(19) LIETUVOS RESPUBLIKA
VALSTYBINIS PATENTŲ BIURAS
V P B

(10) **LT IP379 A**

(12) **PARAIŠKOS APRAŠYMAS**

(21) Paraiškos numeris: **IP379**

(22) Paraiškos padavimo data: **1993 03 05**

(41) Paraiškos paskelbimo data: **1994 11 25**

(62) Paraiškos, iš kurios dokumentas išskirtas, numeris: —

(86) Tarptautinės paraiškos numeris: —

(86) Tarptautinės paraiškos padavimo data: —

(85) Nacionalinio PCT lygio procedūros pradžios data: —

(30) Prioritetas: **9002558, 1990 08 02, SE**
PCT/SE91/00517, 1991 08 01, SE

(51) Int. Cl. (2006): **G06F 7/00**
G06F 7/38
G06F 7/48
G06F 9/302
G06F 9/44
G06F 12/00
G11C 11/412
G11C 11/417
G11C 11/419
G11C 15/00

(71) Pareiškėjas:
CARLSTEDT ELEKTRONIK AB, Industrivagen 55, S-433 61 Partille, SE

(72) Išradėjas:
Lars Gunnar CARLSTEDT, SE

(74) Patentinis patikėtinis/atstovas:
Reda ŽABOLIENĖ, Advokatės Redos Žabolienės kontora METIDA, Verslo centras VERTAS, Gynėjų g. 16, LT-01109 Vilnius, LT

(54) Pavadinimas:

Aktyvios atminties įrenginys redukcijos procesoriuje

(57) Referatas:

—

LT IP379 A

SUSKAIDANČIO (REDUKCINIO) PROCESORIAUS AKTYVI ATMINTIS

IŠRADIMO PAGRINDAI

Kompiuteriai buvo išrasti 1940-jų metų laikotarpyje. Nuo to meto jie buvo vystomi revoliuciniu tempu. Nepaisant to, šiuo metu kompiuteriai turi tą pačią architektūrą, kaip ir pirmieji.

Didžiausi patobulinimai buvo padaryti aparatinėje dalyje. VLSI įgyvendinimas ir pasiekimai litografijos srityje įgalino sukurti kompiuterius, susidedančius iš vienos mikroschemos. Tai tokie kompiuteriai, kurie tik prieš penketą metų buvo vadinami superkompiuteriais. Linijų (mikroschemose) plotis siaurėjo eksponentiniu dėsniu ir dabar jis yra mažesnis, negu 1 mikrometras. Naudojamų impulsų dažnio dydis, o taip pat ir aktyvių tranzistorių galingumo diapazonas padidėjo daug kartų. Matyt, linijų plotis bus fiziškai ribojamas iki 0,2 mikrometrų dydžio.

Tuo pačiu metu kompiuterių architektūros kūrėjai nedaug pasiekė silicio panaudojimo srityje. Priešingai, dauguma kompiuterių naudoja daugiau, negu optimalų silicio kiekį, siekiant didesnio jų greitaeigingumo.

Abu šie faktai stabdo vieno atskiro procesoriaus greitaeigingumo vystymą bent artimiausių penkių metų laikotarpiui. Lygiagretieji procesoriai yra naudojami dėl didėjančios aparatinės dalies kainos ir dėl didėjančio sudėtingumo bei daugeliui programų tipų didėjančios programavimo kainos.

Peržvelgiant į jų tarpusavio santykį, aparatinės dalies kaina pamažėjo, tačiau naujų sistemų programavimo kaina pastebimai išaugo ir greitai taps neleistinai didelė.

Kompiuteris yra sudėtingas įvairių blokų įrenginys, tiek aparatine dalimi, tiek ir programine dalimi. Skirtingi pavyzdžiai ir evoliucijos etapai skatino kurti specialius standartus, kurie buvo diegiami sistemose. Dėl šio nesuvienodinimo atsirado daug interfeisų.

Visi šie interfeisai yra pavyzdys, kaip skirtinga kokybė ir stilius kelia keblumus vartotojui arba progrmuotojui, besinaudojančiam mašina - ji reikalauja daug žinių, o dėl jos sudėtingumo programuotojas gali padaryti nepastebimų klaidų.

Pastaruoju metu atsirado ir yra tobulinami taip vadinamieji suskaidantieji (redukciniai) procesoriai. Suskaidantieji procesoriai atlieka programą, turinčią tam tikrą struktūrą, į kurią įeina aritmetinės lygtys, o ši struktūra yra susmulkinama į eilę sumažintų žingsnių. Tokiu būdu, programa nėra vykdoma pagal nustatytą eilės tvarką, kaip kitų rūšių kompiuteriuose. Sukuriant nustatyto dydžio suskaidančius procesorius, buvo susidurima su tam tikrais sunkumais.

Programavimo kalbų paruošimas

Pirmo elektroninio kompiuterio sukūrimas davė pradžią kelių programavimo kalbų sudarymui, kurios tiktų šio tipo kompiuteriams, tokių, kaip FORTRAN, COBOL, ALGOL, BASIC, Pascal. Šios kalbos buvo pavadintos imperatyvinėmis kalbomis, greta taip vadinamų įprastinių kalbų, daugiausia dėl to fakto, kad jos paprastai pateikia programą, susidedančią iš eilės komandų arba nurodymų, kurie turi būti vykdomi eilės tvarka įprastiniu kompiuteriu, pvz., kompiuteriu, sukurtu pagal John von Neumann principus. Didėjantis diskomfortas, naudojantis šiomis kalbomis, privertė išvystyti kitą kalbų seriją: LISP, ISWIM, Scheme (LISP dialektas), ML, Hope, SASL ir tt. Šių kalbų sukūrimą skatino jų schematiškas paprastumas. Nei viena iš konkrečių mašinų įtakos jų kūrimui neturėjo.

Praėjo kiek laiko, kol funkcinės kalbos pradėjo traukti dėmesį. Viena iš priežasčių buvo tai, kad funkcinės programos buvo lėtai vykdomos kompiuteryje. Vėlesni darbai parodė, kad tam tikrais atvejais programų vykdymo greitis gali būti artimas arba toks pats, kaip ir įprastinių kalbų programų, atliekamų įprastiniais kompiuteriais, nors funkcinės programos ir nėra skirtos šio tipo kompiuteriams.

Programinės įrangos krizė

Didėjantis diskomfortas, naudojantis privalomojo tipo kalbomis, plačiu mastu paskatino dėti pastangas sukurti funkcines kalbas. Kalba apie programinės įrangos krizę prasidėjo apie 1970m. Programos darėsi vis sudėtingesnės ir dažnai turėdavo daugybę klaidų, buvo sunkiai įvedamos ir ypač sunkiai tobulinamos. Viena iš tų keblumų priežasčių yra tai, kad buvo dedami per dideliu lūkesčiai, jog aukšto lygio imperatyvinės kalbos turėtų supaprastinti programavimą. Šios kalbos nėra tokio aukšto lygio, kaip jos gali atrodyti. Privalomosios kalbos vis dar yra adaptuojamos pagal ankstyvųjų kompiuterių koncepciją - von Neumann tipo kompiuterių koncepciją, ir programavimo lygis vis dar yra artimas aparato lygiui. Funkcinio programavimo kalbos turi kai kurias savybes, kurios mažina labiausiai įprastus programavimo kalbų nepatogumus.

Papildoma informacija ir paaiškinimai gali būti randami knygoje "Functional Programming Using Standard ML", Ake Wikstrom, Prentice Hall 1987.

Tuo tikslu, kad būtų pilnai suprantami išradimo tikslai ir privalumai, yra svarbu suprasti, iš ko susideda funkcinis požiūris kompiuteriniame darbe. Istoriškai lyginant, labiausiai tiktų imperatyvinis požiūris.

Išsireiškimas "funkcinis požiūris" reiškia, kad programos yra parašytos funkcine kalba, įvedamos į atmintį ir atliekamos kompiuteryje, parenkant aparatinės dalies sudėtį, tinkamą tokiai kalbai. Taip pat, išsireiškime "imperatyvinis požiūris" turima galvoje, kad programa yra parašyta imperatyvine (privalomąja) kalba, įvedama į atmintį ir atliekama kompiuteryje, parenkant aparatinės dalie sudėtį, tinkamą šiai kalbai.

Tačiau yra įmanoma įvesti ir atlikti įprastiniu kompiuteriu programas, parašytas funkcine kalba. Priešingas atvejis yra taip

pat įmanomas - programos, parašytos privalomomis kalbomis gali būti atliekamos kompiuteriu, skirtu vykdyti programas, parašytas funkcinėmis kalbomis.

Funkcinių kalbų savybės

Į programą, parašytą funkcine kalba, gali būti žiūrima, kaip į rinkinį objekto savybių apibrėžimų, ir kaip į kompiuterines taisykles. Apibrėžimai yra deklaratyvioji dalis, o kompiuterinio darbo taisyklės, arba suskaidymas į dalis, arba perrašymo taisyklės sudaro operatyvinę dalį, kurią naudoja kompiuteris darbo metu. Funkcinės kalbos sudaro aukštesnio lygio interfeisą kompiuteryje, kas įgalina programuotoją nepaisyti kompiuterio aparatinės dalies sudėties. Teigiamu papildomu efektu yra tai, kad funkcinės programos dažnai yra trumpesnės ir jas lengviau suprasti, negu įprastines privalomasias programas. Vienas iš pagrindinių funkcinių kalbų trūkumų yra tai, kad funkcinės programos turi būti transliuojamas į įprastines kalbas tam tikslui, kad jas galima būtų atlikti įprastiniais kompiuteriais. Tai yra atliekama kompiliatorių arba interpretuojančių programų pagalba. Aišku, kad funkcinio požiūrio naudingumas yra šiek tiek menkinamas to fakto, kad apartinė dalis neturi priskirtos dalies funkcinių programų įvedimui ir jų atlikimui kuo efektyviau.

APIBRĖŽIMAI

Žemiau yra pateikiamas išsireiškimų sarašas, naudojamas šiame aprašyme, ir jų reikšmės:

element elementas	šiek tiek didesnė dalis duomenų struktūroje;
list sąrašas	sutvarkyta elementų eilė, kurioje kiekvienas elementas gali būti sąrašu;
inserted list įterptas sąrašas	sąrašo dalis, per maža saugojimui viename uždarame reiškinyje. Ji pati gali at-

stovauti palyginti ilgus sarašus;

closure uždaras reiškinys	hierarchinės struktūros kategorija, apsakanti procesa. Visos šios kategorijos turi šaknį, kuri unikaliai apibūdina uždaraji reiškinį. Darbas suskaidančiu kompiuteriu yra atliekamas uždarų reiškiniu pagalba. Visas įrenginio būvis yra transformuojamas suskaidymais;
object storage objektinė atmintis	atmintis, įskaitant lasteles objektu saugojimui. Pavyzdžiui, asociatyvi atmintis;
storage cell atminties lastelė	lastelė, skirta objektų saugojimui. Ji Ji saugo atmintyje uždarų reiškiniu lasteles, kurios gali turėti ryšį su kitomis uždarų reiškiniu lastelėmis kitose atminties lastelėse;
cell closure uždara lastelė	atminties lastelės turinys;
storage cell field atminties lastelės laukas	laukas, esantis atminties lastelėje;
closure element uždaras elementas	duomenų elementas, saugomas atminties lastelės lauke;
closure identifier uždaro reiškinio identifikatorius	uždaro reiškinio elementas, vienareikšmiškai jį pažymintis;
canonical closure kanoninis uždaras reiškinys	uždaras reiškinys, kuris negali būti daugiau suskaidytas, pvz., lastelė, kuri neturi daugiau uždaro reiškinio identifikatoriu ženklu, kokia nors kitas uždaras lasteles, kurios gali būti suskaidytos tokiu būdu, kad ši uždara lastelė galetų būti dar suskaidyta;

goal tikslas	apdorojamas (vykdomas) uždaras reiškinys, pvz., skaidomas;
father tėvas	uždaras reiškinys, turintis ne mažiau vieno identifikatoriaus savo vertės/žymėjimo lauke;
son sūnus	uždaras reiškinys, susijęs su kitu reiškiniu per jo identifikatorių, kuris žymi tą sūnų;

Sūnus taip pat gali būti ir tėvu. Tėvas taip pat gali būti sūnumi. Sūnus gali turėti daugiau, negu vieną tėvą. Tėvas gali turėti daugiau, negu vieną sūnų.

