KONPUTAEZINTASUN FROGAPEN BATZUK DIAGONALIZAZIO TEKNIKA ERABILIZ

J. Ibáñez

A. Irastorza

A. Sánchez

Aurkibidea

Diagonalizazio teknika

Arlo informatikoa oso azkar garatu da eta garatzen ari da. Hala ere, bere oinarri matematikoak, Informatika Teorikoa delakoaren gunea, 30. hamarkadan finkatuta geratu ziren. Horrela, teknologikoki elektronika digitalean oinarritutako lehenengo ordenadoreak azaldu zirenean (40. hamarkadaren bukaera aldera), jadanik Konputagarritasunaren Teoriaren oinarrizko hipotesiak ezarrita zeuden, ziztu biziko aldaketak eragin dituzten urte askok aldatzerik lortu ez duten hipotesiak. Jadanik garai hartan Alan Mathison Turing-ek frogatu zuen, ahalik eta potentzia handienekoa imajinatuta ere, inolako ordenadorek ebatzi ezingo zituen zenbait gai edo arazo bazeudela. Balizko algoritmorik ez duten problema horiek, *konputaezinak* deitzen ditugunak, ez dira salbuespenak edo zerbait patologikoa, ukigarriak dira eta adibide ugari aurki dezakegu. Programen portaeraren inguruan planteatzen diren problemen artean, asko dira konputaezinak. Adibidez, bi programa emanda baliokideak ote diren, hots, hasierako baldintza berdinekin lan egiterakoan emaitza bera sortzen ote duten esaten duen algoritmorik ez dago. Programa baten egikaritzapena agindu zehatz batetik pasako denentz aurrez esateko algoritmorik ere ez dago. Familia horretako, zalantzarik gabe, kide ezagunena, geratze problema, aurretik aipatutakoen eta beste problema askoren konputaezintasunaren "arduradun" nagusia da: sarrerako datu zehatz batzuk hartzerakoan programa bat begizta infinituan geratuko ote den era orokorrean erabakitzeko algoritmorik ez dago. Problema konputaezinen lehenengo adibidea da eta horrela, Konputagarritasunaren Teoriari buruzko edozein liburutan [Har 87, MAK 88, SW 88] derrigorrezko kapitulua eskaintzen zaio.

Problema baten konputagarritasuna frogatu nahi denean hau ebatziko duen algoritmo zehatz bat aurkitzea nahikoa den bitartean, bere konputaezintasuna frogatzeko lanetan hasten garenean, horrelako algoritmorik ez dagoela, hots, existitzen diren algoritmoak problema hori ebazteko gai izango ez direla egiaztatuko duen argumentu logikoa behar dugu. Izaera unibertsaleko argumentu hori ezartzea ez da batere erraza izaten, eta normalean, absurduraino eramandako frogapen batekin erlazionatuta egon ohi da. Helburu hori lortzeko zenbait teknika daude, horien artean *diagonalizazioarena*, *laburgarritasunarena* eta *puntu finkoarena* garrantzitsuenak direlarik.

Horietatik oinarrizkoena diagonalizazioaren teknika da, nahiko ezaguna, ez baita Informatika Teorikoaren tresna espezifikoa. Georg Ferdinand Ludwig Philipp Cantor-ek asmatu zuen frogapen formal mota hau lehendabiziko aldiz 1873. urtean aplikatu zen, Multzoen Teoriaren arloan eta zenbaki arrunten multzoaren kardinaltasuna gainditzen duten, hots, zenbagarriak ez diren multzoak badaudela frogatzeko. Teknika honek kontraesanaren bidezko frogapena eta bitarteko eraiketa-prozesu berezi bat [Sta 81] konbinatzen ditu. Objektu-klase infinitua eta zenbagarria

{O_1, O_2, O_3, ...} eduki behar da, eta objektu bakoitza independenteak diren atributuen zerrenda infinitu eta zenbagarri baten bitartez defini daiteke $O_i = \{a_{i1}, a_{i2}, a_{i3}, ...\}$. Kontraesana O′ objektu berria eraikiz bilatzen da. Objektu hori, bere izaeragatik, esandako klase horretakoa izan beharko litzateke, baina aldi berean klaseko O_i kide bakoitzaren desberdina, bere *i*-garren atributuan, a_{ii}, hain zuzen ere. Absurdua, O′ *objektu diagonalak* klaseko kidea izateagatik eta ez izateagatik heredatzen dituen ezaugarri kontraesankorretatik sortzen da.

Dokumentu honetan diagonalizazio teknika erabiltzerakoan objektuak funtzio konputagarriak izango dira, eta funtzio bat definitzen duten atributuak, balizko datuei funtzio horrek esleitzen dizkien emaitzak izango dira. Gure helburua, jadanik baditugun funtzio guztien desberdina izango den *funtzio konputagarri diagonala* eraikitzea izango da: funtzio berri hori *i*-garren funtzio konputagarriaren desberdina izango da, *i*-garren datuarentzat posible diren emaitzen artean, desberdina den bat emango duelako.

Gure kasuan, programen gödelizazioaren fenomenoa dela kausa, teknikak autoerreferentziarekin ere badu erlaziorik. Funtzio diagonala konputagarrien klasean egon behar dela eta ondorioz, funtzioen zerrendan *e* leku bat dagokiola egiaztatu ondoren, *e*-garren datuaren gainean duen portaera jakiten saiatzen garenean aurkitu ohi da absurdua. Funtzioak, hauek konputatzen dituzten programen arabera ordenatzen direnez, *e*-garren funtzioa *e* lekua betetzen duen programak konputatzen duena da. Bestaldetik, programak adierazpide-eskema baten arabera ordenatuak izan dira; eskema horrek programak eta datuak biunibokoki erlazionatzea ahalbidetzen du, eta hortaz, *e*-garren programa *e*-garren datuaren bidez adierazten da. Ondorioa da, labur esanda, absurdua *e*-garren lekuan dagoen programa bere buruaren gainean aplikatzerakoan sortzen dela. Eskema autoerreferentzial honengatik sortzen da formulazio diagonal askoren eta paradoxa Errusselliarren arteko paralelismoa.

Hala ere dokumentu honetan ez gara diagonalizazio teknika bera azaltzen saiatuko, ezaguntzat jotzen baitugu. Gure helburua, zailtasun-maila desberdineko adibide bilduma baten bitartez, teknika hori argitzea da. Zailtasuna ulertzeko modua subjektiboa dela badakigu, baina ikasketa pixkanakakoa izan dadin adibideak ordenatzen saiatu gara. Hala eta guztiz ere, adibide bakoitzaren edukiak ez du zerikusirik besteekin, hortaz, edozein ordenatan kontsulta litezke.

Oinarrizko kontzeptuak eta notazioa

Programen portaera sarrera/irteeraren arabera deskribatu ahal izateko, sarrera batzuentzat indefinituta egon daitezkeen funtzio partzialak erabili behar ditugu. ψ funtzioak x-ren gainean aplikatzerakoan konbergitzen badu, hots irudia badu, $\psi(x)\downarrow$ ikurraren bidez adierazten dugu. Funtzioa puntu horretan indefinituta badago, berriz, $\psi(x)\uparrow$ bezala adierazten dugu. Funtzioek, oro har, edozein argumentu-kopuru eduki dezakete, baina diagonalizazioaren teknika aplikatzeko sarrera bakarreko funtzioetara laburtzea komeni da, funtzio diagonalaren eraikitzea erraztearren. Horregatik dokumentu honetan sartzen ditugun adibide guztiak mota horretakoak dira.

Funtzio bat *konputagarria* da, hau kalkulatuko duen programaren bat baldin badago, programazio-lengoaiaren batean idatzitako programa. Funtzioen konputagarritasuna frogatzeko dokumentu honetan erabilitako lengoaia while programen lengoaia da. Honen gainean, programak laburtu eta bere esanahia hobeto ulertu ahal izateko, makrolengoaia bat definitu da [IIS 98]. Hala ere, konputagarritasuna frogatzeko erabilitako edozein programazio-sistemak funtzio konputagarrien multzo bera definitzen duela jakina da. While programen lengoaian lanerako erabiltzen diren datuak hitzarmenezko $\sum$ alfabeto baten gaineko karaktere-kateak edo hitzak dira. Haatik, balizko beste datu-multzoen eta $\sum^*$-ko elementuen arteko egokitzapen errazak defini daitezke, eta horrela, katea bakoitzak multzo berriaren elementu bat adieraziko du eta hitzen gaineko konputazioa mota berriaren elementuen adierazpidearekin bat etorriko da. Era horretan inplementatutako datu-mota batzuk dira boolearrena ¡, zenbaki arruntena Õ, eta while programa beraiena ere ÷. Lehen kasuan ezik begi-bistakoak diren arrazoiengatik, inplementazioak beti bijektiboak egiten dira.

Behin zenbaki arruntak inplementatu ondoren balizko datuen multzoa zenbat dezakegu, $\sum^* = \{w_0, w_1, w_2, w_3, \ldots\}$, non w_i i zenbakia adierazteko erabiltzen den hitza den. Era berean, while programen multzoa ere zenbat dezakegu, hots, ÷ $= \{P_0, P_1, P_2, P_3, \ldots\}$, non P_i w_i hitzak adierazten duen programa den. Orduan, w_i hitza (eta, hedapenez i) P_i programaren *kodea* dela esaten da.

Zenbaketa honetan balizko programa guztiak daudenez, argumentu bakarreko funtzio konputagarri guztien klasea ere zerrenda batean zenbat dezakegu $\{\varphi_0, \varphi_1, \varphi_2, \varphi_3, \ldots\}$, non φ_i funtzioa P_i programak argumentu bakarra ematen zaionean konputatzen duen funtzioa den. Orduan, w_i hitza (eta, hedapenez i) φ_i funtzioaren *indizea* dela esaten da. Gainera φ_i funtzioaren eremua eta heina, W_i eta R_i deituko ditugu, hurrenez hurren.

Hori guztia dela eta, ψ funtzioa konputagarria dela baieztatzea, $\psi \cong \varphi_e$ betetzen duen *e* indizea existitzen dela baiestea da, hots, aurretik aipatu zerrendaketa horretan ψ funtzioak leku bat eduki behar duela esatea.

Funtzio konputagarrien multzoan badago bat, dokumentu honetan maiz erabiliko delako, arreta berezia merezi duena: *funtzio unibertsala*. Funtzio hau while programen interpretatzaile edo itzultzailea da, eta beraz, edozein sarreraren gainean edozein funtzio konputagarriren portaera simulatzeko gai da. Funtzio unibertsala Φ hizkiaren bidez adierazten da eta honako definizio formala du:

$$\Phi(x, y) \cong \begin{cases} \varphi_x(y) & \varphi_x(y)\downarrow \\ \perp & \text{bestela} \end{cases}$$

Funtzioak konputagarri eta konputaezinen artean sailkatzen ditugun bezala, multzoen artean ere bi klase bereiz ditzakegu: multzo errekurtsibo edo erabakigarrien klasea, $\sum_0$, eta multzo errekurtsiboki zenbagarri edo sasierabakigarrien klasea, $\sum_1$. Edozein A multzok bi funtzio atxikirik ditu: bere ezaugarri funtzioa, C_A eta bere funtzio erdiezaugarria χ_A. Bi funtzio horiek honela definitzen dira:

$$C_A(x) = \begin{cases} \text{true} & x \in A \\ \text{false} & x \notin A \end{cases}$$

$$\chi_A(x) \cong \begin{cases} \text{true} & x \in A \\ \perp & x \notin A \end{cases}$$

A multzoa *erabakigarria* dela esaten dugu bere ezaugarri funtzioa konputagarria bada, eta *sasierabakigarria* dela diogu bere funtzio erdiezaugarria konputagarria bada.

