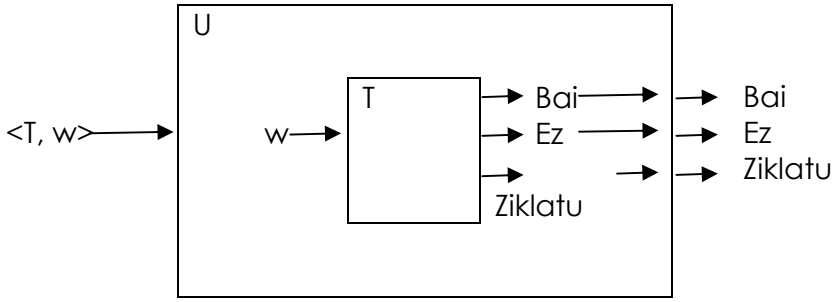
L_{bai} bereizgarria da

$$L_{bai} = \{ <T,w> \mid T \text{ makinak w hitzarentzat "Bai" erantzuten du} \}$$

L_{bai} bereizgarria dela frogatzeko, L_{bai} lengoaiako hitzentzat, hau da, *T makinak w hitzarentzat "Bai" erantzuten du* baldintza betetzen duten <T, w> erako hitzentzat (edo bikoteentzat) "Bai" erantzuten duen makina bat eraiki behar da.



U makinari (edo funtzioari) <T, w> hitza (edo bikotea) emango zaio, hau da, T makina edo funtzioa eta T makina edo funtzioarentzat datua den w elementua.

Jarraian T makina edo funtzioa w datuarekin exekutatuko da eta U funtzioak T funtzioak itzulitako emaitza jasoko du:

- T makinak edo funtzioak "Bai" itzultzen badu w hitzarentzat, orduan U funtzioak edo makinak ere "Bai" itzuliko du <T, w> hitzarentzat.
- T makinak edo funtzioak "Ez" itzultzen badu w hitzarentzat, orduan U funtzioak edo makinak ere "Ez" itzuliko du <T, w> hitzarentzat.
- T makinak edo funtzioak ziklatu egiten badu, hau da erantzunik ez badu itzultzen w hitzarentzat, U makina erantzunaren zain geratuko da eta, ondorioz, U makinak edo funtzioak ere ziklatu egingo du <T, w> hitzarentzat.

Hala ere, T makinak w-rentzat "Bai" erantzuten duenean U makinak <T, w> hitzarentzat ere "Bai" erantzungo duenez, L_{bai} lengoaia bereizgarria da, lengoaia horretakoak diren hitzentzat "Bai" erantzuten duen makina bat (U makina) existitzen delako.

Konturatu <T, w> hitza edo bikotea L_{bai} lengoaiakoa ez bada, hau da, T makinak w hitzarentzat "Ez" erantzuten badu edo T makinak w hitzarekin ziklatu egiten badu, orduan U makinak ere "Ez" erantzungo duela edo ziklatu egingo duela <T, w> hitzarekin. Beraz, L_{bai} lengoaiakoak ez diren hitzentzat ez du "Ez" erantzungo beti, batzutan ziklatu egingo baitu.

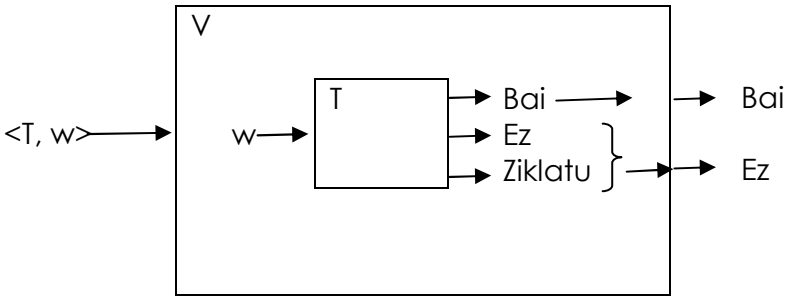
Beraz U makinaren bidez <T, w> erako hitzak edo bikoteak L_{bai} lengoaiakoak direneko kasuak ondo kontrolatzen dira baina <T, w> erako hitzak edo bikoteak L_{bai} lengoaiakoak ez direneko kasuak ez dira ondo kontrolatzen, izan ere <T, w> erako hitzak edo bikoteak L_{bai} lengoaiakoak ez direneko kasu batzuetan ez baita erantzungo, ziklatu egingo da.