Closure position Uždaro reiškinio padėtis	rodo, ar reiškinys yra šaknis, ar susikirtimo (tinklelio) taškas;
root šaknis	pati viršutinė uždaro reiškinio lastelė, esanti reiškinio medyje;
node susikirtimo taškas	uždaro reiškinio lastelė, esanti reiškinio medyje ir nesanti šaknimi;
where kur	atminties lastelės laukas, užimantis uždaro reiškinio padėtį;
type tipas	tipo kodas, esantis uždaroje lasteleje, pvz., bito pavidalas, rodantis objekto savybę, pvz., instrukcijos kodas;
lazy tingus	uždaros lastelės elementas, kuris rodo, ar galima su ja operuoti (vykdyti), ar tai reikia atidėti, arba ji yra neveikianti;
identifier identifikatorius	specialus uždaro elemento tipas, naudojamas saugomo atmintyje objekto pažymėjimui;
environment aplinka (periferija)	objektai gali būti grupuojami, suteikiant jiems tą pačią aplinką;

value/des. reikšmh/ žymėjimas	uždaras elementas, kuriame yra saugoma arba paprasta reikšmė, pvz., tiesioginis priskyrimas, arba nieko, arba žymėjimas, rodantis priklausymą kitam uždaram reiškiniui, pvz., netiesioginis priskyrimas;
core cell centrinė lastelė	aritmetinės struktūros vienetas pagal šį išradimą. Centrinė lastelė gali atlikti aritmetinius veiksmus, įskaitant uždarų struktūrų skaidymą;
numeric ALU skaitmeninis loginis blokas	skaitmeninis aritmetinis blokas, galintis atlikti bazines skaitmenines arba logines operacijas. Centrinė lastelė naudojasi skaitmeniniu ALU, atliekant skaitmenines operacijas;
full register pilnas registras	registro praplėtimas visame centrinės lastelės plote;
core word centrinis žodis	pilno registro turinys, esantis centrinėje lastelėje;
limited register ribotas registras	registras, esantis centrinės lastelės pratęsime, esančiame riboto ploto apimtyje, turintis apimtį, pakankamą priimti uždarą vertės/žymėjimo tipo lastelę;
element word elementarus žodis	riboto registro turinys arba pilna registro dalis, turinti tą patį praplėtimą, kaip ir ribotas registras;
num word skaitmeninis žodis	elementaraus žodžio dalis, rodantis reikšmę arba žymėjimą;
tag word požymio žodis	elementaraus žodžio dalis, turinti etikete, rodančia reprezentacijos požymį skaitmeniniame žodyje;

reduction skaidymas (sumažinimas)	uždaro reiškinio perrašymas / performavimas pagal tam tikros naudojamos programavimo kalbos taisykles;
behavior režimas	laiko impulsų struktūros elementas, toks kaip eilės tvarka arba eilių keitimas ties porto įėjimu. Taip pat naudojamas ir pakeistos formos, analogiškai, kaip yra naudojama matematikoje , eilučių teorijoje, pavyzdžiui, "1" , kuris iš principo gali būti režimo požymiu.

IŠRADIMO TIKSLAI

Išradimo tikslas yra pateikti skaičiavimo prietaisą, turintį aprašomąją programavimo kalbą.

Kitas išradimo tikslas yra pateikti skaičiavimo prietaisą, dirbantį, kaip skaidantysis procesorius.

Dar kitas išradimo tikslas yra pateikti skaičiavimo prietaisą, turintį programavimo kalbą, kuri būtų paprasta ir kuri leistų programuotojui atsitraukti nuo visų susijusių su aparatine dalimi detalių ir kuris įgalintų sudaryti programas, užimančias mažiau laiko jų parašymui ir darant mažiau klaidų.

Kitas išradimo tikslas yra pateikti skaičiavimo prietaisą, turintį vieną kalbą, kaip programavimo kalbą, operacinę sistemą ir komunikacinį protokolą.

Dar kitas išradimo tikslas yra pateikti skaičiavimo prietaisą, galintį operuoti su funkcine kalba, surišta bendru unifikavimu.

Aukščiau minėti tikslai iš esmės yra pasiekiami, sukonstravus skaičiavimo prietaisą, turintį:

a) aktyvios atminties priemones, įskaitant daugelį atminties langelių, kiekviena iš kurių turi informaciją, galinčią duoti pradžią operacijos atlikimui;

b) ne mažiau, kaip vieną porto įtaisą, sujungtą su minėta aktyvia atminties priemone, ir

c) ne mažiau vienos aplinkos (periferijos) priemonės, sujungtos su minėtu, ne mažiau, kaip vienu portu.

Pageidautina, kad prietaisas turėtų priemones, esančias porto įtaise, signalo eilės tvarkos lyginimui su eile, esančia atmintyje ne mažiau, kaip vienoje atminties langelėje, esanti eilės tvarka atmintyje turinti nepažymėtus eilės tvarkos elementus, taip pat ir aktyvios atminties priemonėse ir priemones perrašyti lyginamą eilės tvarką į nieką, pvz., ko nors, kas neatitinka - kai lyginimas parodo aišku skirtumą arba perrašymui minėtos lyginamos eilės į nurodytą eilę.

Lyginimo priemonės gali atlikti sulyginimą grupėmis ne mariau, kaip po du elementus iš nustatyto skaičiaus sąrašo elementų.

Signalo eilės tvarka yra pageidautina, kad būtų pateikiama, kaip atrinktas signalas, turintis laiko impulsus su individualiais periodais, signalo eilės tvarka būtų sarašas iš elementų grupių, kiekviena iš grupių turėtų trtukmės laiką ir ne mažiau kaip vieną signalą tame laikotarpyje. Nustatytas skaičius sąrašo elementų kiekvienoje grupėje yra tinkamas po du poromis, o kiekviena pora turinti laiko ir signalų kiekio kombinaciją.

Priemonės gali būti pateikiamos vieno iš minėtų porto proiemonių, sinchronizavimui su ne mažiau, kaip kitu minėtų porto priemonių ir atlikimui lygiagretaus signalų kiekio indikavimo iš minėtų portų kiekvienos laiko trukmės metu. Tuo įvairūs signalų kiekiai gali būti iššaukiami skirtinguose portuose tuo pačiu metu ir juos galima lyginti vieną su kitu.

Atminties lastelių priemonės yra pageidautina pateikti pačioje aktyvios atminties struktūroje, kuri adaptuota saugoti kompiuterinę programą pilno ir nepilno kodavimo pavidalu su abstrakčia sintakse, kuri aprašo eilę skirtingų abstrakčių objektų lygčių pagalba, kiekviena atminties lasteles priemonė gali įvesti į atmintį ne mažiau, kaip dalį vienos minėtos sintaksinės lygties vienu metu atitinkamų duomenų pavidalu ir/arba programos struktūros pavidalu.

Kiekviena lygties rūšis gali taip pat turėti atitinkamą lygties rūšį, rodančią, kad tai yra programos forma, visos programų formos yra išraiškos, kurios yra pritaikytos suskaidymui tarpusavyje ir tokiu būdu esančios. tai daro įmanoma parašyti interpretacijas ir kompiliatorius kitoms kalboms bet kurios norimos rūšies.

Perrašymas reiškia bendravimą su tomis atminties lastelėmis, įskaitant sintaksės išraiškų perrašymą sintaksės išraiškomis pagal nustatytas perrašymo taisykles.

IŠRADIMO REZIUMĖ

Skaičiavimo prietaisas pagl šį išradimą yra kompiuteris, turintis aprašomąją programavimo kalbą, esančią aparatinėje dalyje, šiuo atveju programavimo kalba yra ir mašininė kalba. Tačiau, kaip ir kiekviename šiuolaikiniame elektroniniame prietaise, skaičiavimo prietaiso pagal šį išradimą sujungimai gali būti imituojami bendros paskirties kompiuteryje su interpretatoriumi arba kompiliatoriumi. Tokių mašinų pavyzdžiai yra įrenginiai, sukurti M68000 procesoriaus pagrindu, pavyzdžiui, Sun3. Tačiau pagal šį išradimą mašina turi portą, kuris turi unifikavimo galimybes.

Programų vykdymas suskaidymo būdu

Norima atlikti programa gali būti pateikiama tiesioginiu uždarų reiškinių grafu, kur kiekviena programos dalis yra atstovaujama uždaru reiškiniu. Vykdymo metu šis tiesioginis uždarų reiškinių grafas yra laipsniškai skaidomas atitinkamai pagal naudojamos kalbos suskaidymo taisykles. Kai nebelieka vykdymui uždarų reiškinių, programos vykdymas yra baigtas. Tiesioginiai uždarų reiškinių grafai gali būti laikomi medžio struktūra, kur kiekvienas susikertantis medžio elementas yra uždaras reiškinys ir kur pačioje viršūnėje esantis susikirtimo taškas (elementas) yra vadinamas šaknimi. Programos vykdymas suskaidymo būdu toliau atliekamas įprastu būdu, skaidant medžio struktūrą iš apačios į viršų, suskaidant medžio dalis, esančias toliausiai nuo šaknies ir einant šaknies link. Šis programos vykdymo būdas yra paprastai vadinamas vykdymu, paleidžiamu nuo užklausimo, t.y., programos dalių vykdymas priklausantis nuo kitų dalių įvykdymo rezultato yra atidedamas tol, kol gaunamas tas rezultatas.

TRUMPAS BRĖŽINIŲ APRAŠYMAS

Šio išradimo pilnam suprtimui ir tolesniam tikslų ir pranašumų paaiškinimui yra duodamos nuorodos į sekančius aprašymus prie pridedamų brėžinių, kur:

FIG. 1 yra principinė skaičiavimo prietaiso pagal šį išradimą schema;

FIG. 2 yra signlų eilės schema prie porto;

FIG. 3 rodo principinę skaičiavimo prietaiso įgyvendinimo blok - schemą pagal šį išradimą;

FIG. 4 rodo skirtingų atminties lastelių laukų panaudojimą objektų įvedimui į atmintį pagal prietaisą , parodytą FIG. 3.

FIG. 5 rodo pavyzdį, kaip gali būti įvesta į atminties lasteles funkcija, įvedant objektą į atmintį;

FIG. 6 rodo porto įmontavimo į aplinką schemą;

FIG. 7 rodo taip vadinamo H-porto schemą;

FIG. 8A, 8B, 8C rodo skirtingus registrų tipus struktūriniame aritmetiniame bloke pagal skaičiavimo įrenginį, parodytą FIG. 3;

FIG. 8D rodo struktūrinio aritmetinio įrenginio įgyvendinimo schemą;

FIG. 9A, iki 9F rodo įvairias duomenų įvedimo į atmintį formas struktūriniame aritmetiniame bloke;

FIG. 10A iki 10H,
11A iki 11G,
12A iki 12H rodo skaičiavimo įrenginio pagal šį išradimą darbo pavyzdį.