Garrantzitsua da azpimarratzea sasierabakigarria den A multzo ez-huts baten elementuak f funtzio oso eta konputagarri batek defini ditzakeela, multzo hori f funtzioaren heinaren berdina izango delarik. Ezaugarri hori diagonalizazioaren teknika aplikatzeko erabil daiteke, era horretan A-ren elementuak zerrendaketa honen bitartez lor daitezkeelako, $A = \text{ran}(f) = \{f(0), f(1), f(2), f(3), \ldots\}$.

$$\psi(x) = \begin{cases} x+1 & \varphi_x(x)+1>3 \\ 0 & \textbf{bestela} \end{cases}$$ **funtzioa ez da konputagarria**

Demagun ψ funtzioa konputagarria dela, orduan **x** kodea duen edozein programarentzat $\varphi_x(\mathbf{x})>2$ denentz jakin dezakegu, $\psi(\mathbf{x})>0$ den begiratzea besterik egin gabe. Horretan oinarrituta funtzio konputagarri guztien desberdina izango den ρ funtzio diagonal eta konputagarria defini dezakegu. $\rho(\mathbf{x})$ zer izango den ezartzeko, **x**-ren balio bakoitzeko honakoa kontuan hartuko dugu:

- Baldin $\psi(\mathbf{x})>0$ orduan badakigu $\varphi_x(\mathbf{x})+1>3$ betetzen dela. Hortaz ρ eta φ_x funtzioak **x** puntuan desberdinduarazi ditzakegu, horretarako $\rho(\mathbf{x})$-rentzat 2 baino handiagoa ez den balio bat definituz, esaterako 0 (2 edo txikiagoa den edozein balio aukeratzea ere posible zen, baita indefinitutako balioa uztea ere).

- Baldin $\psi(\mathbf{x})=0$ orduan badakigu $\neg(\varphi_x(\mathbf{x})+1>3)$ betetzen dela, hots, $\varphi_x(\mathbf{x})$ 2 dela, edo balio txikiago bat, edo bestela dibergitzen duela. Hortaz, $\rho(\mathbf{x})$-ri 2 baino handiagoa den balio bat, esaterako 5, emanez, ρ eta φ_x funtzioak **x** puntuan desberdinak izatea lor dezakegu.

		$\Psi(0)=0$	$\Psi(1)>0$	$\Psi(2)=0$	$\Psi(3)>0$	
x	ρ	φ_0	φ_1	φ_2	φ_3	
0	5	$\neg(\varphi_0(0)>2)$	?	?	?	
1	0	?	$\varphi_1(1)>2$	?	?	
2	5	?	?	$\neg(\varphi_2(2)>2)$	?	...
3	0	?	?	?	$\varphi_3(3)>2$	
		?	?	?	?	

...

Eskema adibide-taulan erakusten dugu eta ikusten denez sortu berri dugun funtzioa eta beste funtzioak bereizten diren puntuak taularen diagonalean daude. Goiko lerroan, **x** indize bakoitzaren gainean aplikatzerakoan ψ funtzioak itzuli litzakeen emaitzak azaltzen dira. Taulan ilunduta dauden gelaxketan, hots, taulako diagonaleko gelaxketan, bertako balioen inguruan lortzen den informazioa ikusten

dugu. Azkenik, ezkerreko zutabean, zerrendako funtzio guzti-guztietatik bereizteko, ρ funtzioak hartuko lituzkeen balioak adierazten dira.

Azaldu berri dugun eskema jarraituz, frogapena osatuko dugu. Demagun ψ konputagarria dela. Orduan honako funtzioa definitzen dugu:

$$\rho(x) = \begin{cases} 5 & \psi(x) = 0 \\ 0 & \psi(x) > 0 \end{cases}$$

eta konputagarria da, honako programak frogatzen duen legez:

if ψ(X1) = 0 **then** X0 := 5; **else** X0 := 0; **end if**;

Programa honek, bere makroak hedatu eta lortutako while programa kodetu ondoren, **e** kodea edukiko du, hots, $\rho = \varphi_e$. Funtzioaren definizioarengatik, honakoa betetzen dela badakigu: $\forall y(\rho(y)=0 \vee \rho(y)=5)$. Baina funtzio hori bere indizearen gainean aplikatzearen emaitza zein den egiaztatzen saiatzen bagara, absurdura helduko gara.

$$\rho(\mathbf{e})=0 \overset{\rho=\varphi_e}{\Rightarrow} \varphi_{\mathbf{e}}(\mathbf{e})=0 \Rightarrow \neg(\varphi_{\mathbf{e}}(\mathbf{e})>2) \overset{\psi \text{ def.}}{\Rightarrow} \psi(\mathbf{e})=0 \overset{\rho \text{ def.}}{\Rightarrow} \rho(\mathbf{e})=5 \Rightarrow \neg\rho(\mathbf{e})=0$$

$$\neg\rho(\mathbf{e})=0 \overset{\rho \text{ def.}}{\Rightarrow} \rho(\mathbf{e})=5 \overset{\rho=\varphi_e}{\Rightarrow} \varphi_{\mathbf{e}}(\mathbf{e})=5 \Rightarrow \varphi_{\mathbf{e}}(\mathbf{e})>2 \overset{\psi \text{ def.}}{\Rightarrow} \psi(\mathbf{e})=\mathbf{e}+1 > 0 \overset{\rho \text{ def.}}{\Rightarrow} \rho(\mathbf{e})=0$$

$\rho(\mathbf{e})$-k ez 0 ezta 5 ere, ezin duela hartu frogatu dugu, beraz absurdura heldu gara. Zentzugabekeria hori hasieran egin dugun suposaketatik, hau da, ψ konputagarria zela esaten genuenetik etorri da, hortaz, funtzio hori konputaezina dela frogatuta geratzen da.

$A = \{ x: \forall y\ (y \leq x \rightarrow \varphi_x(y) \neq y) \}$ ez da erabakigarria

A multzoa erabakigarria dela suposatzen dugu, eta beraz, bere ezaugarri funtzioa C_A konputagarria izanik, **x** kodea duen programa emanda, hau A multzokoa denentz jakin dezakegula. Hipotesi horretan oinarriturik, eta diagonalizazioaren bidez, funtzio konputagarri guztien desberdina izango den ψ funtzio konputagarria defini dezakegu. ψ hau eta φ_x funtzio konputagarri bakoitza gutxienez puntu batean (ahal bada, diagonaleko **x** sarreran) desberdintzen saiatuko gara.

- **x** hitza A-n badago (eta hori badakigu $C_A(\mathbf{x})$=true delako), **x**-ren berdina edo txikiagoak diren **y** hitz guztientzat φ_x funtzioak konbergitzen duela eta **y**-ren desberdina den balio bat itzultzen duela jakingo dugu. Hau da, **x**-rentzat, zehazki, $\varphi_x(\mathbf{x}) \neq \mathbf{x}$ dela. Orduan $\psi(\mathbf{x})$ era askotara definitzeko aukera dugu, adibidez, **x** edo $\perp$, eta horrela $\neg\psi(\mathbf{x}) \cong \varphi_x(\mathbf{x})$ edukiko genuke. Gauza bera lortuko genuke $\varphi_x(\mathbf{x})+100$, $\varphi_x(\mathbf{x})+1$ edo $\varphi_x(\mathbf{x})*2+2$ hartuta ere, $\varphi_x(\mathbf{x})$-k konbergitzen duela badakigulako.

- **x** hitza A-n ez badago (eta hori badakigu $C_A(\mathbf{x})$=false delako), **x**-ren berdina edo txikiagoak diren **y** hitzetako batentzat φ_x funtzioak dibergitzen duela edo **y**-ren berdina den balio bat itzultzen duela jakingo dugu. Baina ez dakigu zehazki zein hitzentzat gertatzen den ezta beste hitzen gainean zer egiten duen ere. Hortaz, **x** hitzaren gainean φ_x-k duen portaerari buruzko informaziorik ez dugu, eta puntu horretan φ_x-ren desberdina izateko moduan $\psi(\mathbf{x})$ definitzeko aukera edukiko dugunik ez dirudi.

Bigarren kasu horretan ψ eta φ_x funtzioak **x** puntuan lokalki desberdintzea lortuko ez dugun arren, ψ funtzioa φ_x-tik orokorrean desberdintzeko moduan definitzen saia gaitezke. Kasu honetan φ_x-ri buruz dakigun bakarra bere **x** indizea A-ren barnean ez dagoela da, ondorioz ψ-ren indize guztiak A-ren barnean egotea lortzea irtenbide ona litzateke. Horrela P_x ψ funtzioa konputatzen duten programetako bat ez dela ziur jakingo dugu, eta ondorioz, hau φ_x-ren desberdina izango da (zein puntu zehatzetan jakingo ez dugun arren).

Muga jakin baten berdina edo txikiagoak diren balioentzat ezezik, datu guztientzat gure funtzio diagonala konbergentea izatea eta sarrerakoaren desberdina den balio bat itzultzea egingo dugu. Horrela bere indize guztiek, nahi adina handiak izan arren, A multzoak ezartzen dituen murrizpenak betetzen dituztela ziur jakingo dugu. Osoa izango denez, **h** deitzea egokiagoa izango da.

- **x** A multzoan badago, h(**x**) balioa $\varphi_x(\mathbf{x})+\mathbf{x}+1$ bezala definituko dugu. Horrela **x**-ren desberdina izango da eta gainera $\varphi_x(\mathbf{x})$-ren desberdina ere. Hau egin dezakegu $\varphi_x(\mathbf{x})\downarrow$ dela badakigulako.

- **x** A multzoan ez badago, ez dugu **x** puntua behar inolako funtzio konputagarritik lokalki desberdintzeko. Hala ere, h(**x**) **x**+1 bezala definituko dugu (**x**-ren desberdina den beste edozein balio ere baliagarria da). Horrela gure h funtzioaren indizeak A multzoan egotea eta φ_x funtziotik era orokorrean desberdintzea lortuko dugu.

