez dela erabakigarria ondoriozta dezakegu.

4.6 Lengoaia bereiztezinak

Atal honetako lehenengo azpiatalean lengoaia bereiztezinak badaudela frogatuko dugu era orokorrean, inolako adibide zehatzik eman gabe. Bigarren azpiatalean bereiztezina den lengoaia bat aurkeztuko da eta lengoaia hori bereiztezina dela frogatuko da era formalean.

4.6.1 Lengoaia bereiztezinak badaude

Badakigu $A = \{0, 1\}$ alfabetoa hartzen badugu, Turing-en makina bakoitza A^* multzoko hitz baten bidez kodetu daitekeela. L_{TM} lengoaia Turing-en makinak adierazten dituzten A^* multzoko hitzez osatutako lengoaia dela kontsideratuko dugu:

$$L_{TM} = \{w \mid w \in A^* \wedge w \text{ Turing-en makinaren bat adierazten duen karaktere-katea da } \}$$

A^* zenbagarria denez eta L_{TM} lengoaia A^* multzoaren azpimultzoa denez, L_{TM} multzoa ere zenbagarria da. L_{TM} zenbagarria denez eta 2^{A^*} (lengoaia denen multzoa) zenbaezina denez, L_{TM} multzoan 2^{A^*} multzoan baino elementu gutxiago daude eta ondorioz makina-kopurua lengoaia-kopurua baino txikiagoa da. Gainera Turing-en makina bakoitzak lengoaia bat bereiz dezake. Horrek esan nahi du lengoaia batzuentzat ez dagoela makinarik eta ondorioz lengoaia horiek bereiztezinak dira derrigorrez, beraiek bereizten dituen makinarik ez baita existitzen.

4.6.2 Bereiztezina den lengoaia bat

L_{bai} lengoaiaren osagarria hartuko dugu:

$$\overline{L_{bai}}$$

Kontraesanaren teknika erabiliz, $\overline{L_{bai}}$ bereiztezina dela frogatuko dugu. Horretarako $\overline{L_{bai}}$ bereizgarria dela suposatuko dugu. Beraz justu $\overline{L_{bai}}$ lengoaiakoak diren $\langle T, w\rangle$ erako hitzentzat "Bai" erantzuten duen J makina bat existitzen dela suposatuko dugu. Ondorioz, $\langle T, w\rangle$ hitza $\overline{L_{bai}}$ lengoaiako bada J makinak "Bai" erantzungo du eta $\langle T, w\rangle$ hitza $\overline{L_{bai}}$ lengoaiako ez bada J makinak "Ez" erantzun dezake edo ziklatu egin dezake.

Bestalde badakigu L_{bai} lengoaia bereizgarria dela. Beraz justu L_{bai} lengoaiakoak diren $\langle T, w\rangle$ erako hitzentzat "Bai" erantzuten duen I makina bat existitzen dela badakigu.

Orain kontraesana sortuko dugu J eta I makinak erabiliz. Izan ere J eta I erabiliz L_{bai} lengoaia erabakitzen duen K makina bat eraiki baitezakegu. K makinari sarrerako datu bezala $\langle T, w\rangle$ erako hitz bat ematen zaionean, K makinak honako urrats hauek jarraituko ditu:

1. K makinak I eta J makinak simulatu edo exekutatuko lituzke aldi berean (paraleloan) $\langle T, w\rangle$ hitzarentzat.
2. $\langle T, w\rangle$ hitzak derrigorrez L_{bai} lengoaiakoa edo $\overline{L_{bai}}$ lengoaiakoa izan behar duenez, I makinak edo J makinak "Bai" erantzungo du.
3. I makinak "Bai" erantzuten badu $\langle T, w\rangle$ hitzarentzat, horrek $\langle T, w\rangle$ hitza L_{bai} lengoaiakoa dela esan nahiko luke eta K makinak "Bai" erantzungo du.
4. J makinak "Bai" erantzuten badu $\langle T, w\rangle$ hitzarentzat, horrek $\langle T, w\rangle$ hitza L_{bai} lengoaiakoa ez dela esan nahiko luke eta K makinak "Ez" erantzungo du.

Beraz $\langle T, w\rangle$ erako edozein hitz izanda ere, K makinak "Bai" erantzungo du $\langle T, w\rangle$ hitza L_{bai} lengoaiako baldin bada eta K makinak "Ez" erantzungo du $\langle T, w\rangle$ hitza L_{bai} lengoaiako ez bada. Beraz L_{bai} lengoaia erabakigarria izango litzateke K makinari esker. Baina hau kontresana da, L_{bai} ez dela erabakigarria frogatuta baitaukagu.

$\overline{L_{bai}}$ bereizgarria dela suposatzeak kontraesanera garamatzanez, $\overline{L_{bai}}$ erabakigarria ez dela ondoriozta dezakegu.