SIŪLOMO ĮGYVENDINIMO APRAŠYMAS

Skaičiavimo įrenginys

Skaičiavimo įrenginys pagal šį išradimą, kurio principinė schema yra parodyta FIG. 1, turintis aktyviąją atmintį U1, žemiau vadinamą aktyviąja objektų atmintimi. Ji yra aktyvi, nes į ją įeina atminties ląstelės, kiekviena iš kurių turi galimybę pradėti skaičiavimo operacijų vykdymą. Siūloma, kad kiekviena objekto atminties ląstelė būtų padalinta į keletą atminties laukų, o susieta paieška objekto atmintyje būtų atliekama turinyje, esančiame atminties laukuose. Atminties ląstelė gali saugoti uždarą ląstelę. Uždara ląstelė gali atmintyje turėti kitų uždarų ląstelių pažymėjimus, kaip uždaro reiškinio elementus. Funkcija gali būti įvesta į atmintį, kaip uždaras medis, į kurį įeina keletas uždarų ląstelių, žyminčių viena kitą. Žemiau pateikiama tolesnis šių savybių aprašymas.

Objekto įvedimas į atmintį sudaro vidinį skaičiavimo įrenginio pagal šį išradimą režimą. Įvedant objektą į atmintį yra įvedami skirtingose vietose keli vidiniai režimai PR1, PR2,...,PRn kiekvienam portui. Portų skaičius yra įjungtas į aktyviąją objekto atmintį U1. Įėjimo į portus signalai RW_{U2} iki RW_{Us}, einantieji į portus U2 gali būti paduodami iš sensorių arba iš informacijos, gaunamos iš prietaisų, esančių už skaičiavimo įrenginio ribų, pvz., relių žodžių pavidalu.

Režimai

Režimas yra bazinių elementų struktūra, kuri gali kisti laiko bėgyje. Tokiu būdu, režimas yra laiko kategorija, tačiau jis nenurodo, kodėl struktūra keičiasi. Laiko struktūra gali būti laikoma, kaip kuriamo proceso būvis. Skaičiavimo įrenginys pagal šį išradimą yra ypatingai pritaikytas aprašomajam programavimo kalbos tipui, žemiau vadinamam H. Ji neturi tikslių būvio ir perėjimo parametrų, kaip privalomosios kalbos. Vietoj to, visi

režimai yra charakterizuojami lygtimis. Terminas "režimas" yra tas pats, kaip ir struktūra laike, eilutė arba eilučių rinkinys. Jis taip pat yra naudojamas pakeista forma visų galimų reikšmių atvejui, palyginkite su tuščios eilutės savoka matematinėje eilučių teorijoje.

Portas transformuoja įeinantįjį signalą į formą, tinkamą aktyviai objektų atminčiai, pavyzdžiui, skaitmeninę formą, ir ją perduoda tokia forma, kuri gali būti sulyginama su informacija, esančia aktyvioje objekto atmintyje. Ši informacija, lokalinė konkrečiam portui, turi vieną arba keletą alternatyvių režimų PR1, PR2,..., PRn.

Portas Ui, turinti numerį tarp2 ir 5, peržiūri realios aplinkos dalis, kaip režimą, kuris susideda iš begalinės laiko reikšmių eilutės. Eilutė yra saugoma medžio struktūros pavidale skaičiavimo įrenginio viduje, kaip tai dar bus toliau aprašyta. Tokios eilutės pavyzdys yra parodytas FIG 2. Atrinktų signalų eilutė gali keistis su laiku, t.y., pavyzdžių periodai gali būti individualūs, kas matoma išb FIG. 2, arba parinkti iš kompiuterio programos arba pateikti iš aplinkos. Čia pateikta signalo eilutė, kaip porinių elementų sąrašas. Kiekviena pora turi laiko trukmę ir signalų kiekį tos trukmės metu. Taip pat yra galima turėti daugiau negu vieną signalą per laiko vienetą. Pavyzdžiui, tai gali būti atliekama per portą, turintį galimybę apdoroti daugiau negu vieną signalą per laiko vienetą arba turint keletą lygiagrečių portų, iš kurių kiekvienas turi galimybę apdoroti vieną signalą per laiko vienetą. Portai tokiu atveju yra sinchronizuojami vienas su kitu.

Režimų unifikavimas

Pagal šį išradimą įėjimas ir išėjimas per portą į aplinką yra atliekamas, unifikuojant procesus, einančius iš vienos jo pusės, pvz., iš vidinių skaičiavimo įrenginio režimų ir iš išorinių jo aplinkos režimų. Programa, parašyta funkcine kalba H, patekusi į aktyvią objekto atmintį sukuria lokalinį režimą. Realios aplinkos režimas rašo vieną protokolą realiame laike. Tokiu būdu, skaičiavimo įrenginio portas pagal aplinką registruoja tikrąjį aplinkos režimą, stebimą iš skaičiavimo įrenginio. Jis yra unifikuojamas su protokolu, aprašytu kompiuteryje. Kaip tai yra padaroma skaičiavimo įrenginyje pagal šį išradimą, bus aprašyta toliau. Skaičiavimo įrenginio tiek vidiniai režimai, tiek ir išoriniai režimai yra modeliuojami tokiu pat būdu, kas teikia galimybę labai elegantiškai atlikti operacijas, tuo skaičiavimo įrenginį darant tinkamu atlikti operacijas realiame laike.

Tokiu būdu, į skaičiavimo įrenginį įeina ne mažiau, kaip viena lyginimo galimybė eilutės signalų, patenkančių į portą iš aplinkos, lyginimui su eilute, saugoma ne mažiau, kaip vienoje atminties ląstelėje, daugumoje atvejų, keliose atminties ląstelėse, esančiose aktyvioje atmintyje. Unifikuotos operacijos metu visi alternatyvūs vidiniai režimai, kurie neatitinka režimams, esantiems realioje aplinkoje, yra ištrinami. Teisinga programa turi turėti tiktai vieną iš alternatyvių režimų, paliktų po operacijų unifikavimo ir šis režimas turi sutapti su nagrinėjamo porto režimu. Jei nė vienas iš alternatyvių vidinių režimų neatitinka realios aplinkos režimui nagrinėjamame porte, unifikavimo operacija to pasėkoje nieko neduos, o tai yra laikoma, kaip programinė klaida.

H portas

Aukščiau pateiktas įėjimo - išėjimo porto aprašymas gali būti apibendrintas, kad būtų galima tvarkyti kitų duomenų struktūrų įėjimus - išėjimus, o ne ryšius tarp struktūrų. Šiuo atveju, vietoj to, kad unifikavus vidinį režimą su išoriniu režimu, unifikuojant laiko trukmes ir signalų kiekius, gali būti atliktas bitų modelio unifikavimas. Be to, įeinantis signalas gali atstovauti duomenų struktūras, tokias, kaip H kalbos programas. Pavyzdžiui, atminties lastelės laukas porto atmintyje gali būti unifikuojamas su bito modeliu porto įėjime.

Pavyzdys: vidinė struktūra panaudojant ($1 $2) gali būti unifikuojama su bito modeliu įėjime, tokiu būdu, galutinė unifikuota vidinė struktūra atrodys taip: apply(+ list(1 2)), o tai yra parodyta struktūroje FIG. 11A ir 11B. Čia $1 ir $2 nurodo "bet kokį bito modelį". $1 yra unifikuotas su bito modeliu, rodančiu komandos kodą "+" , o $2 yra unifikuotas su bito modeliu, rodančiu "list(1 2)".

Per portą H, kuris priima H kalbos kodą, programa gali būti įvesta į objekto atmintį. H portas gali taip pat būti panaudotas tarp procesorių, kas įgalins juos perduoti programas, duomenis arba juos abu. Perduodamas programos kodas gali viduje būti pažymėtas, kaip duomenys, kad būtų išvengta jo vykdymo tuoj pat. H portas gali taip pat būti panaudotas, kaip interfeisas tarp skaičiavimo įrenginio pagal išradimą ir kito skaičiavimo įrenginio tipo, kuris gali būti žinomas anksčiasu ir būti įprastinio tipo.

Objekto atmintis

Realizuojant skaičiavimo įrenginį, parodytą FIG. 3, į objekto atmintį įeina eilė atminties lastelių 2, kiekviena iš kurių yra padalinta į keleta atminties lauku, kaip tai dar bus aprašyta toliau. Kadangi tiksli objekto atminties struktūra nėra šio

išradimo dalis, ji smulkiau nėra aprašinėjama. Objekto atmintis, kuri gali būti tokia, kaip atmintis 1, yra aprašyta mūsų susijusioje paraiškoje Nr PCT/SE71/00513.

Aritmetinio bloko struktūra

Kiekviena atminties ląstelė gali turėti atmintyje uždarą ląstelę, įskaitant keletą uždarų elementų. Objekto atmintis 1 yra susijusi su struktūriniu aritmetiniu bloku 3 per atminties šyną BUS_1 iki BUS_N, pakankamos apimties, kad sujungtų kiekvieną bito ląstelę su bloku 3. kiekvienas bitas kiekvienoje atminties ląstelėje yra sujungtas su atitinkamais bitais visose kitose atminties ląstelėse per šynos schemą. Struktūrinis aritmetinis blokas 3 gali būti tam tikra objekto atminties 1 dalis, į kurią laikinai yra pervedamas uždaras reiškinys, kurį reikia suskaidyti suskaidymo procedūros metu. Tačiau yra įmanoma turėti tam tikrą bloko 3 struktūrą, kaip bus parodyta toliau, sudarant galimybę greitoms ir tiesioginėms suskaidymo operacijoms. Kadangi tiksli bloko 3 struktūra nėra šio išradimo dalis, ji smulkiau nėra aprašinėjama. Blokas, kuris gali būti realizuojamas, kaip blokas 3, yra aprašytas mūsų pateiktoje paraiškoje PCT/SE91/00514.

Skaičiavimo įrenginys, parodytas FIG.3, yra sumontuotas aplink šynos įrenginį BUS_1 iki BUS_N, taigi, praktiškai visi su duomenimis operuojantys įtaisai turi tiesioginį priėjimą į jį. Tokiu būdu, blokas 3 ir portai gali būti laikomi priklausą aktyviai objekto atminčiai, t.y., kaip tam tikros jos ląstelės.

Centrinis valdymo blokas

Centrinis valdymo blokas CU (control unit) valdo tiek objekto atmintį, tiek ir bloką 3. Kai kurie portai 4, 5, 6, 11 taip pat yra valdomi nuo centrinio valdymo bloko CU ir yra sujungti su bloku 3. Du iš portų - 4 ir 5 yra parodyti , kaip unifikuojantieji portai, sujungti su aplinka. Praplėtimo įrenginiai 7 ir 8, atitinkamai, tokie, kaip sensorius arba prietaisas, valdomas

nuo/arba valdantis skaičiavimo įrenginį, yra sujungtas su kiekvienu portu. Portas 6 yra H portas, sujungtas su kitu skaičiavimo įrenginiu 9, kuris pateikia programas, kurios turi būti įterptos į skaičiavimo įrenginį. Portas 11 yra H portas, sujungtas su kitu skaičiavimo įrenginiu 12, kuris, pavyzdžiui, yra tokio pat tipo, kaip ir skaičiavimo įrenginys nuo 1 iki 6, 10, 11, su kuriais CU turi pastovų ryšį. Keli skaičiavimo įrenginiai, tokio tipo, kaip parodyta FIG. 3, gali būti tiesiogiai sujungti į perdavimą tarp jų atminties šynos schemos ir laikinosios uždaros atminties (neparodyta), kiekviename įrenginyje sudarant daugelį lygiagrečių skaičiavimo įrenginių. Tokiu būdu, gali būti išlaikytos globalinės paieškos galimybės. Tačiau, kai kuriais atvejais, yra patogu sujungti tarpusavyje du ar tris skaičiavimo įrenginius arba daugelį lygiagrečių skaičiavimo įrenginių per H portą, taip, kad kiekvienas skaičiavimo įrenginys dirbtų, kaip atskiras skaičiavimo įrenginys, o atskiri prietaisai siunčia programas ir/arba duomenis tarp jų atskiram apdorojimui. H portas 11 taip pat gali būti naudojamas, kaip interfeisas ir kitokio tipo procesoriui. Kai H portas perduoda programą, perdavimas yra toks pats, kaip ir duomenų perdavimas, o skaičiavimo įrenginys juos tvarko tokiu pat būdu.