Eskema hurrengo adibide-taulan argitzen dugu. Bertan funtzio berria, funtzio batzuetatik diagonaleko puntuan desberdintzen dela eta beste batzuetatik era orokorrean desberdintzen dela ikus dezakegu. Azkeneko honetan ezin dugu jakin zein puntu zehatzetan desberdintzen diren, baina ziur dakiguna da baten indizeak A-n daudela eta bestearenak, berriz, ez. Funtzioaren definizio formala honakoa da:

$$h(x) \cong \begin{cases} \varphi_x(x)+x+1 & x \in A \\ x+1 & x \notin A \end{cases}$$

		$\notin A$	$\in A$	$\notin A$	$\in A$	$\notin A$	$\notin A$	$\in A$	
x	h	φ_0	φ_1	φ_2	φ_3	φ_4	φ_5	φ_6	...
0	0+1	?	$\neq 0$	?	$\neq 0$	?	?	$\neq 0$	
1	$\varphi_1(1)+1+1$	?	$\neq 1$	?	$\neq 1$	?	?	$\neq 1$	
2	2+1	?	?	?	$\neq 2$	?	?	$\neq 2$	
3	$\varphi_3(3)+3+1$	?	?	?	$\neq 3$	?	?	$\neq 3$	...
4	4+1	?	?	?	?	?	?	$\neq 4$	
5	5+1	?	?	?	?	?	?	$\neq 5$	
6	$\varphi_6(6)+6+1$	?	?	?	?	?	?	$\neq 6$	

...

A erabakigarria dela suposatuz, definitu berri dugun funtzioa honako programak konputa dezake:

if $C_A(X1)$ **then** X0 := Φ(X1, X1) + X1+1; **else** X0 := X1+1; **end if**;

Bere makroak hedatu ondoren, programa hau kode bezala **e** izango duen while programa batean bihurtuko da, **e** hori h funtzioaren indizea izango da, hau da, $h \cong \varphi_e$ beteko da. Sarrera guztientzat h funtzioak konbergitu eta sarrerakoaren desberdina den balio bat itzultzen duenez, $\mathbf{e} \in \mathbf{A}$ beteko da, $\forall y\ (h(y)\downarrow \wedge\ h(y)>y)$ baita.

Azter dezagun zein den funtzioaren portaera bere **e** indizearen gainean:

$$\varphi_e(\mathbf{e}) \overset{h \cong \varphi_e}{=} h(\mathbf{e}) \overset{h \text{ def.}}{=} \varphi_e(\mathbf{e})+\mathbf{e}+1$$

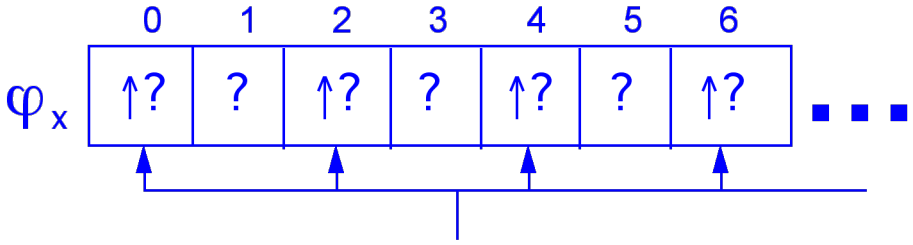
$\varphi_e(\mathbf{e}) = \varphi_e(\mathbf{e})+\mathbf{e}+1$ betetzeko aukera bakarra $\varphi_e(\mathbf{e})\uparrow$ izatea da baina hau $h \cong \varphi_e$ funtzio osoa izatearekin kontraesanean dago.

Jarraitu dugun arrazonamendu guztia zuzena da, hortaz kontraesana A erabakigarria dela suposatzetik dator. Horrela, A erabakigarria ezin dela izan frogatuta geratzen da.

{ x: ∀y $\varphi_x(2*y)\downarrow$ } multzoa ez da erabakigarria

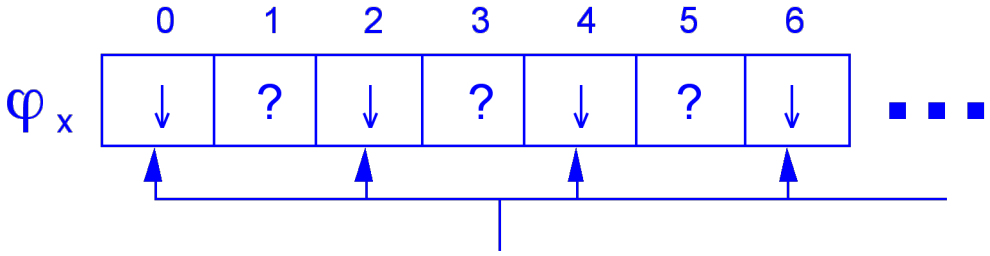
Datu bikoiti guztietan konbergitzen duten programa guztiak barnean hartzen dituen multzo horri A dei diezaiogun. A erabakigarria dela suposatu eta horretan oinarriturik, funtzio konputagarri guztien desberdina izango den **h** funtzio diagonala eraiki behar dugu. Suposaketa horrekin, **x** kodea duen edozein programa harturik, A multzoan dagoenentz galde dezakegu.

- **x** A multzoan ez badago, **x**-ri buruz lortutako informazioa minimoa da: P_x-k datu bikoitiren batentzat dibergitzen du, baina ez dakigu zeinentzat. Beraz **h** funtzioa sarrera konkretu batentzat φ_x-ren desberdina egitea zaila izango zaigu. φ_x-ri buruz dakigun bakarra bikoitiren batentzat dibergitzen duela denez, zehatz-mehatz **x** puntuan **h** eta φ_x desberdintzen saiatu beharrean, **h**-k sarrera bikoiti guztietan konbergitu dezan saia gaitezke. Horrela **h** funtzioa globalki, era orokorrean, φ_x-ren desberdina izatea lortuko dugu.



Posizio horietako batzuetan φ_x funtzioak dibergitzen duela badakigu, baina ez dakigu zehazki zeinetan, ezta gainontzekoekin zer gertatzen den ere

- **x** A multzoan badago, φ_x-k datu bikoiti guztientzat konbergitzen duela dakigu. Honek φ_x eta **h** desberdinak izatea lortu ahal izateko balio handiko informazioa ematen digu.



φ_x funtzioak posizio horietan guztietan konbergitzen duela badakigu, beraz interesatzen zaizkigun balioak kalkula ditzakegu eta beren arabera lan egin

Bigarren kasuan oraindik arazo txiki bat daukagu: $x \in A$ bada eta gainera bikoitia bada $\varphi_x(x)\downarrow$ dela dakigu, beraz bere balioa kalkulatu eta **h**(**x**) desberdina izatea

lor dezakegu. Baina x bakoitia bada $\varphi_x(x)$-ri buruz ez dakigu ezer eta lehen bezain salduta gaude. Hau konpontzeko bide bat φ_x eta **h** funtzioak **x** puntuaren ordez, $2*x$ sarreran desberdinduaraztea da, puntu horretan konbergentzia beti daukagulako.

Laburtuz, funtzio diagonala eraikitzeko gure estrategia izango da:

- **x** bakoitia bada ez daukagu murrizpenik, balio hori ez delako gure **h** inolako φ_x funtziotik deserdintzeko erabiltzen (0 itzultzea aukeratzen dugu).
- **x** bikoitia bada bi betebehar ditugu. Alde batetik, konbergentzia beharrezkoa da, **h** A-n ez dauden indizeak dituzten funtzioetatik era orokorrean desberdintzen dela ziurtatzeko. Aldi berean, **x** balio bikoitian **h** funtzioak sortuko duen emaitza agian **h** eta $\varphi_{x/2}$ lokalki desberdinduarazteko erabili beharko da. $x/2 \in A$ bada, desberdinketa lokal hori beharrezkoa da eta, **x** bikoitia denez, $\varphi_{x/2}(x)\downarrow$, eta orduan gure **h** funtzioak horren desberdina den balio bat itzuli beharko du, esaterako, $\varphi_{x/2}(x)+1$. $x/2 \notin A$ bada, berriz, ez dugu arduratu beharrik, **h** eta $\varphi_{x/2}$ era orokorrean desberdinduko direlako (0 aukeratzen dugu baita ere).

h funtzioa $x/2$ indizea duen funtzioarekin alderatzearen garrantzia azpimarratzea garrantzizkoa da. **x**-ren balio bikoitiak erabiltzean (0, 2, 4, 6, …), **h** eta $\varphi_{x/2}$ konparatzeak, gure **h** funtzio konputagarri guztiekin (φ_0, φ_1, φ_2, φ_3,…) alderatzen dugula eta guztien desberdina izatea lortzen dugula esan nahi du. Eskema hurrengo orrialdeko adibide-taulak argituko du.

Frogapena osatzeko lehen aipatutako eraiketa hori erabiliko dugu. Demagun A erabakigarria dela. Orduan honako funtzioa konputagarria da:

$$h(x) = \begin{cases} 0 & x \bmod 2 = 1 & \text{(edozer gauza izan zitekeen)} \\ 0 & x \bmod 2 = 0 \wedge x/2 \notin A & \text{(konbergitzea nahikoa da)} \\ \varphi_{x/2}(x)+1 & x \bmod 2 = 0 \wedge x/2 \in A & \text{(h funtzioa lokalki desberdintzeko)} \end{cases}$$

programa honek frogatzen duen moduan

if X1 mod 2 = 0 **and** C_A(X1 / 2) **then** X0 := Φ(X1 / 2, X1) + 1; **else** X0 := 0; **end if**;

Programa honek, bere makroak hedatu ondoren, **e** kodea izango du. Bere definizioaren arabera $\forall y$ **h**$(2*y)\downarrow$ beteko da, hortaz, $\forall y\ \varphi_e(2*y)\downarrow$. Honetatik gutxienez hiru gauza ondoriozta ditzakegu: **h** = φ_e , **e** $\in$ A eta $\varphi_e(2*e)\downarrow$. Baina azken balio hori kalkulatzen saiatzen bagara:

$$\left.\begin{array}{l} 2*e \bmod 2 = 0 \\ 2*e/2 = e \in A \end{array}\right\} \Rightarrow \varphi_e(2*e) \overset{\varphi_e = h}{=} h(2*e) \overset{h\ \text{def.}}{=} \varphi_{2*e/2}(2*e)+1 = \varphi_e(2*e)+1 \Rightarrow \varphi_e(2*e)\uparrow$$

Alde batetik $\varphi_{\mathbf{e}}(2^*\mathbf{e})\downarrow$ dela daukagu, baina bestetik $\varphi_{\mathbf{e}}(2^*\mathbf{e})\uparrow$. Kontraesan hori A erabakigarria dela suposatzetik dator, ondorioz hori ezinezkoa dela frogatuta geratzen da.

		$\in A$	$\in A$	$\notin A$	$\in A$	$\notin A$	$\in A$	$\notin A$
x	h	φ_0	φ_1	φ_2	φ_3	φ_4	φ_5	φ_6
0	1	0	3	?	0	?	7	?
1	0	?	?	?	?	?	?	?
2	5	0	4	?	2	?	7	?
3	0	?	?	?	?	?	?	?
4	0	0	2	?	4	?	7	?
5	0	?	?	?	?	?	?	?
6	7	0	7	?	6	?	4	?
7	0	?	?	?	?	?	?	?
8	0	0	8	?	8	?	4	?
9	0	?	?	?	?	?	?	?
10	4	0	4	?	10	?	3	?