L_{bai} ez da erabakigarria (erabakiezina da)

L_{bai} = { <T, w> | T makinak w hitzarentzat "Bai" erantzuten du }

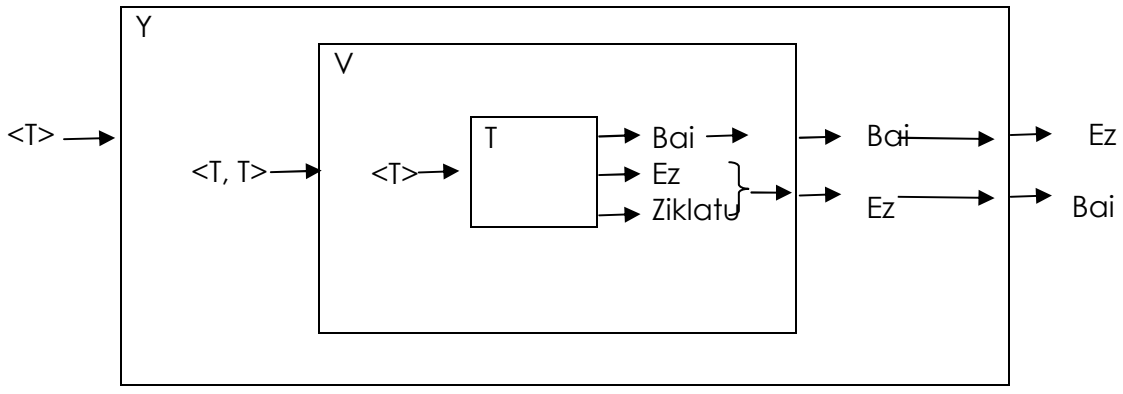
L_{bai} erabakigarria dela frogatzeko, L_{bai} lengoaiako hitzentzat, hau da, *T makinak w hitzarentzat "Bai" erantzuten du* baldintza betetzen duten <T, w> erako hitzentzat "Bai" erantzuten duen eta baldintza hori betetzen ez duten hitzentzat "Ez" erantzuten duen makina bat eraiki beharko genuke. Makina horrek ezingo luke ziklatu inoiz. Beraz aurreko atalean eraiki dugun U makinak ez digu balio L_{bai} erabakigarria dela frogatzeko, U makinak batzutan ziklatu egiten baitu.

Kontraesanaren teknika erabiliz L_{bai} ez dela erabakigarria frogatuko dugu. Horretarako erabakigarria dela suposatuko dugu eta *T makinak w hitzarentzat "Bai" erantzuten du* baldintza betetzen duten <T, w> erako hitzentzat "Bai" erantzuten duen eta baldintza hori betetzen ez duten hitzentzat "Ez" erantzuten duen eta inoiz ziklatzen ez duen V makina bat existitzen dela suposatuko dugu.



Beraz, V-ri <T, w> erako hitz bat ematen diogunean, hau da, T makina edo funtzio baten deskribapena edo definizioa eta T makina horri eman beharreko w datua ematen dizkiogunean, V makinak (edo funtzioak) T makina edo funtzioa exekutatzen du (T makina edo funtzioari deitzen dio) datu bezala w emanez. T makinak "Bai" erantzuten badu w hitzarentzat, orduan V makinak ere "Bai" erantzungo du <T, w> hitzarentzat (edo datuarentzat). T makinak "Ez" erantzuten badu edo ziklatu egiten badu w hitzarentzat, orduan V makinak "Ez" erantzungo du <T, w> hitzarentzat.

V makina hori existitzen bada, orduan honako Y makina ere eraiki dezakegu:



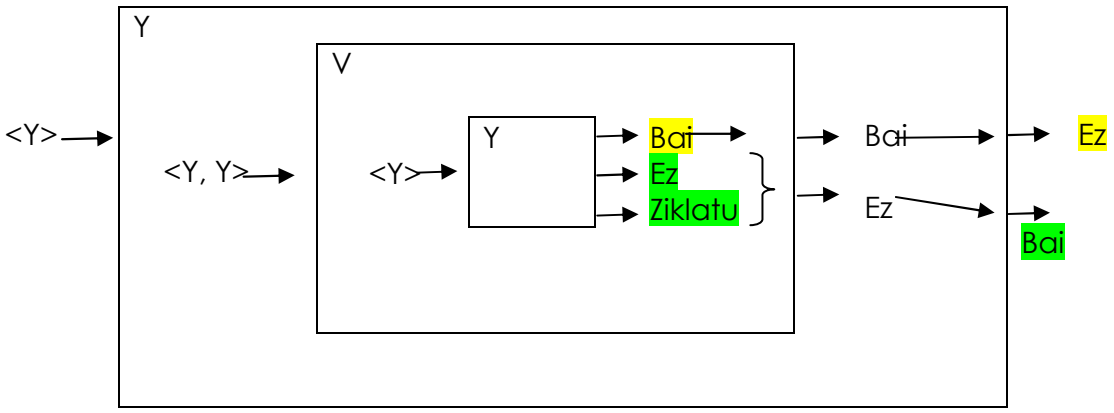
Y makinari T makina bat edo funtzio baten definizioa, hau da <T> pasatzen diogunean, Y makinak V makinari deitzen dio eta datu bezala T makinaren definizioa ematen dio, V-k T exekuta dezan, eta T exekutatzerakoan erabili beharreko datu bezala ere T bera pasatzen dio V-ri (w parametroaren lekua hartuz). Lehen azaldu den bezala, T makinak <T> hitzarentzat (hau da, T makina beraren definizioarentzat) "Bai" itzultzen badu, V-k "Bai" itzuliko du <T, T> hitzarentzat eta Y-k kontrakoa, hau da, "Ez" itzuliko du <T> hitzarentzat. T makinak <T> hitzarentzat (hau da, T makina beraren definizioarentzat) "Ez" itzultzen badu edo ziklatu egiten badu, V-k "Ez" itzuliko du <T, T> hitzarentzat eta Y-k kontrakoa, hau da, "Bai" itzuliko du <T> hitzarentzat.

Kontraesana noiz sortzen da? Y makinari datu bezala <Y> ematen diogunean, hau da, Y-ri Y beraren deskribapena edo definizioa ematen diogunean.

- Y makinari Y makinaren definizioa, hau da, <Y> pasatzen diogunean, Y makinak V makinari deitzen dio eta datu bezala Y makina beraren definizioa ematen dio, V-k Y exekuta dezan, eta Y exekutatzerakoan erabili beharreko datu bezala ere Y bera pasatzen dio V-ri (w parametroaren lekua hartuz).
- Lehen azaldu den bezala, Y makinak <Y> hitzarentzat (hau da, Y makina beraren definizioarentzat) "Bai" itzultzen badu, V-k ere "Bai" itzuliko du <Y, Y> hitzarentzat eta Y-k kontrakoa, hau da, Y makinak <Y> hitzarentzat "Ez" itzuliko du.

  KONTRAESANA: Y makinak <Y> hitzarentzat "Bai" itzultzen badu, Y makinak <Y> hitzarentzat "Ez" itzuliko du.
- Y makinak <Y> hitzarentzat (hau da, Y makina beraren definizioarentzat) "Ez" itzultzen badu edo ziklatu egiten badu, V-k "Ez" itzuliko du <Y, Y> hitzarentzat eta Y-k kontrakoa, hau da, Y makinak <Y>-rentzat "Bai" itzuliko du.

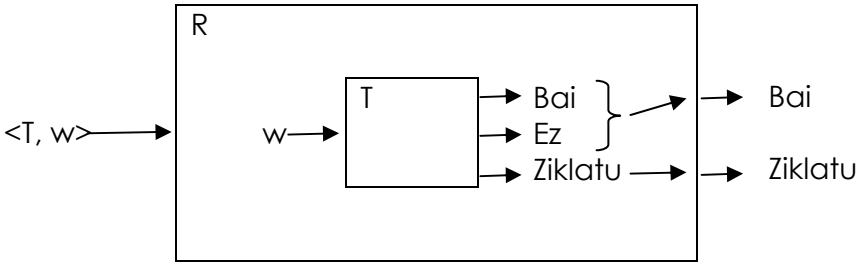
  KONTRAESANA: Y makinak <Y> hitzarentzat "Ez" itzultzen badu edo ziklatu egiten badu, Y makinak <Y> hitzarentzat "Bai" itzuliko du.



L_{halt} bereizgarria da

L_{halt} = { <T, w> | T makinak w hitzarentzat "Bai" edo "Ez" erantzuten du }

L_{halt} bereizgarria dela frogatzeko, L_{halt} lengoaiako hitzentzat, hau da, *T makinak w hitzarentzat "Bai" edo "Ez" erantzuten du* baldintza betetzen duten <T, w> erako hitzentzat (edo bikoteentzat) "Bai" erantzuten duen makina bat eraiki behar da.