Skaitmeninis ALU

Prie bloko 3 galo būti prijungtas papildomas prietaisas 10 tam tikrų specialios rūšies operacijų atlikimui. Tokio prietaiso pavyzdžiu gali būti skaitmeninis aritmetinis blokas, toliau vadinamas skaitmeniniu ALU. Tačiau, papildomas prietaisas 10 gali taip pat būti naudojamas tikslų (paskirties) ir pan. sulyginimui. Skaitmeninis ALU gali būti beveik bet koks ALU bendros paskirties, tačiau ypač gerai tinkamas ALU skaičiavimo įrenginiui pagal šį išradimą yra aprašytas mūsų pateiktoje paraiškoje Nr PCT/SE91/00515.

Sulyginimas

Aktyvi atmintis, įskaitant bloką 3, turi ne mažiau, kaip sulyginimo galimybę lyginti signalų seką, einančią į vieną iš portų, pavyzdžiui, 4, su ne mažiau, kaip viena seka, patenkančia į ne mažiau, kaip vieną iš kitų portų, pavyzdžiui, 6.

Ši sulyginimo galimybė gali būti atlikta tiesiogiai atmintyje arba portuose, pateikiant reikšmes, turinčias uždarą formą, tokią, kaip intervalų poras, kiekviena iš kurių turi trukmę ir reikšmę signalų kiekio pavidalu. Tačiau taip pat yra įmanoma atlikti tam tikrą sulyginimą papildomame bloke 10. Taip pat , kai yra lyginami du išoriniai režimai, sulyginama seka yra perrašoma į nieką , jei sulyginimas parodo skirtumą, pvz., jei trukmės laikas arba signalų kiekiai ties portais ir aplinkoje nėra tie patys. Kitu atveju, sulygintos sekos yra perrašomos į vieną ir tą pačią tam tikrą seką.

Tokiu būdu, aktyvi atmintis 1, 3 turi struktūrą, galinčią įvesti į atmintį ir vykdyti kompiuterinę programą apibrėžtai arba neapibrėžtai koduojant abstrakčiąją sintaksę. Sintaksė aprašo eilę skirtingų abstrakčiųjų objektų lygčių pagalba. Kiekviena atminties ląstelė gali įvesti į atmintį vienu metu ne mažiau , kaip dalį sintaksės lygčių atitinkamų duomenų ir/arba programų struktūrų pavidalu. Sintaksė smulkiau bus aprašyta žemiau.

Objekto atminties galimybės

Objekto atmintis 1 turi iš esmės daugiau intelektinių savybių, negu įprasto RAM tipo atmintis. Ji yra asociatyvi, kas daro galimu teikti daugiau paslaugų, negu tik "skaityti" arba "įrašyti", kaip yra įprastinio tipo RAM atmintyje.

Kaip jau minėta aukščiau, objekto atmintis yra padalinta į atminties ląsteles 2, kiekviena iš kurių turi keletą atminties laukų. Teikiamos paslaugos yra aukšto lygio. Pavyzdžiui, yra galima surasti visus įvykius su tam tikru duomenų elementu, kokioje jis būtų individualioje saugojimo ląstelių vietoje ir

perrašyti tam tikrą rastą duomenų elementą globaliai, pvz., visoje objekto atmintyje, į naują reikšmę, naudojant vien tik vieną atminties komandą. Kadangi objekto atmintis yra asociatyvi, ši perrašymo operacija gali būti atlikta dviem fiziniais atminties ciklais, nepriklausomai nuo liečiamų atminties ląstelių skaičiaus.

Atminties ląstelė

Atminties ląstelės 2 realizavimas yra schematiškai parodytas FIG. 4. Joje gali būti saugomi dviejų tipų elementai ir ji turi atminties laukus, ypatingai pritaikytus saugomiems elementams. Šie laukai FIG. 4 yra pažymėti tais pačiais pavadinimais, kaip ir juose saugomi elementai.

Kiekviena atminties ląstelė yra pritaikyta saugoti ne mažiau, kaip keletą savybių, turinčių sekančią numeraciją:

žymes, nurodančias, ar reiškinys šioje ląstelėje turi būti suskaidytas;

žymes, nurodančias, ar reiškinys yra medžio sudėtinė dalis bei jo savybes;

žymes, nurodančias, kaip yra sudaromas reiškinys;

žymes, nurodančias, ar reiškinys sudaro pakartotinų režimų skaičių;

žymes, nurodančias, ar reiškinys yra tik sąrašo dalis, turinti kitus sąrašo narius, saugomus kitoje atminties ląstelėje.

Pirmojo tipo elementai, žemiau vadinami rekvizitiniais elementais, aprašo skirtingus atminties ląstelių būvius. Vienas iš tokio tipo elementų yra LAZY (tingus), kas rodo, ar ląstelė yra tuščia, tuo atveju ląstelės turinys yra laikomas pasyvia informacija, exec, pvz., yra exec - vykdymo stovyje, arba laukimo, pvz., ląstelės įvertinimas yra atidėtas ir ji laukia rezultatų, kad būtų toliau vykdoma. Kitas rekvizitinis elementas yra TYPE (tipas), į kurį įeina tipo kodas (par, seq, apply, list, unify ir tt.).

Sekantis elementų tipas, toliau vadinamas objekto elementais, aprašo identifikavimą, aplinką arba reikšmę. Tai yra IDENTIFIER, ENVIROMENT, VALUE/DES. Šie objekto elementai yra pritaikyti įvedimui į atmintį centrinių registrų dalimis, pateikiamomis zonose HEAD ir NUM (žr. FIG. 7B, 7C). Kiekvienas iš šių elementų turi elemento žodį, kuris savo ruožtu, yra padalintas į žodžius num ir tag.

Žodis tag (požymis, etiketė)

Antrojo tipo uždari elementai turi požynio žodį, kuris rodo num žodžio savybę. Požymiai - etiketės yra dviejų tipų, netiesioginės etiketės, pvz., etiketės, naudojamos identifikatoriams ir aplinkai ir tiesioginės etiketės, naudojamos paprastoms reikšmėms ir pan.

Netiesioginių etikečių pavyzdžiai yra cls, canon ir open. Jei etiketės žodis yra cls, tai reiškia, kad num žodis atstovauja uždarą reiškinį, kuris gali būti skaidomas. Jei etiketės žodis yra canon, tai reiškia, kad num žodis atstovauja uždarą reiškinį, kuris negali būti toliau skaidomas. Jei etiketės žodis yra open, tai reiškia, kad num žodis atstovauja uždarą reiškinį, kuris yra įterptas sarašas.

Tiesioginių etikečių pavyzdžiai yra discr, cont, unused ir nothing. Jei etiketės žodis yra discr, tai reiškia, kad num žodis yra sveikas skaičius. Jei etiketės žodis yra cont, tai reiškia, kad num žodis yra plaukiojančio kablelio reikšmė. Jei etiketės žodis yra unused, tai reiškia, kad num žodis reiškia nieką, pvz., uždaro reiškinio unifikavimas, turintis lauką, pažymėtą nothing, visada bus niekas.

Identifikatoriai ir aplinka

Jei atminties lastelėje yra identifikatoriaus laukas, proceso būvis iš šios lastelės gali būti perduodamas į struktūrinį aritmetinį bloką 3, toliau vadinama centrine lastele. Kiekviena iš atminties lastelių laukų VALUE/DES gali turėti identifikatorių, rodantį kitą uždaro reiškinio lastelę, tokiu būdu sudarant ryšį su ta kita uždara lastele. Į aplinkos laukus gali įeiti identifikatorius, rodantis uždaro reiškinio šaknį sistemos dalyje, pvz., uždarų reiškinių medyje, sudarančiame aplinką. Tačiau aplinkos laukas taip pat gali turėti ir kitą panaudojimą. Aplinka gali būti panaudota išlaikyti struktūros kūrimo taką (pėdsakus), įvedant į visų sukurtų uždarų lastelių aplinkos atmintį kuriamą identifikatorių. Pavyzdžiui, visos uždaros lastelės, esančios submedyje, kuriose visi simboliai, turintys tuos pačius pavadinimus , reiškia tą patį, gali būti sugrupuojamos su ta pačia aplinka. Tokiu būdu, visa struktūra gali būti išrenkama per vieną uždarą reiškinį medyje, per šaknį, vien tik vienos operacijos pagalba.

Žymėjimo funkcija gali būti laikoma, kaip nukreiptas ryšys nuo tėvo į sūnų, pvz., uždaras elementas unikaliai identifikuoja uždarą lastelę. Mašinos, turinčios asociatyvinio tipo objekto atmintį, režimas yra pavaizduojamas, kaip kryptinis uždarų reiškinių grafas.

Tokiu būdu, jei uždaro reiškinio aplinka yra duota, šioje aplinkoje gali būti surasta uždaro reiškinio šaknis. Uždara aplinkos šaknis ATMINTIES lauke WHERE turi ypatingą žymę (pvz., "1"). Aplinkos uždaro reiškinio susikirtimo taškas turi kitą žymę (pvz., "0") lauke WHERE.

Pavyzdys

FIG. 5 yra parodytas atminties lastelių pavyzdys, į kurių atmintį yra įvestas reiškinys

id_1 = list(pr(1 2 3) par(4 5 6)),

kuris reiškia dviejų lygiagrečių reikšmių kombinacijos sąrašą. Pirmoji lygiagreti kombinacija par(1 2 3) turi požymį id_2, o antroji lygiagreti kombinacija par(4 5 6) turi požymį id_3.

Šaknies atminties lastelė, turinti uždarą lastelę su požymiu id_1, medyje turi etiketę cls, turi nuorodą exec lauke LAZY, turi "1" lauke WHERE, nuorodą list lauke TYPE ir turi id_2 ir id_3 pirmuose value/des laukuose. Šių laukų etiketės yra pažymėtos canon, kadangi šių laukų turinys yra netiesioginis ir susijęs su kanoninėmis uždaromis lastelėmis.

Atminties lastelė, esanti susikirtime, turinti uždarą lastelę su požymiu id_2, turi stovį "0" lauke WHERE , turi nuorodą par lauke TYPE ir turi diskretines reikšmes 1, 2, 3, pirmuose value/des laukuose. Šių laukų etiketės yra pažymėtos discr.

Atminties lastelė susikirtime, turinti uždarą lastelę su požymiu id_3, turi "0" lauke WHERE, nuorodą par lauke TYPE ir turi diskretines reikšmes 4, 5, 6, įvestas į atmintį pirmuose value/des laukuose. Šių laukų etiketės taip pat yra pažymėtos discr.

Portas, kaip interfeisas su išore

Kaip buvo anksčiau minėta, procesorius pagal šį išradimą turi vieną arba kelis portus duomenų įvedimui/išvedimui į arba iš įrenginio. Kiekvienas portas yra blokas, užtikrinantis išorės aplinkos ir procesoriaus įrenginio ryšį. Portas yra pageidautina prijungti prie keitimo interfeiso (neparodyta) objekto atmintyje (žr. FIG.3). I porta galima žiūrėti, kaip į objekto atminties praplėtimą, kadangi jis gali turėti keturias atminties lasteles, tokių pat funkcijų ir galimas sujungti su objekto atminties

lastelėmis 2 (žr. FIG. 4). Portas atlieka įėjimo/išėjimo unifikuotas operacijas.