$A= \{ x: W_X=\{y: y \bmod 2 = 0\} \}$ ez da erabakigarria

Demagun A erabakigarria dela, hau da, bere ezaugarri funtzioa C_A konputagarria dela. Horrela izanik, **x** indizea duen funtzio bat emanda A multzokoa denentz galdetu ahal izango dugu eta honetan oinarrituz funtzio diagonal konputagarri bat, Ψ, definituko dugu. Hau eraikitzeko Ψ funtzio konputagarri guztietatik desberdinduaraziko dugu, Ψ eta φ_X funtzioak gutxienez puntu batean desberdinak direla ziurtatuz.

- **x** A multzoan badago, φ_x funtzioaren eremua bikoiti guztien multzoa dela jakingo dugu (hau da, **x** bikoitia bada $\varphi_x(x)\downarrow$ eta bakoitia izanik, $\varphi_x(x)\uparrow$), eta $\Psi(x)$ definitu ahal izango dugu, esaterako, lehen kasuan dibergi dezan eta bigarrengoan konbergi dezan.

- $x \notin A$ bada, φ_x funtzioak balio bikoitien multzoarekin bat ez datorren eremu bat duela jakingo dugu bakarrik, baina ezin dugu jakin zein den horren arrazoia: φ_x bikoitiren batentzat dibergentea dela edo bakoitiren batentzat konbergentea. Funtzioaren portaera orokorrari buruzko informazioa edukiko dugu, baina **x** hitzaren gainean, ezta beste edozein hitzen gainean ere, zer egiten duen jakiteko modurik ez dugu izango (zehazki $\varphi_x(x)\uparrow$ edo $\varphi_x(x)\downarrow$ den ez dakigu). $\Psi(x)$ definitzeko modurik ez dugu izango, puntu horretan desberdina izan dadin.

Bigarren kasu honetan Ψ eta φ_X funtzioak **x** hitzan edo beste edozein hitzetan lokalki desberdintzea lortuko ez dugunez, Ψ funtzioa φ_X-tik orokorrean desberdinduarazten saiatu beharko dugu. Ψ funtzioari A multzoak ezartzen dituen murrizpenak betearaziko dizkiogu, hau da, bikoiti guztietan, eta horietan soilik, konbergituaraziko diogu. Beraz, <u>**x** bakoitia bada Ψ funtzioak dibergitu egingo du</u> eta <u>**x** bikoitia bada konbergitu egin beharko du</u>. Hitz bikoitietarako itzuli beharko duen balioa ezingo da edozein izan, aurrean esan bezala Ψ funtzioa eta A multzokoak direnak sarreraren batean lokalki desberdintzeaz ere arduratu behar baikara.

- **x** bikoitia bada eta $x \in A$, badakigu $\varphi_x(x)\downarrow$ dela. Hortaz, $\Psi(x)$-k ere konbergentea izan beharko du, eta gainera $\varphi_x(x)$-ren desberdina, esaterako $\varphi_x(x)+3$.

- **x** bikoitia bada eta $x \notin A$, $\varphi_x(x)\uparrow$ edo $\varphi_x(x)\downarrow$ izango ote den ez dakigu baina $\Psi(x)$ konbergentea izan behar da ezarri dugun murrizpen orokorrarengatik. Edozein balio itzul dezake behintzat, $\varphi_x(x)$ zer den kontutan hartu gabe, Ψ funtzioaren indizeak A multzoan egongo direlako eta φ_x-renak, aldiz, ez.

Oraindik ere arazo bat dugu: Ψ funtzioa, indize bakoitia izanda A multzoan dauden funtzioen desberdina izatea nola lortu?. Horietatik era lokalean desberdindu beharko ginateke, baina ezin dugu. $x \in A$ izanik φ_x funtzioak **x** balio bakoitiaren gainean

dibergituko duela badakigu. Baina murrizpen orokorrak Ψ funtzioa balio bakoiti horren gainean dibergitzera ere behartzen du, eta ondorioz bi funtzioak puntu horretan desberdintzeko gai ez gara.

Azken finean, zailtasunaren oinarria honetan datza: Ψ funtzioa datu bikoitien gainean definitzeko askatasuna bakarrik daukagun bitartean, aldi berean programa guztiak, kode bikoitikoak nahiz bakoitikoak kontuan hartu behar ditugu, bai batzuk, bai besteak A multzoan egon daitezkeelako. Bi beharrak bete ahal izateko, **x** sarrera $\varphi_{\mathbf{x}}$ funtzioan kontuan hartu ordez, **x** hitz bikoitia Ψ funtzioa $\varphi_{\mathbf{x}/2}$-tik desberdintzeko balizko puntu bezala hartuko dugu, eta beraz, diagonal nagusiaren ordez 2 maila duen beste bat erabiliko dugu.

Aurreko eskema kontuan hartuz, Ψ: $\Sigma^* \longrightarrow \Sigma^*$ funtzioa honela definituko dugu:

$$\psi(x) \cong \begin{cases} \varphi_{x/2}(x)+3 & x \bmod 2 = 0 \ \wedge \ x/2 \in A \\ 3 & x \bmod 2 = 0 \ \wedge \ x/2 \notin A \\ \perp & x \bmod 2 = 1 \end{cases}$$

Funtzio hori konputagarria da, honako programak frogatzen duelarik:

if X1 mod 2 = 0 **then**

if C_A(X1 / 2) **then** X0 := Φ(X1/2, X1)+3; **else** X0 := 3; **end if**;

else X0 := $\perp\!\!\!\perp$; **end if**;

Programa honetatik, bere makroak hedatu ondoren, **e** kodea izango duen while programa lor dezakegu, hortaz, $\Psi \cong \varphi_{\mathbf{e}}$. Funtzioak sarrera bikoiti guztiekin konbergitu eta sarrera bakoitiekin dibergitzen duenez, **e**$\in$A betetzen da.

Ikus dezagun zein den Ψ funtzioaren portaera 2***e** hitzaren gainean aplikatzerakoan. Hitz bikoitia da eta gainera (2***e**)/2$\in$A, beraz aukera bakarra:

$$\varphi_{\mathbf{e}}(2*\mathbf{e}) \overset{\psi \text{ konput.}}{=} \Psi(2*\mathbf{e}) \overset{\psi \text{ def.}}{=} \varphi_{2*\mathbf{e}/2}(2*\mathbf{e}) + 3 = \varphi_{\mathbf{e}}(2*\mathbf{e}) + 3 \Rightarrow \varphi_{\mathbf{e}}(2*\mathbf{e})\uparrow$$

argi eta garbi kontraesana dena, $\varphi_{\mathbf{e}}(\mathbf{2}*\mathbf{e})$ konbergentea delako (**e**$\in$A izateagatik).

Datu bikoiti bat (2***e**) aurkitu dugu, zeinaren gainean Ψ funtzioak bere desberdina den emaitza bat eman behar duen. Jarraitu dugun arrazonamendu guztia zuzena da, hortaz kontraesana A erabakigarria dela suposatzetik dator. A erabakigarria ez dela ondoriozta dezakegu, beraz.

		∈A	∈A	∉A	∈A	∉A	∈A	∉A
x	ψ	φ_0	φ_1	φ_2	φ_3	φ_4	φ_5	φ_6
0	$\varphi_0(0)+3$	↓	↓	?	↓	?	↓	?
1	⊥	↑	↑	?	↑	?	↑	?
2	$\varphi_1(2)+3$	↓	↓	?	↓	?	↓	?
3	⊥	↑	↑	?	↑	?	↑	?
4	3	↓	↓	?	↓	?	↓	?
5	⊥	↑	↑	?	↑	?	↑	?
6	$\varphi_3(6)+3$	↓	↓	?	↓	?	↓	?
7	⊥	↑	↑	?	↑	?	↑	?
8	3	↓	↓	?	↓	?	↓	?
9	⊥	↑	↑	?	↑	?	↑	?
10	$\varphi_5(10)+3$	↓	↓	?	↓	?	↓	?

Diagonalizazioaren aplikazioaren inguruko hausnarketarekin jarraituz, har dezagun $B = \{ x \in \Sigma^* : W_x = \{y: y \bmod 2 = 0\} = R_x \}$ multzoa une batez. Hau ere ez da erabakigarria eta diagonalizazioaren bidez frogatzeko aurreko eskema zein neurritaraino izango litzatekeen erabilgarria ikus dezagun.

Kasu honetan definitu beharko genukeen funtzioaren indizeak B multzoan egon beharko lirateke, aurreko kasuan gertatutako arazo bera konpontzeko: funtzio hori, beren indizeak B multzoan ez dituzten funtzioetatik orokorrean bereiztu ahal izateko. Eraiki dugun Ψ funtzioak B-ren lehen murrizpena betetzen du (hots, bere domeinua bikoiti guztien multzoa da), baina heinean, aldiz, bikoitiak nahiz bakoitiak izan

ditzakegu, beraz bere **e** indizea ez da B multzoan egongo eta ez dugu gure helburua lortuko. Kasu honetan eraiki beharko genukeen funtzio diagonalak bere heinean bikoiti guztiak, eta bikoitiak bakarrik, daudela kontrolatu beharko luke, emaitza bikoitiak datu bikoitien artean era ordenatuan banatuz, adibidez.

$$\zeta(x) = \begin{cases} 2 & 2*x \in R_x \\ 8*x & \text{bestela} \end{cases}$$ **funtzioa ez da konputagarria**

ζ konputagarria dela suposatzen dugu eta bere aldetik funtzio konputagarri guztien desberdina izango den funtzio diagonal konputagarri bat definituko dugu.

ζ konputagarria bada, programa guztiak harturik, ζ funtzioak 2 balioa zeintzuetarako hartzen duen eta beste balioren bat zein programetarako hartzen duen bereiztu ahal izango dugu:

- $\zeta(\mathbf{x}) \neq 2$ bada, orduan $2^*\mathbf{x}$ hitza $\varphi_{\mathbf{x}}$-ren heinean ez dagoela badakigu, eta beraz irudi moduan $2^*\mathbf{x}$ sortzen duen sarrerarik ez dagoela. Eraikiko dugun Ψ funtzioa $\varphi_{\mathbf{x}}$ horren desberdina izan dadin, $2^*\mathbf{x}$ hitza Ψ-ren heinean sartzeko modua egingo dugu. Adibidez, $\mathbf{x}$ sarrerarentzat Ψ funtzioak $2^*\mathbf{x}$ itzuliko du.