R makinari (edo funtzioari) <T, w> hitza (edo bikotea) emango zaio, hau da, T makina edo funtzioa eta T makina edo funtzioarentzat datua den w elementua.

Jarraian T makina edo funtzioa w datuarekin exekutatuko da eta R funtzioak T funtzioak itzulitako emaitza jasoko du:

- T makinak edo funtzioak "Bai" itzultzen badu w hitzarentzat, orduan R funtzioak edo makinak ere "Bai" itzuliko du <T, w> hitzarentzat.
- T makinak edo funtzioak "Ez" itzultzen badu w hitzarentzat, orduan R funtzioak edo makinak "Bai" itzuliko du <T, w> hitzarentzat.
- T makinak edo funtzioak ziklatu egiten badu, hau da, erantzunik ez badu itzultzen w hitzarentzat, R makina erantzunaren zain geratuko da eta, ondorioz, R makinak edo funtzioak ere ziklatu egingo du <T, w> hitzarentzat.

Hala ere, T makinak w-rentzat "Bai" edo "Ez" erantzuten duenean R makinak <T, w> hitzarentzat "Bai" erantzungo duenez, L_{halt} lengoaia bereizgarria da, lengoaia horretakoak diren hitzentzat "Bai" erantzuten duen makina bat (R makina) existitzen delako.

Konturatu <T, w> hitza edo bikotea L_{halt} lengoaiakoa ez bada, hau da, T makinak w hitzarekin ziklatu egiten badu, orduan R makinak ere ziklatu egingo duela. Beraz, L_{halt} lengoaiakoak ez diren hitzentzat ez du "Ez" erantzungo, ziklatu egingo baitu.

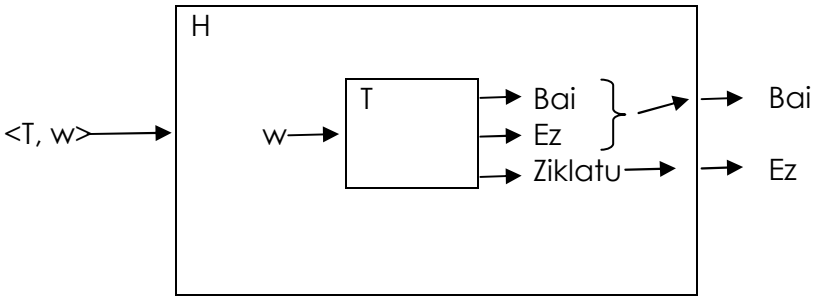
Beraz R makinaren bidez <T, w> erako hitzak edo bikoteak L_{halt} lengoaiakoak direneko kasuak ondo kontrolatzen dira baina <T, w> erako hitzak edo bikoteak L_{halt} lengoaiakoak ez direneko kasuak ez dira ondo kontrolatzen, izan ere <T, w> erako hitzak edo bikoteak L_{halt} lengoaiakoak ez direneko kasuetan ez baita erantzungo, ziklatu egingo da.

L_{halt} ez da erabakigarria (erabakiezina da)