Vidinis režimas

Objekto atmintis užduoda procesoriaus vidinį režimą, po vieną kiekvienam iš esančių portų. Režimas yra semantinė laiko struktūros prasmė, kuri gali būti pateikta, kaip uždara struktūra. Laiko struktūra gali būti laikoma, kaip proceso, susidedančio iš begalinės reikšmių sekos, būvis.

Išorinis režimas

Įėjimo į porta(us) signalai gali ateiti iš sensorių arba įtaisų, esančių už procesoriaus ribų, t.y., iš realios aplinkos. Portas įeinančius signalus paverčia į skaitmeninę forma, pritaikytą objekto atminties formatui, t.y., uždaram saugojimui. Ši atmintis yra laiko struktūra , t.y., režimas, kurį galima lyginti su procesoriaus vidiniu režimu. Tokiu būdu, žiūrint iš procesoriaus pusės, ši laiko struktūra gakli būti laikoma išoriniu režimu.

Įėjimas ir išėjimas per porta yra atliekamas, unifikuojant procesus, pateikiamus iš abiejų jo pusių, t.y., iš vidinio procesoriaus režimo ir išorinio režimo, ateinančio iš aplinkos. Tiek vidinis, tiek ir išorinisd režimai yra modeliuojami tokiu pačiu būdu, kas įgalina naudotis labai elegantišku operaciniu metodu ir pilnai panaudoti procesorių atlikti operacijas realaus laiko mastelyje.

Išorinio režimo pavyzdys, pvz., laiko impulsų seka, yra parodytas FIG. 2. Atrinktų signalų seka gali keistis laike, t.y., atrinkti periodai gali būti skirtingi, kaip matosi iš FIG. 2. Čia signalų seka yra pateikta, kaip porinių elementų eilutė. Į kiekvieną porą įeina trukmės laikas ir signalų kiekis šiame laiko periode. Signalų kiekiai yra pažymėti q_i, o laiko trukmė - t_i, kur i yra skaičius nuo 1 iki 6.

Portas, skirtas ryšiui su aplinka, toks, kaip portas 4, FIG. 3, gali būti sujungtas, pavyzdžiui, su sensoriu 7. Prie porto esantis interfeisas bendru požiūriu gali tvarkyti visus signalų tipus. Tačiau tokio tipo interfeisas yra sudėtingas ir dažnai gali įvesti sutrikimus sistemoje. Tokiu būdu, pageidautinas interfeisas turi turėti sekančias savybes:

1. Signalas išmatuojamas tam tikrais laiko tarpais. Išmatuotas signalas yra laikomas pastoviuy iki sekančio matavimo.
2. Laikas gali būti užduodamas arba tam tikru ciklu, nustatomu porte, arba tam tikru ciklu, nustatomu įrenginio programoje, arba tam tikrais iontervalais, nustatomais programoje arba išoriniais laiko signalais.
3. Išmatuotas signalas yra pateikiamas į įrenginį skaitmenine forma.
4. Skaitmeninis signalas yra loginis arba užkoduotas skaitmenine forma, kuri gali nurodyti sveiką skaičių arba plaukiojančio kablelio reikšmę.

Tokiu būdu, skaičiavimo įrenginys pagal šį išradimą nepriima ir neišduoda reikšmių, kaip įprastiniai procesoriai. Pagal šį išradimą procesoriaus portas yra laikomas interfeisu tarp procesoriaus programos ir jo aplinkos. Skaičiavimo įrenginys yra adaptuotas funkcinei programinei kalbai H. Interfeisas yra toks, kad įvykių seka pateikiama taip, kad ji iš vienos jo pusės pasireiškia, kaip signalas ir, iš kitos pusės, kaip H struktūra procesoriuje. Aplinka nusako savo fizinės būsenos dalį, o procesorius - savo. Tai yra atliekama iš abiejų porto arba interfeiso pusių, taip, kad jie sudaro to paties reiškinio vaizdą, skaičiavimo įrenginyje duodantį realaus laiko vaizdą, kiek tik įmanoma, artimesnį tiksliam realaus laiko masteliui.

Programa yra įvedama į aktyviąją objekto atmintį ir sukuria vidinį režimą. Programos vykdymas yra paremtas tuo, kad tiek skaičiavimo įrenginio programa ir aplinka išlaiko interfeise tą pačią realaus laiko įvykių seką.

Programoje interfeisas yra aprašytas, kaip begalinis reikšmių, [illegible] ir laiko trukmės sąrašas, t.y., kiekvieno signalo reikšmės trukmė laiko tarpe. Tokiu būdu, kiekvieno intervalo eigoje interfeisas apibūdina signalų porą ir laiko trukmę. Visos tos poros yra pateikiamos eilės tvarka, o portas realaus laiko režimą paverčia į tokio pat tipo sąrašą.

Įėjimai ir išėjimai į ir iš skaičiavimo įrenginio yra atliekami unifikuotu būdu. Tai reiškia, kad kiekviename intervale trukmės laikas ir reikšmės yra tokie patys iš abiejų porto pusių.

Programoje gali būti nurodyta tam tikra signalo kiekybinė reikšmė arba ji gali likti nenurodyta, naudojant ypatingą simbolį, pavyzdžiui, $, rodantį visas galimas reikšmes.

Tiksli programa turi turėti palikta tik vieną kurį nors režimą po unifikavimo operacijos ir šis režimas turi atitikti režimą ties nagrinėjamu portu. Jei nei vienas iš galimų vidinių režimų neatitinka realios aplinkos prie konkretaus porto, unifikavimo operacija rezultate duos nieką, kas yra laikoma, kaip programinės kalbos klaida, kadangi niekas rodo esant prieštaravimą. Turint daugiau, negu vieną po unifikavimo palikta režimą, tai irgi laikoma, kaip programinė klaida.

Kai vidinis režimas nurodo tam tikrą signalo reikšmę, taip pat vadinamą signalo kiekiu, ši tam tikra signalo reikšmė yra išėjimas į aplinką. Jei vidinis režimas signalo reikšmę palieka nenurodytą, ši signalo reikšmė yra įėjimas iš aplinkos.

Kai vidinis režimas nurodo tam tikrą intervalo laiko trukmę, tada aplinka priima laisvą laiko trukmę. Jei vidinis režimas palieka laiko trukmę nenurodytą,

Įrenginio veikimas yra simetrinis tokiu požiūriu:

1. Į intervalą, gaunantį lygtį jo metu, įeina trukmės laikas ir signalo kiekis.
2. Programoje intervalas yra aprašomas signalo kiekio pavidalu laiko trukmėje.
3. Aplinka gali aprašyti tokį intarvalą skirtingais būdais, kaip aukščiau aprašyta.

4. Portas nustato signalo kiekį iš abiejų jo pusių. Tai reiškia, kad signalo kiekis turi būti determinuotas. Skaičiavimo įrenginio programa nustato trukmės laiką arba jo kiekybę iš porto pusės, einančios į skaičiavimo įrenginį. Išėjimas yra atliekamas tam tikros laiko trukmės metu ir/arba signalo kiekis yra perduodamas į porto laidininkus puseje, einančioje į aplinką.

Porto,skirto ryšiui su aplinka, realizavimas

FIG. 6 yra parodytas portas, skirtas intervalų įvedimui ir išvedimui. Portas yra sujungtas su struktūriniu aritmetiniu bloku 3, arba tiesiai prie objekto atminties 1, turinčios šyną DU. Kiekviena šyna įrenginyje DU sudaro šyną, turinčią, pavyzdžiui, 38 jungtis. Portas yra valdomas nuo centrinio bloko CU. Portas turi identifikuojantį registrą P_{ID}. Identifikatorius, įvestas į registrą P_{ID}, gali būti panaudotas, kaip identifikatorius objekto atminčiai 1, t.y., jis gali būti panaudotas sujungti portą su uždaromis ląstelėmis objekto atmintyje. Šis identifikatorius yra naudojamas tik procesoriaus viduje. Į portą taip pat įeina identifikuojantis registras CE_{ID}, kuris gali būti naudojamas tokiu pačiu būdu, kaip ir registras P_{ID}. Į portą taip pat įeina eilė atminties ląstelių CE_{DU}, CE_{LAST}, CE_{NEXT}, CE_{ID}, kiekviena iš kurių iš esmės yra sudaryta taip pat ir veikia tokiu pačiu būdu, kaip ir objekto atmintis 1.

Kiekviena atminties ląstelė gali saugoti uždarą reiškinį, turintį ne mažiau trijų laukų, skirtų trijų uždarų elementų saugojimui, pavyzdžiui, during, trukmės laiką ir signalo kiekį.

Ne mažiau dviejų paskutinių paminėtų uždarų elementų gali būti saugomi atminties laukuose, kurie yra reikšmės/priskyrimo pavidalo. Išraiška during gali būti perduota į lauką TYPE. Kadangi portas visada pateikia operaciją during, šis TYPE laukas gali būti praleidžiamas.

Tačiau realizavimo metu, parodytame FIG.6, during yra 'apply ('@) pavidalo, esantis lauke kartu su priskirtu funkcijos vardu (during kodas). Funkcijos vardas yra saugomas pirmame reikšmės/priskyrimo lauke porto atminties lastelėse, kaip ir visiems funkcijų pavadinimams (tokiems, kaip + - *) ir priklausantiems tipui 'apply. Kodas during gali būti identifikatoriumi, rodančiu funkcijos paskirtį, kuri yra saugoma objekto atmintyje.

Identifikatorius id gali taip pat būti saugomas reikšmės/priskyrimo lauke. Šis identifikatorius jungia porto struktūrą su struktūra during, esančia objekto atmintyje.

Jei lauko, turinčio identifikatorių, nereikia, tada tik trys reikšmės/priskyrimo laukai yra iš tikrųjų reikalingi pagal realizavimą, parodytą FIG. 6. Tačiau taip pat gali būti ir ketvirtas reikšmės/priskyrimo laukas unused nenaudojama) tam tikslui, kad porto registrai būtų suderinami su kitomis atminties lastelėmis skaičiavimo įrenginyje. Taip pat yra galima prijungti papildoma prietaisą, skirtą VALUE IN/OUT prie ketvirto reikšmės/priskyrimo registro. Tai porta paverčia į portą, skirta reikšmių grupei, t.y., laiko trukmei ir dviems signalų kiekiams.

Kiekviena atminties lastelė porte gali tokiu būdu turėti rekvizitinius laukus. Pagaliau, pastarosios atminties lastelės gali būti tarpusavyje sujungtos per šynas W_{M1}, W_{I1}, W_{T1}, W_{V1}, W_{E1}, W_{T2}, W_{V2}, W_{E2}, kiekviena šyna turi, pavyzdžiui, 38 jungtis ir yra sujungta kiek sudėtingiau su bitų lastelėmis kiekviename registre. Indeksas I rodo, kad šyna yra jungiama prie registro during kodo laukelių , Indeksas I rodo, kad šyna yra jungiama prie registro during kodo laukelių ,

Indeksas I rodo, kad šyna yra jungiama prie registro during kodo laukelių, indeksas T rodo, kad šyna yra jungiama prie trukmės laiko reikšmės/žymėjimo registro laukelių, v -kad šyna sujungta su signalo kiekio reikšmės/žymėjimo laukų registruose, indeksas E rodo, kad šyna yra jungiama prie papildomų reikšmės/

/žymėjimo registro laukelių . Indeksas 2 rodo, kad šyna yra jungiama prie aplinkos, o indeksas 1 - kad šyna yra prijungta prie aktyvios objekto atmintiės 1, 3 - yra galimybė prijungti per pakeičiantį interfeisą (neparodyta).