- $\zeta(\mathbf{x}) = 2$ bada, $2^*\mathbf{x}$ hitza $\varphi_{\mathbf{x}}$-ren heinean dagoela badakigu, eta beraz $\varphi_{\mathbf{x}}(\mathbf{y}) = 2^*\mathbf{x}$ betetzen duen $\mathbf{y}$ sarrera bat badagoela badakigu. $\mathbf{y}$ hori zehatz-mehatz zein den ez dakigun arren. Gure Ψ funtzioa $\varphi_{\mathbf{x}}$ horren desberdina izan dadin, inolako sarreratan $2^*\mathbf{x}$ ez itzultzea lortu behar dugu. Hau aurreko erabakiarekin bateragarria da, $\zeta(\mathbf{x}) = 2$ betetzen duten $\mathbf{x}$ balioak eta $\zeta(\mathbf{x}) = 8^*\mathbf{x} \neq 2$ betetzen duten $\mathbf{x}$ balioak berdinak ezin baitira izan (ezta beraien bikoitzak ere). $\mathbf{x}$ datuarentzat funtzioa indefinitua uztea nahikoa izango zaigu, beste inolako sarrerantzat $2^*\mathbf{x}$ emaitza zehatza ez dela sortuko badakigulako.

		$\zeta(0)=2$	$\zeta(1)=8$	$\zeta(2)=2$	$\zeta(3)=2$	$\zeta(4)=32$	
		$0\in R_0$	$2\notin R_1$	$4\in R_2$	$6\in R_3$	$8\notin R_4$	
x	Ψ	φ_0	φ_1	φ_2	φ_3	φ_4	
0	$\perp$	0?	$\neg(\varphi_1(0)=2)$	4?	6?	$\neg(\varphi_4(0)=8)$	
1	2	0?	$\neg(\varphi_1(1)=2)$	4?	6?	$\neg(\varphi_4(1)=8)$	
2	$\perp$	0?	$\neg(\varphi_1(2)=2)$	4?	6?	$\neg(\varphi_4(2)=8)$	
3	$\perp$	0?	$\neg(\varphi_1(3)=2)$	4?	6?	$\neg(\varphi_4(3)=8)$	...
4	8	0?	$\neg(\varphi_1(4)=2)$	4?	6?	$\neg(\varphi_4(4)=8)$	
			...				

Ideia aurreko adibide-taulan adierazten dugu.

Esandako eskema hori frogapena egiteko erabiltzen dugu. Demagun ζ konputagarria dela. Orduan honako funtzioa konputagarria izango da, ondoren emandako programak frogatzen duenez:

$$\psi(x) \cong \begin{cases} 2*x & \zeta(x) \neq 2 \\ \perp & \zeta(x) = 2 \end{cases}$$

if $\zeta(X1) \neq 2$ **then** X0 := 2*X1; **else** X0 := $\perp\!\!\!\perp$; **end if**;

Aurreko programak, bere makroak hedatu eta lortutako while programa kodetu ondoren, $\psi \cong \varphi_{\mathbf{e}}$ beteko duen **e** kode bat edukiko du.

2***e** balioa $R_{\mathbf{e}}$ multzokoa izango ote denentz galde dezakegu, eta bi alternatibetatik bat bera ere onargarria ez dela ikusiko dugu.

$$2*\mathbf{e} \notin R_{\mathbf{e}} \Rightarrow \zeta(\mathbf{e}) = 8*\mathbf{e} \neq 2 \overset{\psi \text{ def.}}{\Rightarrow} \psi(\mathbf{e}) = 2*\mathbf{e} \overset{\psi \cong \varphi_e}{\Rightarrow} \varphi_{\mathbf{e}}(\mathbf{e}) = 2*\mathbf{e} \Rightarrow 2*\mathbf{e} \in R_{\mathbf{e}}$$

$$2*\mathbf{e} \in R_{\mathbf{e}} \Rightarrow \zeta(\mathbf{e}) = 2 \overset{\psi \text{ def.}}{\Rightarrow} \psi(\mathbf{e}) \uparrow \Rightarrow 2*\mathbf{e} \notin \text{ran}(\psi) \overset{\psi \cong \varphi_e}{\Rightarrow} 2*\mathbf{e} \notin R_{\mathbf{e}}$$

Beraz kontraesana sortzen da, ζ-ren konputagarritasunaren hipotesitik bakarrik etor daitekeena, horrela azken hori faltsua dela frogatuta geratzen da.

$B = \{ \mathbf{x} \in \Sigma^* : \exists \mathbf{z} \;\; 2*\mathbf{z} \in R_{\mathbf{x}} \}$ multzoa kontuan hartuko bagenu, zeinen ezaugarri funtzioak formalki ζ-ren antza duen, hasiera batean diagonalizazioaren teknika antzeko eraren batera aplika zitekeela pentsa genezakeen. Alabaina, horrela egiteak sortuko lituzkeen zailtasunak ikus ditzakegu, teknika hau ezingo dela beti aplikatu ondorioztatzeko.

x kodea duen programa baten gainean aplikatuta, B multzoaren ezaugarri funtzioak $\varphi_{\mathbf{x}}$-ren inolako balio lokalen inguruko informazio nahikorik ez du inoiz ematen. Beraz, ψ funtzioa definitzen dugunean, puntu zehatz batean $\varphi_{\mathbf{x}}$ funtzioaren desberdina izatera ezin dugu behartu, eta horrela funtzioak era orokorrean desberdinduarazi beharko ditugu:

- $\mathbf{x} \notin B$ denean, gure $\varphi_{\mathbf{x}}$ funtzioak bere heinean balio bikoitirik ez duela dakigu. Orduan ψ funtzioak sarreraren batentzat balio bikoiti bat itzultzea arrazoizkoa dirudi, horrela bi funtzioak desberdinak izatea lortuko baitugu.

- $\mathbf{x} \in B$ denean, $\varphi_{\mathbf{x}}$ funtzioari buruz dakigun bakarra bere heinak balio bikoitiren bat duela da. Hortaz, ψ funtzioak balio bikoitirik ez itzultzea interesatuko zaigu, biak desberdinak izan daitezen.

  Bi murrizpenak bateraezinak direla argi eta garbi ikusten da.

$A = \{ x: 2*x \in R_x \wedge 2*x \in W_x \}$ ez da erabakigarria

A erabakigarria dela suposatu eta horretan oinarrituta funtzio konputagarri guztien desberdina izango den ψ funtzio diagonala eraiki behar dugu. Suposaketa horrekin, **x** indizea duen edozein programa hartuta, A multzoan dagoenentz galde dezakegu.

- $x \notin A$ bada, **x**-ri buruz dugun informazioa edo $\varphi_x(2*x)\uparrow$ dela edo bestela 2***x** heinean ez dagoela da. Baina ez dakigu bietako zein gertatzen den edo biak gertatzen baldin badira ere. ψ eta φ_x funtzioak bereiztarazteko 2***x** puntuan egingo dugu, bere irudia 2***x** izan dadin bidea jarriz. Horrela, aipatu bi kasuetan bi funtzioak desberdinak izango dira, $2*x$ balioa ψ-ren eremuan eta heinean sartzen baitugu.

- $x \in A$ bada, φ_x funtzioak 2***x** hitzan konbergitzen duela eta gainera 2***x** heinean dagoela dakigu, nahiz eta hori zeinen irudia den ezezaguna izan. ψ funtzioa φ_x-ren desberdina izatea lortzeko, 2***x** puntuan dibergiaraztea nahikoa izango da.

Beraz, ez dugu zehazki diagonala pasatzen, sarrera bikoitiek osatzen duten diagonala baizik, sarrera bakoitietan gertatzen denak ariketaren ebazpenean eraginik ez duelako. Hobeto esanda, eragin pittin bat badu, ψ funtzioak balio horietarako konbergituko balu bere heinean komeni ez zaizkigun elementuak sartzeko arriskua edukiko genuke. Horrela, datu bakoiti guztietan dibergituko dugu. Hurrengo orrialdeko irudian diagonalean pasatzen ditugun eta funtzio berria konputagarri bakoitzetik desberdintzeko erabiliko diren gelaxkak ikus ditzakegu.

Aipatu eraikuntza hori frogapena egiteko erabiliko dugu. Demagun A erabakigarria dela. Orduan honako funtzio hau ere konputagarria izango da, ondoren jartzen dugun programak frogatzen duen legez:

$$\psi(x) \cong \begin{cases} \perp & x \bmod 2 = 1 \\ \perp & x \bmod 2 = 0 \wedge x/2 \in A \\ x & x \bmod 2 = 0 \wedge x/2 \notin A \end{cases}$$

if X1 mod 2 = 0 **and not** C_A(X1 / 2) **then** X0 := X1; **else** X0 := $\perp\!\!\!\perp$; **end if**;

Programa hau, bere makroak hedatu ondoren, **e** kodea edukiko duen while programa batean bihurtuko da, hots, $\psi \cong \varphi_e$ beteko da. Horrelako funtziorik ez dagoeka ikus dezagun, bere indizea ez A-n, ezta bere osagarrian ere, ezingo litzakeelako egon, $\psi(2*\mathbf{e})$-rentzat baliozko definiziorik ez baitago:

$$e \in A \overset{\psi \text{ def.}}{\Rightarrow} \psi(2*e)\uparrow \overset{\psi \cong \varphi_e}{\Rightarrow} \varphi_e(2*e)\uparrow \Rightarrow 2*e \notin W_e \Rightarrow e \notin A$$

$$e \notin A \overset{\psi \text{ def.}}{\Rightarrow} \psi(2*e)=2*e \overset{\psi \cong \varphi_e}{\Rightarrow} \varphi_e(2*e)=2*e \Rightarrow 2*e \in W_e \wedge 2*e \in R_e \Rightarrow e \in A$$

Beraz $e \in A \Leftrightarrow e \notin A$ daukagu. Kontraesan hau A erabakigarria dela suposatzetik dator, beraz hori ezinezkoa dela frogatuta geratzen da.

		$\in A$	$\in A$	$\notin A$	$\in A$	$\notin A$
x	ψ	φ_0	φ_1	φ_2	φ_3	φ_4
0	$\uparrow$	$\downarrow$0?	2?	$\neq$4?	6?	$\neq$8?
1	$\uparrow$	0?	2?	$\neq$4?	6?	$\neq$8?
2	$\uparrow$	0?	$\downarrow$2?	$\neq$4?	6?	$\neq$8?
3	$\uparrow$	0?	2?	$\neq$4?	6?	$\neq$8?
4	4	0?	2?	$\uparrow\neq$4?	6?	$\neq$8?
5	$\uparrow$	0?	2?	$\neq$4?	6?	$\neq$8?
6	$\uparrow$	0?	2?	$\neq$4?	$\downarrow$6?	$\neq$8?
7	$\uparrow$	0?	2?	$\neq$4?	6?	$\neq$8?
8	8	0?	2?	$\neq$4?	6?	$\uparrow\neq$8?

$\{ x: \forall y\, \varphi_x(y)\downarrow \wedge R_x \neq \Sigma^* \}$ ez da erabakigarria

Multzo honek definitzen dituen funtzioek aldi berean bi ezaugarri betetzen dituzte: alde batetik sarrera guztietarako konbergitzen dute, eta bestetik balizko irudi guztiak ez daude bere heinean. Beraz, multzo horretan funtzio oso eta ez-supraiektiboak ditugu, eta horregatik OES deituko diogu.