L_{halt} = { <T, w> | T makinak w hitzarentzat "Bai" edo "Ez" erantzuten du }

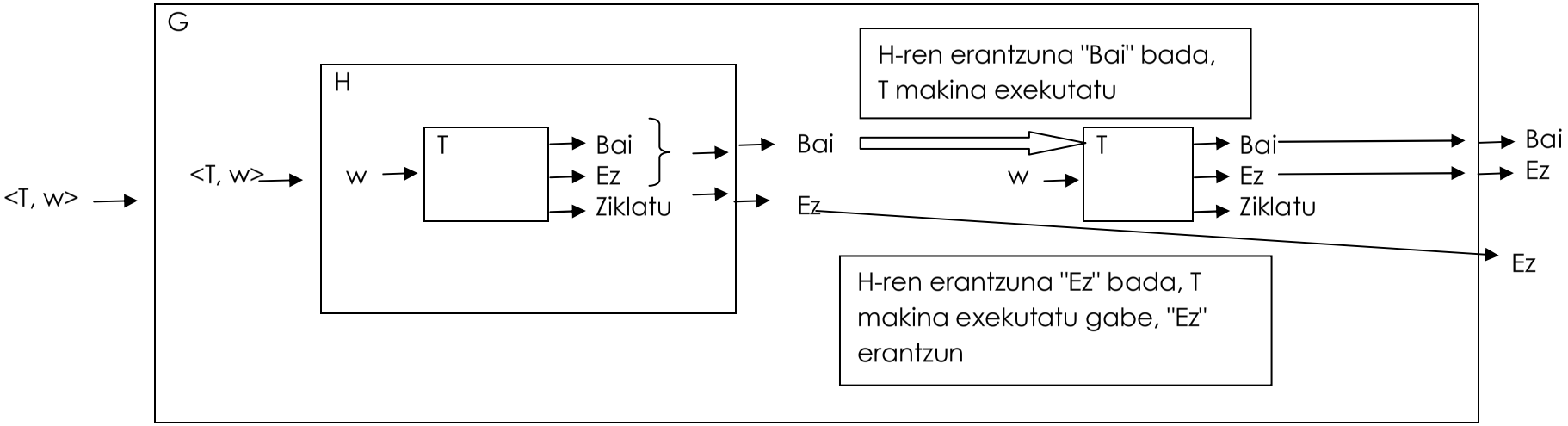
L_{halt} erabakigarria dela frogatzeko, L_{halt} lengoaiako hitzentzat, hau da, *T makinak w hitzarentzat "Bai" edo "Ez" erantzuten du* baldintza betetzen duten <T, w> erako hitzentzat "Bai" erantzuten duen eta baldintza hori betetzen ez duten hitzentzat "Ez" erantzuten duen makina bat eraiki beharko genuke. Makina horrek ezingo luke ziklatu inoiz.

Kontraesanaren teknika erabiliz L_{halt} ez dela erabakigarria frogatuko dugu. Horretarako erabakigarria dela suposatuko dugu eta *T makinak w hitzarentzat "Bai" edo "Ez" erantzuten du* baldintza betetzen duten <T, w> erako hitzentzat "Bai" erantzuten duen eta baldintza hori betetzen ez duten hitzentzat "Ez" erantzuten duen eta inoiz ziklatzen ez duen H makina bat existitzen dela suposatuko dugu.



Beraz, H-ri <T, w> erako hitz bat (edo bikote bat) ematen diogunean, hau da, T makina edo funtzio baten deskribapena edo definizioa eta T makina horri eman beharreko w datua ematen dizkiogunean, H makinak (edo funtzioak) T makina edo funtzioa exekutatzen du (T makina edo funtzioari deitzen dio) datu bezala w emanez. T makinak "Bai" erantzuten badu edo "Ez" erantzuten badu w hitzarentzat, orduan H makinak "Bai" erantzungo du <T, w> hitzarentzat (edo datuarentzat). T makinak ziklatu egiten badu w hitzarentzat, orduan H makinak "Ez" erantzungo du <T, w> hitzarentzat.

H makina hori existitzen bada, orduan honako G makina ere eraiki dezakegu:



G makinari T makina bat edo funtzio baten definizioa eta T makina horri eman beharreko w datua ematen dizkiogunean, hau da, <T, w> erako hitz bat edo bikote bat pasatzen diogunean, G makinak H makinari deitzen dio eta datu bezala <T, w> ematen dio H-ri, H makinak T exekuta dezan w datuarekin. Lehen azaldu den bezala, T makinak w hitzarentzat "Bai" edo "Ez" itzultzen badu, H makinak "Bai" itzuliko du <T, w> hitzarentzat eta kasu horretan G makinak badaki T makinak w hitzarentzat "Bai" edo "Ez" erantzungo duela eta berriro T exekutatuko du w hitzarekin. T-k w datuarekin "Bai" erantzuten badu G makinak ere "Bai" erantzungo du <T, w> hitzarentzat. T makinak w datuarekin "Ez" erantzuten badu G makinak ere "Ez" erantzungo du <T, w> hitzarentzat. Bestalde, H makinak "Ez" erantzun badu <T, w> hitzarentzat, orduan G makinak badaki T makinak w hitzarekin ziklatu egiten duela eta zuzenean "Ez" erantzungo du <T, w> hitzarentzat.

Kontraesana noiz sortzen da? G makinak L_{bai} lengoaia erabakitzeko balio du eta hori ezinezkoa da, L_{bai} lengoaia ez dela erabakigarria frogatuta baitaukagu.

KONTRAESANA: H makina existitzen bada, orduan L_{bai} lengoaia ere erabakigarria da, baina badakigu L_{bai} ez dela erabakigarria.

H makina existitzen bada kontraesana sortzen denez, H makinak ezin du existitu eta ondorioz L_{halt} ez da erabakigarria.