Turi būti pažymėta, kad galima naudotis lauko tipu, kuris turi '@ ir pirmu reikšmės/žymėjimo lauku, turinčiu kodą during, praktiškai praleidžiama, kadangi esanti jame informacija visada bus ta pati, o šis faktas gali būti įgyvendintas centriniame valdymo bloke CU. Taigi, galima sudaryti portą, turintį atminties lasteles, į kurias įeina tik du reikšmės/žymėjimo laukai, vienas, skirtas trukmės laikui, o kitas signalo kiekiui (neparodyta).

Parodytame realizavime keturios porto atminties lastelės saugo sekančius, skirtus apdorojimui uždarus reiškinius, tikrąjį reiškinį, sekantį reiškinį ir' sekančių reiškinių identifikatorius. Atminties lastelės gali turėti priskirtas vietas ir tada visas jų turinys gali būti perduodamas tarp jų. Kiekvienos bito lastelės atitinkama struktūra registruose gali turėti tą pačią struktūrą, kaip ir bito lastelės, esančios aritmetinio bloko 3 struktūroje ir ji yra parodyta mūsų pateiktoje paraiškoje išradimui Nr PCT/SE91/00512 ir PCT/SE91/00513.

Tačiau taip pat yra galima padaryti pirmąsias paminėtas atminties lasteles taip, kad jos galėtų pasikeisti vietomis, keičiant jų pavadinimus, kas gali būti atliekama valdant iš centrinio valdymo bloko CU.

Portas gali turėti skirtingą atminties lastelių skaičių, pvz., vieną atminties lastelę.

Laiko skaitiklis ir komparatorius

I portą įeina laiko skaitiklis T_{COUNT}. Jis yra valdomas laiko signalais $CLOCK_P$ iš centrinio valdymo bloko CU, matuojant laiką. Gražinimas į pradinę padėtį (numetimas) yra atliekamas arba nuo išorinio numetimo signalo arba nuo tokio pat signalo iš valdymo bloko CU. Laiko skaitiklis T_{COUNT} yra sujungtas su pirmu selektoriumi SEL, kuris yra valdomas iš centrinio valdymo bloko CU - sujungiant išėjimą iš laiko skaitiklio T_{COUNT} tiesiai į šyną W_{T2}, sujungtą su laiko atminties laukais atminties lastelėse, arba į laiko komparatoriaus T_{COMP} pirmąjį įėjimą. Antrasis laiko komparatoriaus T_{COMP} įėjimas yra sujungtas su šyna W_{T2}. Išėjimas iš laiko komparatoriaus yra išorinis laiko išėjimas. Išėjimas iš laiko komparatoriaus taip pat yra perduodamas į valdymo bloką CU.

I portą taip pat įeina išorinės reikšmės įėjimo ir išėjimo kojelė VALUE IN/OUT. Ji yra sujungta su konverteriu $CONV_{OUT}$, išėjimo signalui perduoti, pavyzdžiui, per diskretinį - analoginį konverterį. Ji taip pat yra sujungta su konverteriu $CONV_{IN}$, įėjimo signalui perduoti, pavyzdžiui, per analoginį - diskretinį konverterį. Taip atliekamas įėjimo signalo rūšiavimas. Rūšiavimo laikas gali būti imamas iš laiko komparatoriaus T_{COMP} išėjimo arba iš numetimo signalų iš laiko skaitiklio T_{COUNT} (neparodyta). Selektorius SEL_2 yra sujungtas su šyna W_{V2} ir per valdymo signalus yra sujungtas su signalų kiekio registrais atminties lastelėse CE_{DU}, CE_{LAST}, CE_{NEXT}, valdant iš centrinio valdymo bloko CU, jei signalo kiekis perduodamas iš arba į portą. Valdymo blokas CU pirmiausia nuskaito registrų turinį atminties lastelėje, įskaitant registrus, sujungtus su aplinka, norint pamatyti, ar ten yra ypatingas simbolis $, reiškiantis, kad registro vertė yra pasirenkama ir gali būti pakeista bet kuria galima reikšme. Po to, valdymo blokas CU valdo porto selektorių, prijungtą atitinkamai prie registro.

Taip pat yra įmanoma padaryti ir sudėtingesnį portą. Pavyzdžiui, signalų kiekiai, skirti VALUE IN/OUT gali būti

pakeisti formulės, saugomos keitimo schemoje $CONV_{IN}$ įėjimo reikšmems ir $CONV_{OUT}$ išėjimo reikšmems. Keitimo funkcija gali būti, pavyzdžiui, tam tikras skaičius integralų, kiekvienas iš kurių yra skirtas skirtingai signalų kiekio klasei arba panašiai.

Taip pat yra galima paimti dvi reikšmes kiekvienos laiko trukmės metu, pavyzdžiui, vieną pradžioje, kitą laiko trukmės pabaigoje, ir vieną iš jų įvedant į atmintį tiesiai į reikšmės/priskyrimo lauką ir pateikti į skaičiavimo įtaisą (neparodytas) dvi išmatuotas reikšmes ir trukmės laiką. Išėjimas iš skaičiavimo įtaiso tada gali būti įvestas į papildomą reikšmės/priskyrimo lauką (neparodytą), t.y., papildomą selektorinį prietaisą, skirtą VALUE IN/OUT.

Signalo kiekio išėjimas

Išsiunčiama informacija yra sutelkiama atminties ląsteleje CE_{DU}. Tokiu būdu, ląstelė turi informaciją

during trukmės laikas signalų kiekis

Signalų kiekis šiame registre yra žinomas ir, tokiu būdu, yra perduodamas į išėjimą. Signalo kiekis yra pristatomas iš signalų kiekio registro į atminties ląstelę ir perduodamas išorėn per selektorių SEL_2, veikiančio , kaip draiverio laipsnis, bei konverterį $CONV_{OUT}$.

Signalo kiekio įėjimas

Informacija atminties ląsteleje CE_{DU} yra

during trukmės laikas $

Signalų kiekis šiame registre , tokiu būdu, nėra žinomas ir registras yra paruošiamas priimti signalų kiekį. Signalų kiekis yra pristatomas iš porto įėjimo/išėjimo, paverčiamas skaitmenine forma A/D (analoginis-diskretinis) keitiklyje $CONV_{IN}$ ir perduodamas į signalų kiekio registrą, esantį atminties ląsteleje CE_{DU}.

Trukmės laikas yra nustatomas
skaičiavimo įrenginio programoje

Informacija atminties lastelėje CE_{DU} yra

during trukmės laikas signalų kiekis

Signalų kiekis šiame registre , tokiu būdu, yra žinomas. Laiko skaitiklis T_{COUNT} yra numetamas iš centrinio valdymo bloko CU trukmės laiko pradžioje. Į skaitiklį T_{COUNT} tolygiai perduodamas laikas laiko skaitikliu T_{COMP} yra sulyginamas su laiku registre, esančiame atminties lastelėje CE_{DU}. Trukmės laikas baigiasi, kai šie laikai sutampa.

Atminties lastelė CE_{LAST} yra ištuštinama ir atminties lastelė CE_{NEXT} prisipildo šios laiko trukmės metu. Pereinant prie kito periodo, atminties lastelės CE_{DU} turinys perkeliamas į atminties lastelę CE_{LAST}, ir signalų kiekis atminties lastelėje CE_{NEXT} yra perkeliamas į atminties lastelęCE_{DU}.

Trukmės laikas yra nustatomas aplinkoje

Informacija atminties lastelėje CE_{DU} yra

during $ signalo kiekis

Signalų kiekis šiame registre , tokiu būdu, nėra žinomas ir jį turi nustatyti aplinka. Laiko skaitiklis T_{COUNT} yra numetamas iš išorinio valdymo laiko trukmės pradžioje. Į skaitiklį T_{COUNT} tolygiai perduodamas laikas yra perdyuodamas į laiko registrą, esantį atminties lastelėje CE_{DU}. Laiko trukmė baigiasi, kai iš aplinkos ateina sekantis išorinis numetimo signalas. Tokiu būdu, išorinis numetimo signalas valdo selektorių SEL_1, atjungiantį ryšį su laiko skaitikliu tiesiogiai arba nuliniu signalu iš laiko skaitiklio.

Atminties lastelė CE_{LAST} yra ištuštinama ir atminties lastelė CE_{NEXT} prisipildo šios laiko trukmės metu. Pereinant prie kito periodo, atminties lastelės CE_{DU} turinys perkeliamas į atminties lastelę CE_{LAST}, ir turinys atminties lastelėje CE_{NEXT} yra perkeliamas į atminties lastelęCE_{DU}.

Paprastas pavyzdys

Žemiau yra pateikiamas nedidelis pavyzdys, norint iliustruoti porto veikimą. Sakykime, kad mes norime nustatyti vieną iš skirtingų sekų, pirmoji iš kurių turi nustatytą kiekvieną sekundinį signalą, pvz., reikšme 17, o visos kitos laiko trukmės nustatytos, pvz., 1 sek., o antroji seka turi signalų kiekį, nustatytą, pvz., 1, 2, 3, 4 ir tt. sek., tačiau nenurodyta laiko trukmė. Naudojamas portas yra vadinamas $port_1$. Uždaras reiškinys nurodomas:

unify($port_1$ "internal behaviour, vidinis režimas"),

kur "internal behaviour" yra operacija alt, rodanti visas galimas sekas, pvz., sekančio pavidalo dideles duomenų struktūras:

alt(seq(during(1s 17)during(1s $)during(1s 17)....)

seq(during($ 1)during($ 2)during($ 3)....))

Dabar, jei įėjimo signalo seka gali būti unifikuota su aukščiau stovinčia struktūra, įėjimas yra priimamas ir gali būti pradėti atitinkami veiksmai. Vienas priimtos sekos pavyzdys yra (1s 17)(1s 0)(1s 17)(1s 99)(1s 17)), kuri reiškia seką, galimą unifikuoti su pirmąja, pateikta aukščiau. Kita priimta seka bus (2s 1)(4s 2)(1s 3) kuri reiškia seką, galimą unifikuoti su vidaus režimo antraja seka. Nepriimtos sekos pavyzdys yra ((1s 2)...), nes isgnalo kiekis, pvz., 2, negali būti unifikuotas su signalų kiekiais, duotais dviejose vidinio režimo sekose.

H porto realizavimas

H porto realizavimas yra parodytas FIG. 7. Portas yra skirtas sujungti du atskirus skaičiavimo įrenginius pagal šį išradimą. H porto pusė, priklausanti pirmajam skaičiavimo įrenginiui, turi lauką $TYPE_{11}$, keturis reikšmės/žymėjimo laukus V/D_{11}, V/D_{12}, V/D_{13}, V/D_{14} ir identifikuojantį registrą ID_{P11}. H porto pusė, priklausanti antrajam skaičiavimo įrenginiui, turi lauką $TYPE_{21}$ keturis reikšmės/žymėjimo laukus V/D_{21}, V/D_{22}, V/D_{23}, V/D_{24} ir identifikuojantį registrą ID_{P21}. Realizavimo

pavyzdyje, nurodytame FIG. 7, type (tipo, spausdinimo) laukai ir reikšmės/žymėjimo laukai iš abiejų porto pusių yra sujungti su šyna. Perdavimas atliekamas per serijines šynas.

Kadangi turi pranašumų toks atvejas, kad geriau turėti kuo mažiau fizinių tarpusavio sujungimų, geriausias darbo režimas bus tada, kai duomenys bus perduodami tarp porto pusių per vieną serijinę šyną. Tačiau toks perdavimas vidines valdymo schemas, esančias centriniame valdymo bloke CU, daro sudėtingesnes, negu tuo atveju, kai perdavimas yra atliekamas, naudojant vieną perdavimo šyną atminties laukui. Interfeisai lygiagrečių duomenų pavertimui į nuoseklius ir atvirkščiai FIG. 7 nėra parodyti ir nėra aprašyti, kadangi manoma, kad jie yra žinomi kvalifikuotam asmeniui.