Demagun OES multzoa erabakigarria dela. Orduan bere ezaugarri funtzioa C_{OES} konputagarria da. Beraz, **x** indizea duen funtzio bat emanik, hau OES multzoaren barnean dagoenentz galde dezakegu, eta horretan oinarriturik ψ funtzio konputagarria defini dezakegu. Funtzio hori eraikitzeko funtzio konputagarri guztiak erabiliko ditugu eta guztietatik desberdinduaraziko dugu, hots, ψ eta φ_x funtzio bakoitza gutxienez puntu batean desberdinduaraziko ditugu.

- **x** OES multzoan badago, φ_x osoa eta ez-supraiektiboa dela jakingo dugu, beraz $\varphi_x(\mathbf{x})\downarrow$. Era asko ditugu $\psi(\mathbf{x})$ itzulitako balio horren desberdina izan dadin: $\varphi_x(\mathbf{x})$-ren balioa nolabait aldatuko duen balio bat hartuz edo bestela, $\psi(\mathbf{x})\uparrow$ eginez.

- **x** OES multzoan ez badago, φ_x edo ez-osoa edo supraiektiboa dela jakingo dugu. Baina **x** hitzaren gainean zer egiten duen ($\varphi_x(\mathbf{x})\downarrow$ edo $\varphi_x(\mathbf{x})\uparrow$ den) jakiteko modurik ez da izango, eta $\psi(\mathbf{x})$ hitz horretan desberdina izateko moduan ezingo dugu definitu.

Bigarren kasuan ψ eta φ_x funtzioak **x** hitzan lokalki desberdintzea lortu ezingo dugunez, ψ funtzioa, φ_x-tik era orokorrean desberdintzeko moduan definitu beharko dugu. ψ funtzioa osoa eta ez-supraiektiboa izan dadin definituko dugu, OES multzoak ezarritako baldintzak bete ditzan. Gure helburua $\mathrm{dom}(\psi) = \sum^* \neq \mathrm{ran}(\psi)$ lortzea izango da. Lehenengoa erraza da eta bigarrena lortzeko ψ eraikitzean, kontu pixka batekin, emaitz zehatz bat baztertzea nahikoa da. Adibidez, funtzioak 0 itzultzea ekidin dezakegu.

- **x** OES multzoan badago, $\psi(\mathbf{x})$ balioa $\varphi_x(\mathbf{x})+1$ bezala definituko dugu. Balio hori konbergentea eta 0-ren desberdina izango da, eta ψ eta φ_x lokalki desberdintzea ahalbidetuko du.

- **x** OES multzoan ez badago, $\psi(\mathbf{x})$ 0-ren desberdina den edozein baliorekin definituko dugu, esaterako 1. Horrela, hasierako helburua mantentzen dugu, ψ osoa izatea baina 0 balioa bere heinean ez edukitzea. ψ eta φ_x funtzioak orokorrean desberdinduko dira.

  Arrazonamendu hori jarraituz funtzio diagonala definituko dugu:

$$\psi(x) = \begin{cases} \varphi_x(x)+1 & x \in OES \\ 1 & x \notin OES \end{cases}$$

Eskema honako adibide-taulan azaltzen dugu:

		$\in$OES	$\in$OES	$\notin$OES	$\notin$OES	$\in$OES	$\in$OES	
		$R_0 \neq \Sigma^*$	$R_1 \neq \Sigma^*$	R_2?	R_3?	$R_4 \neq \Sigma^*$	$R_5 \neq \Sigma^*$	
x	Ψ	φ_0	φ_1	φ_2	φ_3	φ_4	φ_5	
0	$\varphi_0(0)+$	↓	↓	?	?	↓	↓	
1	$\varphi_1(1)+$	↓	↓	?	?	↓	↓	
2	1	↓	↓	?	?	↓	↓	
3	1	↓	↓	?	?	↓	↓	…
4	$\varphi_4(4)+$	↓	↓	?	?	↓	↓	
5	$\varphi_5(5)+$	↓	↓	?	?	↓	↓	
		…						

OES erabakigarria dela suposatzen dugunez, ψ funtzioaren konputagarritasuna programa honen bidez frogatuta geratuko litzateke:

if C_{OES} (X1) **then** X0:= Φ(X1, X1)+1; **else** X0:=1; **end if**;

Programa horrek, bere makroak hedatu eta lortutako while programa kodetu ondoren, **e** kode bat edukiko du, hau da, $\psi = \varphi_e$ beteko da. Argi eta garbi ψ funtzioa osoa da, beti konbergitzen baitu, eta ez-supraiektiboa, $0 \notin$ heina(ψ) baita, orduan **e** $\in$ OES. Baina funtzioa bere indizearen gainean aplikatzen saiatzen bagara:

$$\mathbf{e} \in OES \overset{\psi \text{ def.}}{\Rightarrow} \varphi_{\mathbf{e}}(\mathbf{e}) = \psi(\mathbf{e}) = \varphi_{\mathbf{e}}(\mathbf{e})+1 \Rightarrow \varphi_{\mathbf{e}}(\mathbf{e})\uparrow$$

eta honekin kontraesanera heldu gara, definitutako ψ funtzioa osoa delako. Kontraesan hori OES erabakigarria dela suposatzetik dator, beraz hau ezinezkoa dela frogatuta geratzen da.

$$\zeta(x) = \begin{cases} 2*x & \varphi_x \textbf{ osoa eta supraiektiboa} \\ & \textbf{ez da konputagarria} \\ 2*x+1 & \textbf{bestela} \end{cases}$$

Demagun ζ funtzioa konputagarria dela. Diagonalizazioaren teknika erabiliz eta ζ-ren konputagarritasunean oinarrituz, konputagarria den baina aldi berean inolako funtzio konputagarrirekin bat ez datorren ψ funtzio bat aurki dezakegu eta horrela kontraesanera hel gaitezke. Horretarako, ψ funtzioa, osoa eta supraiektiboa eta gainera funtzio oso eta supraiektibo guztien desberdina egingo dugu.

Osoa izan dadin, puntu bakoitzean balio bat esleitu behar dugu (hau erraza da) eta supraiektiboa izan dadin zenbaki arrunt guztiak noizbait esleituak izango direla ziurtatu behar dugu (hau da zailena). Hori egiteko modu bat, arrunt guztiak, ahal dela, ordenean esleitzea da: lehenik 0, gero 1, ondoren 2, etab., lehendabizi oraindik esleitu ez dugun lehen balioa kokatzen saiatuz beti, eta ezin bada (osoa eta supraiektiboa den $\varphi_{\mathbf{x}}$ funtzioren batetik **x** puntuan lokalki desberdinduarazi behar dugulako), aske dagoen bigarrena. Bietako bat esleitzea beti izango da posible.

- $\zeta(\mathbf{x})$ bakoitia bada, orduan $\varphi_{\mathbf{x}}$ ez da osoa eta supraiektiboa, eta gure ψ funtzioa $\varphi_{\mathbf{x}}$-tik orokorrean desberdinduko da, horregatik $\psi(\mathbf{x})$-ri oraindik esleitu ez den baliorik txikiena ematen diogu.

- $\zeta(\mathbf{x})$ bikoitia bada, orduan $\varphi_{\mathbf{x}}$ funtzioa osoa eta supraiektiboa da, eta ψ eta $\varphi_{\mathbf{x}}$ funtzioak **x** puntuan desberdinduko dira. Oraindik esleitu ez den baliorik txikiena, $\varphi_{\mathbf{x}}$ osoa izateagatik existitzen den $\varphi_{\mathbf{x}}(\mathbf{x})$-rekin alderatzen dugu. Desberdinak badira hori izango da $\psi(\mathbf{x})$-ri emango zaion balioa, eta berdinak badira oraindik esleitu gabe dagoen hurrengo balioa bilatzen dugu.

Horrela beti esleitzen dugu balio bat (ψ osoa izango da) eta azkenean balizko balio guztiak esleitzen ditugu (ψ supraiektiboa izango da). Definitzen ari garen funtzioa, formalki adierazita, honakoa da:

$$\psi(0) = \begin{cases} 0 & \zeta(0) \bmod 2 \neq 0 \\ 0 & \zeta(0) \bmod 2 = 0 \wedge \varphi_0(0) \neq 0 \\ 1 & \zeta(0) \bmod 2 = 0 \wedge \varphi_0(0) = 0 \end{cases}$$

$$\psi(x) = \begin{cases} \mu z(\forall y < x(\psi(y) \neq z) \wedge \varphi_x(x) \neq z) & \zeta(x) \bmod 2 = 0 \\ \mu z(\forall y < x(\psi(y) \neq z)) & \zeta(x) \bmod 2 \neq 0 \end{cases}$$

non $\mu\mathbf{z}P(\mathbf{z})$ adierazpenak $P(\mathbf{z})$ predikatua betetzen duen **z** txikiena esan nahi duen.

Azaldu dugun moduan, ψ funtzioak honako adibide-taulan erakusten dugun diagonalizazio eskema jarraitzen du:

		ζ(0) bikoitia	ζ(1) bakoitia	ζ(2) bikoitia	ζ(3) bikoitia	ζ(4) bakoitia	ζ(5) bikoitia	ζ(6) bakoitia	
x	**ψ**	**φ_0**	**φ_1**	**φ_2**	**φ_3**	**φ_4**	**φ_5**	**φ_6**	
0	**1**	0	?	↓	↓	?	↓	?	
1	**0**	↓	?	↓	↓	?	↓	?	
2	**3**	↓	?	2	↓	?	↓	?	
3	**4**	↓	?	↓	2	?	↓	?	…
4	**2**	↓	?	↓	↓	?	↓	?	
5	**5**	↓	?	↓	↓	?	7	?	
6	**6**	↓	?	↓	↓	?	↓	?	
		…							

ψ funtzioaren konputagarritasuna ζ-ren konputagarritasunean datza, eta honako programarekin frogatuta geratzen da:

```
LEHEN:=0; BIGARREN:=1; LAG:=0;
while LAG ≤ X1 loop
    if  ζ(LAG) mod 2 ≠ 0
    then   X0:= LEHEN; LEHEN := BIGARREN; BIGARREN := BIGARREN+1;
    else   Z:=  Φ(LAG, LAG);
        if Z ≠ LEHEN
        then                                X0:=LEHEN;  LEHEN:=BIGARREN;
BIGARREN:=BIGARREN+1;
        else X0:= BIGARREN; BIGARREN := BIGARREN+1;
        end if;
    end if;
    LAG := LAG +1;
```

end loop;

Bestalde, ψ funtzioa, eraiketa egin dugun erarengatik, funtzio konputagarri guztien desberdina da, eta hemendik orain ikusiko dugun kontraesanera helduko gara.

Badago **e** indize bat $\psi \cong \varphi_{\mathbf{e}}$ betetzen duena. Gainera eraiketagatik bijektiboa da, orduan $\zeta(\mathbf{e}) \bmod 2 = 0$. Baldin $v = \mu z(\forall y<\mathbf{e}\ (\psi(y)\neq z) \wedge \varphi_{\mathbf{e}}(\mathbf{e}) \neq z)$ orduan $\psi(\mathbf{e}) = v$, ψ -ren definizioarengatik. Hortaz, $\psi(\mathbf{e}) = v \Leftrightarrow \varphi_{\mathbf{e}}(\mathbf{e}) \neq v$, hau da, $\varphi_{\mathbf{e}}(\mathbf{e}) = v \Leftrightarrow \varphi_{\mathbf{e}}(\mathbf{e}) \neq v$. Kontraesana ζ konputagarria zela suposatzeagatik sortu da, beraz hori ezinezkoa dela frogatuta geratzen da.