Lengoaia bereiztezinak badaude

Turing-en makinak azken batean funtzioak direla esan dugu (adibidez, Turing-en makina bat Haskell-eko funtzio bat bezala ikus dezakegu). Eta badakigu funtzio denak 0 eta 1 sinboloak bakarrik erabiliz kode daitezkeela. Beraz funtzioak A = {0, 1} alfabetoaren gaineko hitzak bezala ikus ditzakegu. Esate baterako 00011100001111010 1 funtzio bat izan daiteke.

A = {0, 1} izanda, A* multzoko hitzen bidez kodetu ditzakegu beraz funtzio denak.

Bestetik, badakigu datu denak ere 0 eta 1 sinboloak erabiliz kode daitezkeela. Beraz, existitzen diren lengoaia denak 2^{A^*} multzoan daude, A = {0, 1} izanda.

Turing-en makina bakoitzak 2^{A^*} multzoko lengoaia bat definitu dezake, hau da Turing-en makina batek ezin ditzake bi lengoaia desberdin definitu. Ondorioz, 2^{A^*} multzoko lengoaia bakoitzeko Turing-en makina bat beharko genuke.

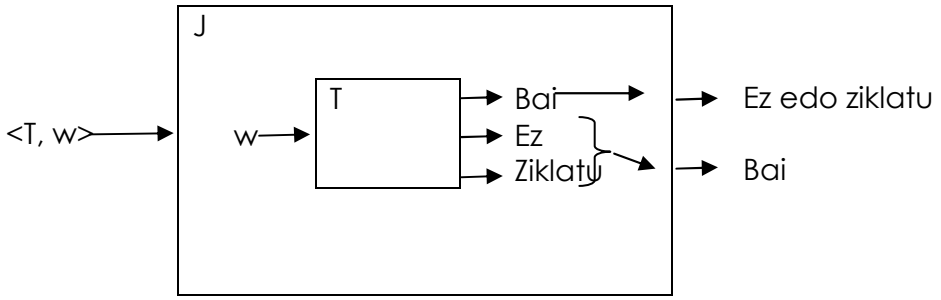
Baina badakigu A* zenbagarria dela eta 2^{A^*} multzoa ez dela zenbagarria, hau da, 2^{A^*} multzoan A* multzoan baino elementu gehiago daude. Beraz lengoaia-kopurua makina-kopurua baino handiagoa da eta ondorioz, lengoaia batzuentzat ez dago makinarik. Makinarik ez duten lengoaia horiek lengoaia bereiztezinak dira.

L_{bai} lengoaiaren osagarria $\overline{L_{bai}}$ ez da bereizgarria (bereiztezina da)

$\overline{L_{bai}}$ = { <T, w> | T makinak w hitzarentzat "Ez" erantzuten du edo ziklatu egiten du }

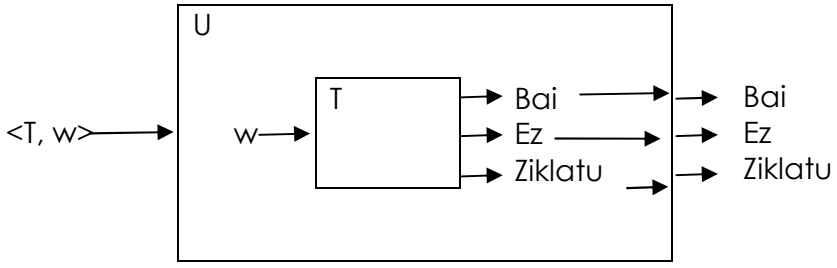
L_{bai} bereizgarria dela frogatzeko, $\overline{L_{bai}}$ lengoaiako hitzentzat, hau da, *T makinak w hitzarentzat "Ez" erantzuten du edo ziklatu egiten du* baldintza betetzen duten <T, w> erako hitzentzat "Bai" erantzuten duen makina bat eraiki beharko genuke (baldintza hori betetzen ez duten hitzentzat "Ez" erantzun lezake edo ziklatu egin lezake).

Kontraesanaren teknika erabiliz $\overline{L_{bai}}$ ez dela bereizgarria frogatuko dugu. Horretarako bereizgarria dela suposatuko dugu eta *T makinak w hitzarentzat "Ez" erantzuten du edo ziklatu egiten du* baldintza betetzen duten <T, w> erako hitzentzat "Bai" erantzuten duen J makina bat existitzen dela suposatuko dugu.