Kiekviena H porto pusė yra sujungta su valdymo bloku CU_1 ir CU_2, atitinkamai, kurie valdo jiems priklausantį skaičiavimo įrenginį. Vienas iš valdymo blokų - CU_1, duoda sinchronizavimo signalus SYNC, sinchronizuojančius kitą valdymo bloką CU_2. Tada, kai turi būti atliekamas perdavimas, viena H porto pusė visuose savo registruose turi ypatingą simbolį $, išskyrus identifikavimo registrą, kuris yra indikuojamas iš valdymo bloko, kuris tokiu būdu perima kitos pusės atitinkamų registrų turinį.

Tada, kai turi būti perduodamas didelis paketas programos duomenų iš vieno skaičiavimo įrenginio į kitą per H portą, identifikatoriai turi būti pakeisti priimančiajame skaičiavimo įrenginyje, kad jie būtų pritaikyti vidiniam numerių paskirstymui, kad tie patys numeriai nebūtų dukart panaudojami. Medis tokiu būdu yra perduodamas arba nuo šaknų iki viršūnės, kas labiau yra pageidautina, arba nuo viršūnės iki šaknų, perduodamuose uždaruose reiškiniuose pakeičiant kiekvieną identifikatorių nauju ir įvedant juos atitinkamai į objekto atmintį.

H kalbos režimas

Programinė H kalba naudoja režimą, kuris turi tris formas

p t a ,

kur p, t ir a yra trys modeliavimo kintamieji. Parametrai p, t ir a modeliuoja lygiagrečios ir nuoseklios sekos laiką bei skirtingus veiksmus. Jie gali būti laikomi, kaip indeksai, kurie nurodo skirtingus lygiagrečius veiksmus, skirtingus įvykius ir pasirenkamus režimus. Šios trys formos yra naudingos, įgyvendinant įvairius režimus, aprašomus specialiomis konstrukcijomis privalomose kalbose. Jos, bendrai paėmus, sudaro pagrindinę H kalbos idėją, jos esmę. Skaičiavimo įrenginio struktūra pagal šį išradimą yra atitinkamai pritaikyta prie proceso režimo, turinčio šias tris formas. Reikia pažymėti, kad kai kuriais atvejais, kintamojo a modeliavimas gali turėti daugiau reikšmių, negu viena, t.y., keletas besikeičiančių įvykių gali įvykti tuo pačiu metu.

Aritmetinio bloko struktūra (pagrindinis elementas)

Kaip buvo aukščiau paminėta, suskaidymo operacijos paprastai yra atliekamos ypatingame aritmetiniame bloke 3, į kurį yra perduodamas ribotas uždarų lastelių kiekis , esantis medžio struktūroje, tuo metu, kai turi būti atliktas suskaidymas.

Dabar pateiksime trumpą aritmetinio bloko 3 (pagrindinių elementų) realizavimo struktūros paaiškinimą ir kokiu būdu lygtys ir reikšmės, saugomos objekto atmintyje turės savo atitikmenis pagrindinės lastelės 3 registruose ir kaip vyksta informacijos apsikeitimas tarp objekto atminties 1 ir pagrindinės lastelės.

Pagrindinio elemento registrai

Registrai, kurie gali būti panaudoti, įgyvendinant pagrindinį bloko elementą (lastelę) yra parodyti FIG. nuo 8A iki 8C.

Registrų struktūra (konfigūracija), kuri gali būti panaudota, raelizuojant pagrindinę lastelę, yra parodyti FIG. 8D.

FIG. 8A yra parodytas registras. Brežinyje iliustruojama, kad registrai yra sudaryti iš registrų lastelių, kiekviena iš lastelių gali turėti atmintyje viena informacijos bitą. Tas vaizdas, kaip yra parodytas registras, parodo, kad registras tęsiasi per skirtingas pagrindinės lastelės matricas, kiekviena registro lastelė yra vienoje iš matricų.

FIG. 8B parodytas registras, kuris tęsiasi per skirtingas pagrindinės lastelės matricas, t.y., pilnas registras. Šis registro tipas savo elementuose gali turėti identifikatorių arba reikšmę, kuri yra matricose NUM ir HEAD, esančiose pagrindinėje plokštėje. Ten taip pat gali būti nurodytas būvis, kuris buvo aprašytas aukščiau, esantis pagrindinės plokštės lastelėse TYPE, LAZY ir CLOS/SIMPLE.

FIG. 8C parodytas registras, kuris yra tik NUM ir HEAD pagrindinės lastelės matricose, t.y., ribotas registras.

FIG. 8D yra parodyta galima registrų struktūra pagrindinėje plokštėje. Pagrindinė plokštė turi bazinius registrus, geriausiai, išdestytus kvadratu, vadinamu bazinio registro matrica. Baziniai registrai yra išdėsyti pagrindinėje eilėje, išilgai vienos iš jų šonų, vadinamoje pagrindinių registrų pusėje. Pagrindinių registrų stulpeliai, kiekviename iš kurių yra vienas pagrindinis registras stulpelio apačioje, yra vadinami papildomais registrais. Pagrindinė plokštė taip pat gali turėti identifikatoriaus registra ir aplinkos registra. Bazinės registrų matricos šone yra išdestyti papildomi registrai.

Pagrindinės plokštės realizavimo schemoje papildomi registrai gali būti tokio pavidalo, kaip yra parodyta FIG. 8C, t.y., riboti registrai, o likusieji registrai, parodyti FIG. 8D, gali būti tokie, kaip parodyti FIG. 8B, t.y., pilni registrai.

Smulkesnis pagrindinės plokštės aparatinės dalies aprašymas yra pateiktas mūsų paraiškoje Nr PCT/SE91/00514. Trumpas duomenų

įvedimo į atmintį formų aprašymas bus duotas, kalbant apie FIG. nuo 9A iki 9F, o kai kurie jų veikimo pavyzdžiai bus duoti, aprašant FIG. 10A iki 10H, 11A iki 11G ir 12A iki 12G.

Pprasta reikšmė

Kaip parodyta FIG. 9A, paprasta reikšmė 25, kaip suskaidymo rezultatas yra tam tikrame registre, pagrindinių registrų zonoje. Šis rezultatas gali būti viena uždaros lastelės dalis.

Vieno lygio struktūra

Tikslą sudaro tai, kas yra įvedama į pagrindinę plokštę tolesniam suskaidymui. Kaip parodyta FIG. 9B, tikslas, turintis tik vieną lygį, sudarantis tipinį uždarą reiškinį, be kreipimosi į kitas uždaras struktūras, yra saugomas pgrindiniame registre. Pavyzdys parodo paprastą aritmetinę operaciją, t.y., reikšmių 1, 2 ir 3 sudėtį. Skaitmeninė komanda (+) yra saugoma pirmame pagrindiniame registre ir elementai, kuriuos reikia apdoroti, yra įvedami į kitus pagrindinius registrus.

Dviejų lygių struktūra

Kaip parodyta FIG. 9C, medis, į kurį įeina dviejų lygių struktūra, gali turėti savo pagrindinį sąrašą, kuris yra vadinamas tėvu ir yra įvestas į pagrindinius registrus horizontaliai ir sarašai, kurie vadinami sūnumis, vertikaliai baziniame registre. Šiame pavyzdyje struktūra, turinti sąrašo pavidalą ((1 2) (3 4)) yra įvedamas į atmintį bazinio registro matricoje. Pagrindinis sąrašas, pvz., 1 ir 3, esantis pirmu elementu sąraše, yra laikomas pagrindinio registro atmintyje, o sūnaus sąrašai, t.y., (1 2) ir (3 4), yra saugomi vertikaliai papildomuose registruose. Pavyzdys, kuriame toks įvedimo į atmintį būdas yra naudojamas, bus pateiktas toliau, žr. FIG. 9A iki 9H.

Trijų lygių struktūra

Kaip parodyta FIG. 9E, medis, į kurį įeina trijų lygių struktūra turi savo šaknį, įvestą į papildomus registrus, o jo vienas sūnus yra įvestas į pagrindinį registrą. FIG. 9D tikslo

medžio šaknis, kurioje komanda yra perstatoma, yra vienu iš

papildomų registrų, o jos sūnus turintis sąrašo pavidalą (id1, id2, id7) yra įvedamas į atmintį baziniuose registruose. Kiekvienas šio sąrašo elementas savo ruožtu yra tėvas, turintis sūnus. FIG. 9E šie sūnūs yra įvesti į bazinius registrus vertikaliai, kur id1 yra pakeičiamas į sąrašą, kurį jis žymi, pvz., (1 2 3), kur id2 yra pakeičiamas į sąrašą, kurį jis žymi, pvz., (11 12 13) ir kur id7 yra pakeičiamas sąrašu, kurį jis žymi, t.y., (21 22 23).

Vamzdyno režimas

Kaip parodyta FIG. 9F, medis, kuris yra įvestas vamzdyno režime, yra įvedamas į pagrindinius registrus kartu su tikslo sąrašu, o tikslo tėvas - į papildomus registrus, be to, abiejuose registruose yra įvesta vykdymo komandos ir jų elementai. Operacinis vamzdyno režimą geriausiai naudoti, skaidant skaitmeninius reiškinius. Vienas iš pranašumų yra tai, kad tarpiniai rezultatai gali būti laikinai įvesti į atmintį pagrindinėje plokštėje, o ne objekto atmintyje.

Tokiu būdu, tikslo medžio šaknies sąrašą geriausiai tinka laikyti registrų skirtingose vietose pagrindinėje plokštėje, priklausomai nuo medžio struktūros lygio ir reikiamų atlikti operacijų.

H kalbos struktūra ir jos aparatinė dalis

Skaičiavimo įrenginiui pagal šį išradimą ypatingai pritaikyta yra programinė kalba, vadinama H kalba. Tradicinės problemos sudėtingumo sprendimų priemonės buvo sudėtingų dalykų slėpimas, o paprastųjų parodymas. Skaičiavimo įrenginys pagal šį išradimą naudoja priešingą metodą. Buvo pašalinti visi nereikalingi programinės įrangos lygiai. H kalba yra naudojama visiems reikalams mašinoje: kaip mašinos kalbą, programavimo kalba, operacinė sistema ir bendravimo protokolui. Tačiau, kaip ir kiekvienam šių laikų elektroniniam prietaisui, skaičiavimo įrenginio pagal šį išradimą schema tam tikru mastu gali būti imituojama kompiuteryje, naudojantis interpretatoriumi arba kompiliatoriumi. Tokios mašinos pavyzdys yra SUn 3. Tačiau mašina turi turėti portą, kuriame yra sutelktos unifikavimo galimybės.

Įprastinės loginės programavimo kalbos, kaip pagrindinę semantinę savybę turi skiriamąją gebą ir unifikavimo galiimybes. Skaičiavimo įrenginys pagal šį išradimą, naudojant H kalbą, skiriamajai gebai pasiekti naudoja modelių suderinimą, o suskaidymo kalbos unifikavimui naudoja modelių suderinimą ir laisvus kintamuosius. H kalboje objektai, įvedami į atminties ląsteles, turi lygiagretaus būvio, nuoseklaus būvio ir pasirenkamo būvio\ savybes.

Tikslo medžio šaknis yra suskaidomo tipo uždaras reiškinys, toks, kaip unify, alt, apply ir tam tikra prasme, par ir seq , kadangi jie kai kada gali būti suskaidyti iki nieko. Funkciniam pritaikymui apply (taip pat vadinamas @) pirmasis elementas yra komanda (+ - * / during), tiesiogiai interpretuojama aparatine dalimi arba identifikatorius, netiesiogiai žymintis uždaro reiškinio sandarą, naudojamą, kaip funkcinis apibrėžimas, o likusieji elementai yra komandos/funkcijos apibrėžimo argumentai. Kaip buvo minėta aukščiau, skaičiavimo įrenginys pagal šį išradimą yra pritaikytas vykdyti operacijas pagal specifinę kalbą H, kuri yra apibūdinama abstrakčia sintakse ir semantika.