Hau, gainera, funtzio konputagarri bijektiboen indizeen multzoa (PERM) erabakigarria ez dela frogatzeko erabili ohi diren eskemetako bat da, ψ funtzioa osoa eta supraiektiboa izateaz gain injektiboa delako, azken hau gure adibiderako beharrezkoa ez zen arren.

$A = \{ x: \varphi_x$ oszilakorra da $\}$ ez da erabakigarria.

η funtzioa oszilakorra dela diogu honakoa betetzen duenean:

I) η osoa da

II) beti $\eta(x) \neq \eta(x+1)$ betetzen da

III) ez dago x daturik $\eta(x) > \eta(x+1) > \eta(x+2)$ betetzen duenik

IV) ez dago x daturik $\eta(x) < \eta(x+1) < \eta(x+2)$ betetzen duenik

Hau da, ondoz ondoko datuetarako sortutako balioak oszilakorrak izatearen ezaugarria duenean (pauso bakoitzean gorantz-beherantz egiten dute).

Diagonalizazio metodoa erabiltzeko A erabakigarria dela suposatu behar dugu eta, funtzio konputagarri guztien desberdina izan arren, hipotesi horren arabera konputatu ahal izango den **g** funtzioa eraiki beharko dugu. Pentsa dezagun A erabakigarria dela, eta ondorioz bere ezaugarri funtzioa C_A konputagarria dela.

Bedi **x** edozein programaren kodea. Hasiera batean $\mathbf{g}(\mathbf{x})$ eta $\varphi_x(\mathbf{x})$ desberdinak izatea nahi dugu eta horretarako $C_A(\mathbf{x})$ galdetuko dugu.

- Erantzuna baiezkoa bada, erraza da: φ_x oszilakorra da, eta ondorioz osoa, beraz, $\varphi_x(\mathbf{x})\downarrow$ dela badakigu eta, behar izanez gero, bere balioa kalkula dezakegu $\mathbf{g}(\mathbf{x})$ desberdina izatea lortu ahal izateko.

- Erantzuna ezezkoa bada, berriz, φ_x-ri buruz dugun informazioa oso eskasa da, eta zehazki ez dakigu ezer, ez $\varphi_x(\mathbf{x})$-ri buruz ezta beste inolako balio zehatzi buruz ere. Horrek **g** eta funtzio konputagarri ez-oszilakorrak orokorrean bereizteko estrategia hartzera behartzen gaitu, **g** oszilakorra eginez.

Horretarako honakoa egingo dugu: oro har, **g** funtzioak 0 (datu bikoitietan) eta 10 (datu bakoitietan) balioen artean oszilatuko du. Hala ere, noizbait, datu zehatz batean, lokalki, funtzio konputagarri oszilakor batetik bereizi behar duelako balio horiek itzuli ezin baditu, orduan bere ordez 1 eta 9 erabiliko ditu, hurrenez hurren. **g**-ren definizioa honakoa izango litzateke:

$$g(x) = \begin{cases} 0 & x \bmod 2 = 0 \wedge (x \notin A \vee (x \in A \wedge \varphi_x(x) \neq 0)) \\ 1 & x \bmod 2 = 0 \wedge x \in A \wedge \varphi_x(x) = 0 \\ 9 & x \bmod 2 \neq 0 \wedge x \in A \wedge \varphi_x(x) = 10 \\ 10 & x \bmod 2 \neq 0 \wedge (x \notin A \vee (x \in A \wedge \varphi_x(x) \neq 10)) \end{cases}$$

A-ren erabakigarritasunagatik, **g** funtzioa programa honek konputatzen du:

if X1 mod 2 = 1 **then**

```
        X0 := 10;
        if C_A(X1) then
                R := Φ(X1, X1);
                if R=10 then X0 := 9; end if;
        end if;
else    X0 := 0;
        if C_A(X1) then
                R := Φ(X1, X1);
                if R=0 then X0 := 1; end if;
        end if;
end if;
```

Konputagarria denez, aurreko programaren makroen hedapena egin ondoren sortutako while programari dagokion **e** kode bat egon beharko da. Kode horrek $\mathbf{g} = \varphi_e$ beteko du. Sortutako funtzioa oszilakorra denez, $\mathbf{e} \in A$ izango da. $\varphi_e(\mathbf{e})$ balioaren izaera aztertzeko bi kasu bereiziko ditugu: **e** bikoitia edo bakoitia izatea.

Baldin **e** bikoitia bada:

$$\left.\begin{array}{r} \varphi_{\mathbf{e}}(\mathbf{e}) = 0 \\ \mathbf{e} \in A \\ \mathbf{e} \bmod 2 \;=\; 0 \end{array}\right\} \overset{\text{g def.}}{\Rightarrow} \; g(\mathbf{e}) = 1 \overset{g=\varphi_e}{\Rightarrow} \varphi_{\mathbf{e}}(\mathbf{e}) = 1 \Rightarrow \varphi_{\mathbf{e}}(\mathbf{e}) \neq 0$$

$$\left.\begin{array}{r} \varphi_{\mathbf{e}}(\mathbf{e}) \neq 0 \\ \mathbf{e} \in A \\ \mathbf{e} \bmod 2 \;=\; 0 \end{array}\right\} \overset{\text{g def.}}{\Rightarrow} \; g(\mathbf{e}) = 0 \overset{g=\varphi_e}{\Rightarrow} \varphi_{\mathbf{e}}(\mathbf{e}) = 0$$

Baldin **e** bakoitia bada:

$$\left.\begin{array}{r} \varphi_{\mathbf{e}}(\mathbf{e}) = 10 \\ \mathbf{e} \in A \\ \mathbf{e} \bmod 2 \;\neq\; 0 \end{array}\right\} \overset{\text{g def.}}{\Rightarrow} \; g(\mathbf{e}) = 9 \overset{g=\varphi_e}{\Rightarrow} \varphi_{\mathbf{e}}(\mathbf{e}) = 9 \Rightarrow \varphi_{\mathbf{e}}(\mathbf{e}) \neq 10$$

$$\left.\begin{array}{r} \varphi_{\mathbf{e}}(\mathbf{e}) \neq 10 \\ \mathbf{e} \in A \\ \mathbf{e} \bmod 2 \;\neq\; 0 \end{array}\right\} \overset{\text{g def.}}{\Rightarrow} \; g(\mathbf{e}) = 10 \overset{g=\varphi_e}{\Rightarrow} \varphi_{\mathbf{e}}(\mathbf{e}) = 10$$

e bikoitia bada, kontraesan batera iristen gara, eta bakoitia izanez gero, beste batera. **g** funtzioa ondo definituta dago baina **e**-ri ezin dio inolako baliorik esleitu, ondorioz, gure hasierako suposaketa, A erabakigarria dela, ezin da egia izan.

$\overline{K}=\{ x: \varphi_x(x)\uparrow \}$ ez da sasierabakigarria

Multzo honen sasierabakigarritasuna bere funtzio erdiezaugarriaren konputagarritasunaren menpe dago:

$$\chi_{\overline{K}}(x) \cong \begin{cases} true & x \in \overline{K} \\ \perp & x \notin \overline{K} \end{cases}$$

Definizioaren baldintzan $x \notin W_x$ edo $\varphi_x(x)\uparrow$ jartzea aukera zitekeen ere:

$$\chi_{\overline{K}}(x) \cong \begin{cases} true & \varphi_x(x)\uparrow \\ \perp & \varphi_x(x)\downarrow \end{cases}$$

$\overline{K}$ multzoa sasierabakigarria dela suposatzen dugu, eta beraz, bere funtzio erdiezaugarria konputagarria dela. Horretan oinarrituz, eta diagonalizazioa erabiliz, funtzio konputagarri guztien desberdina izango den ψ funtzio konputagarria defini dezakegu. ψ funtzioa eta φ_x funtzio konputagarri guztiak gutxienez puntu batean desberdinduaraziko ditugu, diagonaleko **x** puntuan.

- $\mathbf{x} \in \overline{K}$ bada, $\overline{K}$-ren funtzio erdiezaugarriak **x**-ren gainean aplikatzerakoan konbergitu egiten du. ψ funtzioa φ_x-ren desberdina izan dadin, eta gainera puntu horretan, **x**-n konbergitzea nahikoa da.

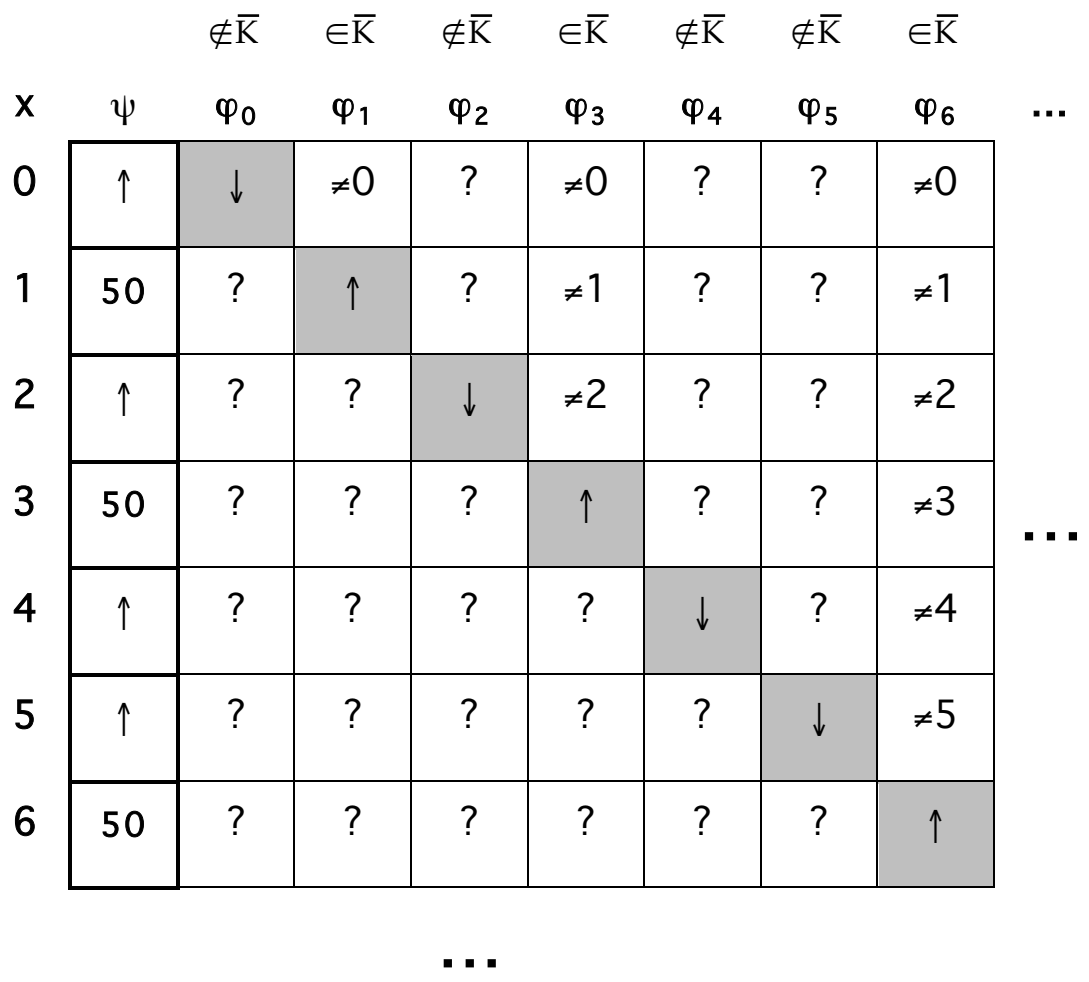
- $\mathbf{x} \notin \overline{K}$ bada, $\overline{K}$-ren funtzio erdiezaugarriak **x**-ren gainean aplikatzerakoan dibergitu egiten du. Honek esan nahi du $\chi_{\overline{K}}(\mathbf{x})$ betetzen den egiaztatze soilak ziklatzera behartzen gaituela. Zorionez, ψ funtzioa **x** puntu horretan φ_x-ren desberdina izateko dibergitzea nahikoa da, egin dezakegun gauza bakarra, bestalde.