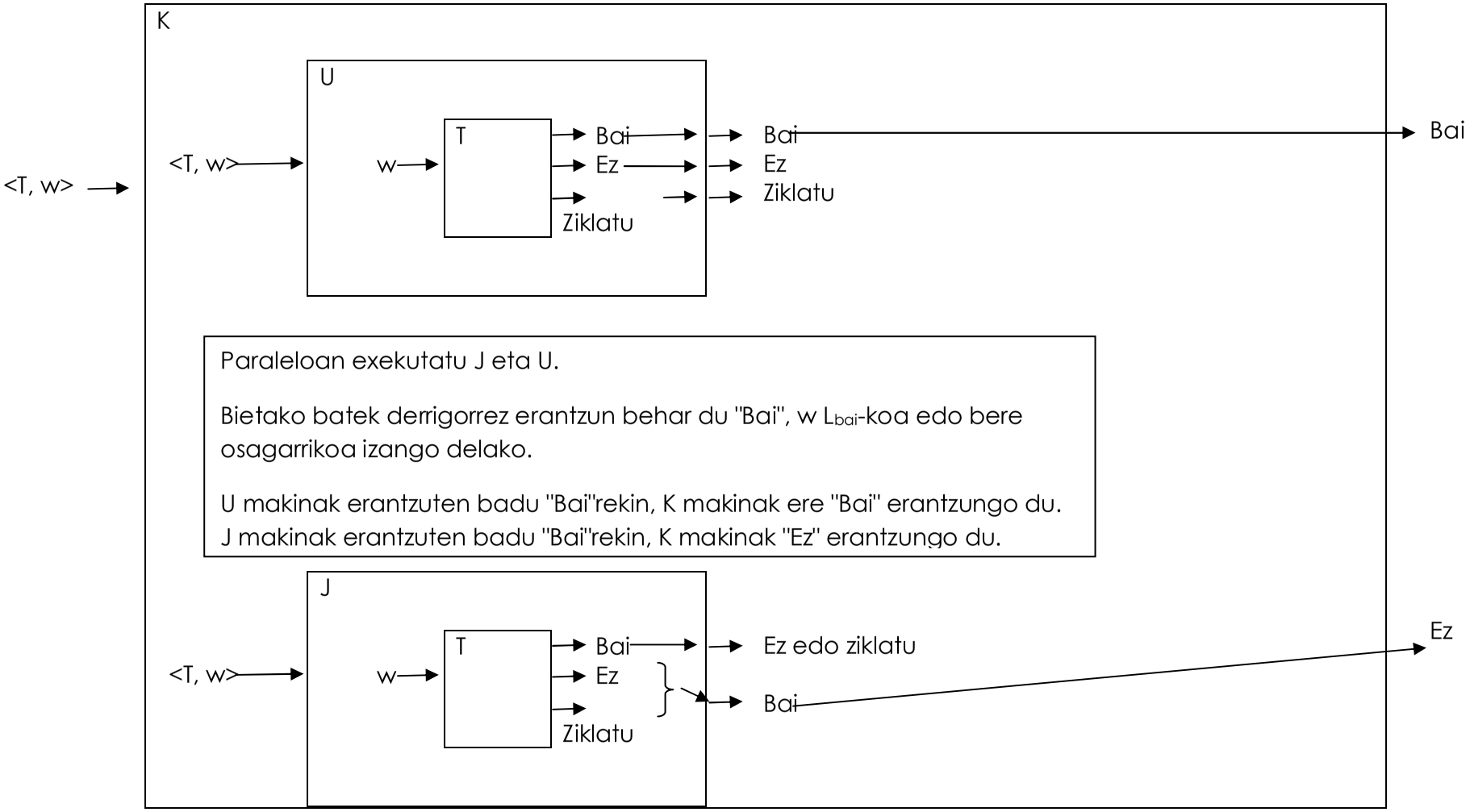


Beraz, J-ri <T, w> erako hitz bat ematen diogunean, hau da, T makina edo funtzio baten deskribapena edo definizioa eta T makina horri eman beharreko w datua ematen dizkiogunean, J makinak (edo funtzioak) T makina edo funtzioa exekutatzen du (T makina edo funtzioari deitzen dio) datu bezala w emanez. T makinak "Ez" erantzuten badu w hitzarentzat edo ziklatu egiten badu w hitzarekin, orduan J makinak "Bai" erantzungo du <T, w> hitzarentzat (edo datuarentzat). T makinak "Bai" erantzuten badu, berdin zaigu J makinak zer egiten duen <T, w> hitzarentzat ("Ez" erantzun edo ziklatu, J makinak L_{bai} lengoaiaren osagarria bereizteko balio behar duelako eta ez dauka L_{bai} lengoaiaren osagarria erabakitzeko balio beharrik).