Eskema hurrengo orrialdeko adibide-taulan azaltzen duguna da. Bertan funtzio berria φ_x funtzio bakoitzetik diagonaleko puntuan desberdintzen dela ikus dezakegu. Funtzioaren definizio formala honakoa da:

$$\psi(x) \cong \begin{cases} 50 & \chi_{\overline{K}}(x)\downarrow \\ \perp & \chi_{\overline{K}}(x)\uparrow \end{cases}$$

$\overline{K}$ sasierabakigarria dela suposatuz, definitu berri dugun funtzioa honako programak konputatzen duela daukagu:

```
X0 := χ_K̄ (X1);
X0 := 50;
```



Programa honen makroak hedatu ondoren, **e** kodea izango duen while programan bihurtuko da, hau da, $\psi \cong \varphi_e$ beteko da. ψ funtzioarentzat bere **e** indizearen gaineko portaera azter dezagun:

$$\varphi_e(\mathbf{e})\uparrow \Rightarrow \psi(\mathbf{e})\uparrow \Rightarrow \chi_{\overline{K}}(\mathbf{e})\uparrow \Rightarrow \mathbf{e} \notin \overline{K} \Rightarrow \varphi_e(\mathbf{e})\downarrow$$

$$\varphi_e(\mathbf{e})\downarrow \Rightarrow \psi(\mathbf{e})\downarrow \Rightarrow \chi_{\overline{K}}(\mathbf{e})\downarrow \Rightarrow \mathbf{e} \in \overline{K} \Rightarrow \varphi_e(\mathbf{e})\uparrow$$

ψ funtzioak **e** puntuan ezin du ez dibergitu ezta konbergitu ere, hortaz kontraesanera heldu gara.

Jarraitutako arrazonamendu guztia zuzena da, hortaz, kontraesana $\overline{K}$ sasierabakigarria dela suposatzeagatik sortu da. Ondorioz, hori ezinezkoa dela frogatuta geratzen da.

$$\xi(x) \cong \begin{cases} x^3 & \varphi_x \text{ gorakorra da} \\ \perp & \text{bestela} \end{cases} \quad \textbf{ez da konputagarria}$$

η funtzioa gorakorra dela esaten dugu, baldin eta:

I) η osoa da

II) $y>z$ betetzen duten y eta z datuetarako, $\eta(y) > \eta(z)$ betetzen da beti.

Hau da, sortutako balioak gero eta handiagoak izatearen ezaugarria duenean. Adibidez, identitatea funtzio gorakorra da.

Diagonalizazioaren metodoa ezin dugu aurreko adibidean bezala erabili. ξ -n oinarrituz, funtzio konputagarri guztien desberdina izango den ψ funtzioa eraikitzerakoan, $\xi(\mathbf{x})$-k gutxien komeni zaigun unean dibergitzen duela aurkitzen dugu: φ_x ez-gorakorra denean. Hemendik bi ondorio atera ditzakegu. Bat, ψ eta φ_x funtzioak ezin ditugu lokalki desberdindu, φ_x funtzioa gorakorra ez izateak sarrera zehatzetan zer gertatzen den ez baitu argitzen. Eta bi, ψ eta φ_x funtzioak ezin ditugu era orokorrean desberdindu, ez baitugu gure funtzio diagonala gorakorra izatea lortzeko aukerarik, **x** puntuan dibergituaraztera behartuta gaudelako (gorakorra izateak osoa ondorioztatzen duela ez dezagun ahantz).

Arazoa da <u>eremua</u> funtzio konputagarri gorakorren indizeez osatuta duen ξ funtzioa ezin dela diagonalizaziorako erabili. Hala ere, arazoa beste leku batetik heldu ahal izango dugu: bere <u>heina</u> funtzio konputagarri gorakorren indizeez osatuta duen funtzioa hartuko dugu. Funtzio berri horrekin, diagonalizazioaren arrazonamenduan konputagarri guztien ordez, funtzio konputagarri eta gorakorrak kontuan hartuko ditugu (hauek maneiatzeko erosoagoak dira, osoak direlako).

Funtzio batetik bestera pasatzeko bidea irekiko diguna multzo sasierabakigarrien karakterizazio teorema izango da. Demagun ξ konputagarria dela. Bere eremua funtzio konputagarri gorakorren indizeez osatuta dago, eta CRE deituko dugu. ξ konputagarria denez, $dom(\xi)$ = CRE multzo sasierabakigarria da. CRE hutsa ez denez (bertan gutxienez identitate funtzioa konputatzen duten programa guztiak egongo dira), karakterizazio teoremak CRE multzoa **g** funtzio oso eta konputagarriaren heinarekin bat datorrela ziurtatzen digu, hau da, CRE = ran(g) = {g(0), g(1), g(2), …, g(i), …}. Beraz, $\mathbf{i} \in$ CRE betetzen duen elementua **g**(**j**) eran idatz daiteke, **j**-ren batentzat.

Orain dugun abantaila, funtzio konputagarri gorakor guztien zerrendaketa hau $\{\varphi_{g(0)}, \varphi_{g(1)}, \varphi_{g(2)} \ldots\}$ erabil dezakegula da. Diagonalizazioa erabiliz, guzti horien desberdina izango den funtzio konputagarri eta gorakorra eraikitzen saiatuko gara.

Adibide-taula har dezagun. Zutabeen goiburukoan azaltzen diren funtzio guztiak gorakorrak, eta ondorioz osoak, direnez funtzio horiek guztiek diagonalean duten balioa zein den jakin dezakegu. Hortaz, puntu horretan beraien desberdina izango den gure **h** funtzio diagonala eraiki dezakegu, balio horri 1 gehituz, adibidez. Gainera **h** gorakorra izatea ere nahi dugu, horretarako aurreko sarreran funtzioak emandako balioa gehi dezakegu. Azaldu dugun, eta taulan agerian geratzen den, eskema definizio induktibo honen bitartez adieraz daiteke:

$$h(0) = 1 + \varphi_{g(0)}(0)$$

$$h(n+1) = h(n) + 1 + \varphi_{g(n+1)}(n+1)$$

argi ikusten den moduan, gorakorra dena.

x	h	$\varphi_{g(0)}$	$\varphi_{g(1)}$	$\varphi_{g(2)}$	$\varphi_{g(3)}$	$\varphi_{g(4)}$	...
0	$1+\varphi_{g(0)}(0)$	↓	↓	↓	↓	↓	
1	$g(0)+\varphi_{g(1)}(1)+1$	↓	↓	↓	↓	↓	
2	$g(1)+\varphi_{g(2)}(2)+1$	↓	↓	↓	↓	↓	
3	$g(2)+\varphi_{g(3)}(3)+1$	↓	↓	↓	↓	↓	...
4	$g(3)+\varphi_{g(4)}(4)+1$	↓	↓	↓	↓	↓	
5	$g(4)+\varphi_{g(5)}(5)+1$	↓	↓	↓	↓	↓	
		...					

h funtzioa konputatuko lukeen programa honakoa litzateke:

```
LAG := 0;  X0 := 0;
while LAG ≤ X1 loop
      X0 := X0 + Φ(g(LAG), LAG) + 1;
      LAG := LAG +1;
end loop;
```

h konputagarria izateagatik, $\mathbf{h} \cong \varphi_{e}$ beteko duen **e** indize bat egongo da. Gainera gorakorra denez $\mathbf{e} \in \mathrm{CRE} = \mathrm{ran}(\mathbf{g})$ izango da. Hortaz, **a** balio bat egon behar da $\mathbf{e} = \mathbf{g}(\mathbf{a})$ beteko duena, eta ondorioz $\mathbf{h} \cong \varphi_{g(a)}$. Funtzioa **a** puntuan aplikatzerakoan zer gertatzen den ikus dezagun:

Baldin **a**=0

$$\varphi_{\mathbf{g}(0)}(0) \cong \mathbf{h}(0) \cong 1 + \varphi_{\mathbf{g}(0)}(0)$$

Baldin **a**>0

$$\varphi_{\mathbf{g}(\mathbf{a})}(\mathbf{a}) \cong \mathbf{h}(\mathbf{a}) \cong \mathbf{h}(\mathbf{a}\text{-}1) + 1 + \varphi_{\mathbf{g}(\mathbf{a})}(\mathbf{a})$$

Bi kasuetan lortzen diren berdintzak egiazkoak izateko aukera bakarra $\varphi_{g(a)}(\mathbf{a})\uparrow$ betetzea da, baina hori $\mathbf{h} \cong \varphi_{g(a)}$ funtzioa gorakorra, eta beraz osoa, izatearekin kontraesanean dago. Kontraesan hori ξ funtzioa konputagarria zela suposatzeagatik etorri da, ondorioz hala ez dela frogatuta geratzen da.

Erreferentziak

[Har 87] D. HAREL. Algorithmics. The spirit of Computing. Addison-Wesley, 1987

[IIS 98] J. IBAÑEZ; A. IRASTORZA; A. SANCHEZ. *While programak. Konputagarritasun Teoria oinarritzeko tresna.* Barne-txostena UPV/EHU/LSI/TR 3-98.

[MAK 88] R. N. MOLL; M. A. ARBIB; A. J. KFOURY. A programming approach to computablity. Springer-Verlag, 1988

[SW 88] R. SOMMERHALDER; S. C. van WESTRHENEN The theory of computability. Programas, Machines, Effectiveness and Feasibility. Addison-Wesley 1.988

[Sta 81] G. STAHL. *El método diagonal en teoría de conjuntos.* Teorema. Vol 11, nº1, pp 27-35, 1981