Bestetik badakigu L_{bai} bereizgarria dela, L_{bai} lengoaiako hitzentzat, hau da, *T makinak w hitzarentzat "Bai" erantzuten du* baldintza betetzen duten <T, w> erako hitzentzat "Bai" erantzuten duen makina bat eraiki dugu lehen.



J makina hori existitzen bada, orduan honako K makina ere eraiki dezakegu:



K makinari T makina bat edo funtzio baten definizioa eta T makina horri eman beharreko w datua ematen dizkiogunean, hau da, <T, w> erako hitz bat pasatzen diogunean, K makinak J eta U makinei deitzen die PARALELOAN eta datu bezala <T, w> ematen die, T exekuta dezaten w datuarekin. J makinak edo U makinak derrigorrez erantzun behar du "Bai" w hitza L_{bai}-koa edo L_{bai}-ren osagarrikoa delako derrigorrez. w hitza L_{bai}-koa bada U makinak "Bai" erantzungo du eta w hitza L_{bai}-ren osagarrikoa bada, J makinak "Bai" erantzungo du.

Lehen azaldu den bezala, T makinak w hitzarentzat "Bai" itzultzen badu, U makinak "Bai" itzuliko du <T, w> hitzarentzat eta kasu horretan K makinak badaki T makinak w hitzarentzat "Bai" erantzungo duela eta K makinak berak "Bai" erantzungo du. Lehen azaldu den bezala, T makinak w hitzarentzat "Ez" itzultzen badu edo ziklatu egiten badu, J makinak "Bai" itzuliko du <T, w> hitzarentzat eta kasu horretan K makinak badaki T makinak w hitzarentzat "Ez" erantzungo duela edo ziklatu egingo duela eta K makinak berak "Ez" erantzungo du. Beraz K makinaren bidez L_{bai} lengoaia erabakigarria izango litzateke, baina badakigu L_{bai} lengoaia ez dela erabakigarria, beraz J existitzen dela suposatuz KONTRAESANA sortu denez, J makinak ezin du existitru, eta L_{bai} lengoaiaren osagarria ez da bereizgarria.

Turing

```
module Turing where

--Jarraian dauden adibide sinple hauen bidez Haskell-en
--Turing-en makinak funtzio bezala nola defini daitezkeen
--erakusten da.
--u_t_m, u_t_m2 eta u_t_m3 funtzioei datu bezala beste funtzio
--bat pasa diezaiekegu, Turing-en makinei beste makina baten
--kodea pasatzen diezaiekegun bezala.


f:: t -> Bool
f e = True

g:: Int -> Int
g x = 0

h:: Int -> Bool
h x
        |x > 0 = True
        |x < 0 = False
        |x == 0 = h x

u_t_m:: t1 -> t2 -> Bool
u_t_m m w = m w
--m w deiak ziklatu egiten badu u_t_m funtzioak ere ziklatu egingo du
--t1 mota ipintzean, hor edozer ipini dezakegula adierazten da. Funtzio
--bat ere ipini dezakegu.
--Adibidez u_t_m h 4
--Beraz lehenengo parametroa funtzio bat izan daiteke (hau da,
--Turing-en makina bat izan daiteke eta bigarren parametroa
--makina horri edo funtzio horri eman nahi diogun datua
--u_t_m h 4 deiak h funtzioa exekutatzen du 4 datuarekin.
--Nolabait, u_t_m funtzioak ordenagailu bat bezala jokatzen du.


u_t_m2:: t1 -> t2 -> Bool
u_t_m2 m w
        |m w == True = True
        |m w == False = False
--m w deiak ziklatu egiten badu u_t_m2 funtzioak ere ziklatu egingo du

--Orokortuz
u_t_m3:: t1 -> t2 -> t3
u_t_m3 m w = m w

--
luzera:: [t] -> Int
luzera z = length z

--Orain honako deia egin dezakegu: u_t_m3 luzera [True, False, True]
--Baina beste dei hau ere egin dezakegu: u_t_m3 luzera [5, 6, 2]

--
karratua:: Int -> Int
karratua x = x * x

--Orain honako deia egin dezakegu: u_t_m3 karratua 6
```