Sintaksė aprašo eilę skirtingų abstrakčių objektų išsireiškimų pagalba. Gali būti naudojami sekantys išsireiškimai: port, nothing, alt(list), par(list), seq(list), unify(list), during(list), cont(v), period(v), - /portas, niekas, pasirenk.(sąrašas), lygiagretus(sąrašas), seka(sąrašas), unifik.(sąrašas), laike(sąrašas), testi(v), period(v)/,

kur (list) yra nurodyta duomenų elementų pavidalu (e_1,..., e_n) objekto atminties ląstelėse, kuriose laikoma išraišką, o v yra tikras skaičius.

Kadangi kiekviena atminties ląstelė schemoje pagal šį išradimą turi ribotą skaičių reikšmės/žymėjimo laukų, sąrašas gali būti įvestas į daugelio ląstelių atmintį. Turi būti pažymėta, kad jei objekto atminties struktūra yra imituojama programos pagalba įprastame kompiuteryje, tada imituojama atminties ląstelė gali turėti kintamą atminties laukų skaičių.

Port	išsireiškimas rodo fizinį portą, t.y., during seką ir yra netiesioginio elemento specialį rūšis.
Nothing niekas	tai tam tikra reikšmė, rodanti prieštaravimą.
Alt pasirinkimas	rodo, kad atminties ląstelė, turinti tokią išraišką, turi išvardintus elementus, laikomus pasirenkamais.
Par pasirinkimas	rodo, kad atminties ląstelė, turinti tokią išraišką, turi išvardintus elementus, laikomus pasirenkamais.
Seq seka	rodo, kad atminties ląstelė, turinti tokią išraišką, turi išvardintus elementus, laikomus sekos elementais.
Unify unifikuoti	rodo, kad atminties ląstelė, turinti tokią išraišką, turi išvardintus elementus, laikomus unifikuojamais.
During trukmė	į sąvoką įeina sąrašo elementai, turintieji trukmę, arba

'@(..)	laikus, kurių metu turi kas nors atsitikti.
cont	nurodo tarpo dydį, turintį nenurodytą skaičių
tęsinys	neapibrėžtų režimų atkarpoje, kurios ilgis v.
cont	nurodo tarpo dydį, turintį nenurodytą skaičių
tęsinys	neapibrėžtų režimų atkarpoje, kurios ilgis v.
period	nurodo tarpo dydį, turintį nenurodytą skaičių
periodas	neapibrėžtų trumpų režimų periode, kurio trukmė yra v.

Sąrašo elementai yra įvedami į atmintį, kaip reikšmės, esančios reikšmės/žymėjimo laukuose.

Perrašymo galimybė yra įmontuota į centrinį valdymo bloką CU, bendraujant su atminties lastelėmis. Taigi, valdymo įrenginys įgyvendina perrašymo taisykles. Jis perrašo minėtas sintaksines išraiškas atitinkamai su nustatytomis perrašymo taisyklėmis.

Centrinis valdymo blokas yra tinkamiausias, turintis skaitmeninę ventilinę skleistinę, kuri yra sujungta su nuskaitymo ir įvedimo į atmintį išraiškomis ir užtikrinanti perrašymo operacijas atitinkamai įmontuotoms taisyklėms. Vienu metu turi būti nuskaitoma tik medžio dalis. Skaitmeninės ventilinės skleistinės, turinčios galimybę reguoti į sudėtinius įėjimus ir duoti tokius pat išėjimus atitinkamai įmontuotiems protokolams, yra aprašytos publikacijoje: Carver Mead and Lynn Conway "Introduction to VLSI Systems", Addison Wesley Publishing Company, 1980.

Perrašymo taisyklės

Perrašymo taisyklės yra sekančios:

alt

alt()	->	niekas
alt(e)	->	e
alt(e_1...nothing...e_n)	->	alt(e_1...e_n) (t.y., reikšmė nothing yra praleista)

alt(e_1...alt(c_1...c_k)...e_n) -> alt(e_1...c_1... ...c_k... e_n)

par

par(e_1...nothing...e_n) -> niekas, jei bet kuri iš reikšmių nuo e_1 iki e_n yra niekas

par(e_1..alt(c_1..c_k)..e_n) -> alt(par(e_1...c_1... ..e_n).. par(e_1..c_k..e_n))

par(e_1..e_n) -> par(e_1..e_n) kitu atv.

seq

seq(e_1...nothing...e_n) -> niekas, jei bet kuri iš reikšmių nuo e_1 iki e_n yra niekas

seq(e_1..alt(c_1..c_k)..e_n) -> alt(seq(e_1...c_1... ..e_n).. seq(e_1..c_k..e_n))

seq(e_1..e_n) -> seq(e_1..e_n) kitu atv.

unify

unify() -> niekas

unify(e) -> e

unify(e_1....e_n) -> e_1, jei visos reikšmės nuo e_1 iki e_n yra tos pačios

unify(e_1....e_n) -> niekas, jei bet kuri reikšmė nuo e_1 iki e_n skiriasi nuo kitų

unify(e_1 e_2 e_3...e_n) -> unify(unify(e_1 e_2) e_3...e_n)

unify(e_1...e_n) -> niekas, jei bet kuri iš reikšmių nuo e_1 iki e_n yra niekas

unify(e_1...alt(c_1...c_k)...e_n) -> alt(unify(e_1...c_1.. ..e_n).. unify(e_1..c_k..e_n))

Tada, kai medžio struktūra turi būti įvedama į objekto atmintį, ne mažiau, kaip viena medžio struktūros dalis turi būti pervesta į bloką 3. Tolesnės medžio struktūros apdorojimo 3 bloke galimybės bus pateiktos vėliau, aprašant konkretų prietaiso, parodyto FIG. 3, realizavimo atvejį.

Valdymo blokas CU nuskaito laukus TYPE uždarose ląstelėse, įvestus į atmintį 3 bloke atitinkamai pagal anksčiau paminėtas perrašymo taisykles ir rezultatą perduoda atgal į objekto atmintį 1. Jei perrašytas rezultatas yra suskaidytas iki reikšmės arba iki nieko, valdymo blokas CU atlieka globalinę paiešką objekto atmintyje, kad pakeistų atminties laukuose esantį turinį į suskaidytą, pažymint suskaidytos struktūros dalį, esančią tėvu.

Toliau bus pateikti pavyzdžiai, kaip vyksta operacijos, naudojant šias sintaksės taisykles. Sintaksės taisykles, kurios nėra pavaizduotos žemiau pateiktuose pavyzdžiuose, yra labai panašios į tas, kurios yra pavaizduotos ir lengvai suprantamos kvalifikuotam asmeniui.

1 PAVYZDYS

Pirmas pavyzdys, parodytas FIG. 10A iki 10H, yra lygiagrečių reikšmių unifikavimas, kuris yra pateikiamas, kaip suskaidomas uždaras reiškinys.

unify(par(1 par(1) 3) par(1 par(1) 2))

kas sudaro suskaidomą uždarą reiškinį, kuriame turi būti atlikti keli lygiagretūs unifikavimai. Šis skaidomas reiškinys turi būti perrašytas, kaip unifikuota lygiagreti struktūra.

FIG. 10A parodytas vidinis suskaidomas uždaras reiškinys. FIG. 10B parodyta, kaip šis reiškinys yra įvedamas į objekto atmintį. Atminties ląstelės, į kurias yra įvedamas skaidomas reiškinys, yra pažymetos FIG. 10A. Ryšiai tarp uždaro elemento ir

uždaros ląstelės yra pažymėti FIG. 10B. Uždara ląstelė, turinti identifikaciją id_1, turi etiketę cls ir turi tipo kodą unify,

nurodytą tipo lauke, o uždaros ląstelės, turinčios identifikaciją id_2, id_3 ir id_4 turi tipo kodą par, nurodytą tipo lauke. Uždara ląstelė, turinti identifikaciją id_1, turi du reikšmės/žymėjimo uždarus elementus, turinčius identifikavimą id_2 ir id_4. Šios uždaros ląstelės turi etiketę canon. Uždara ląstelė, turinti identifikaciją id_2, turi pirmą ir trečią reikšmės/žymėjimo uždarus elementus, turinčius diskretines reikšmes , kurios pažymėtos etikete discr, o jos antras reikšmės/žymėjimo uždaras elementas, turi identifikaciją id_3 ir turi etiketę canon. Uždara ląstelė, turinti identifikaciją id_3, turi pirmą reikšmės/žymėjimo uždarą elementą, turintį sveiką skaičių ir yra pažymėta etikete discr. Uždara ląstelė, turinti identifikaciją id_4, turi pirmą ir trečią reikšmės/žymėjimo uždarus elementus, turinčius diskretines reikšmes , kurios pažymėtos etikete discr, o jos antras reikšmės/žymėjimo uždaras elementas turi identifikaciją id_3 ir turi etiketę canon.

Kaip parodyta FIG. 10C, atminties ląstelių turinys, turintis uždarą ląstelę, identifikuotą id_1, pirmiausiai yra įvedamas į pagrindinę plokštę, patalpinant jos identifikatorių į identifikavimo registrą, kaip id_1, įskaitant tipo kodą unify ir reikšmės/žymėjimo elementus, kaip tikslą pagrindiniame registre, pirmame operacijos žingsnyje.

Kaip parodyta FIG. 10D, sūnūs, turintieji identifikatorius id_2 ir id_4, į bazinius registrus yra įvedami vertikaliai, taip, kad turinys, esąs jų pirmame reikšmės/žymėjimo elemente yra patalpinamas į pagrindinį registrą, pažymėtą identifikatoriumi, o likusieji reikšmės/žymėjimo elementai – į virš esančius registrus. Į pagrindinį registrą taip pat yra įvedami kiekvieno iš sūnų tipo kodai par. Tipo kodas yra įvedamas į registro ląsteles, esančias plokštėse TYPE.

Kaip parodyta FIG 10E, bazinių registrų turinys yra perkeliamas 90° tokiu būdu, kad jų pirmoje eilutėje esantis turinys yra patalpinamas į pagrindinius registrus, o antras vertikalus stulpelis yra patalpinamas bazinių registrų eilutėje, lygiagrečiai pagrindiniams registrams. Tipo kodai par ir unify, esantys identifikatoriaus registre, ir pagrindiniai registrai yra sukeičiami, tai yra atliekama automatiškai iš valdymo bloko. Dabar į bazinius registrus įeina tėvas, turintis tris sūnus, patalpintus į stulpelius.

Dabar kiekvienas sūnus yra pakraunamas atgal į objekto atmintį, naudojant komandą make. Kaip atsakymas, identifikatoriai iš objekto atmintyje įvestų sūnų yra perduodami į pagrindinių registrų atmintį. Galima pastebėti, kad valdymo blokas CU, būdamas ventilinės skleistinės tipo, peržiūri plokščių CLOS/SIMPLE iki TYPE registrų turinį ir duoda komandas, t.y., valdo perjungėjus ir ventilius atitinkamai pagal randama informaciją. Sūnūs yra pavadinti pagal eilės numerį, einantį po id_1, o jau užimti vardai nėra naudojami. Tačiau vardų tvarka nėra svarbi ir gali būti laisvai pasirenkama.

Kaip parodyta FIG 10F, pirmasis sūnus, gauna žymę id_2, sekantis sūnus, turintis uždarus elementus, užimančius identifikatorių id_3, gauna žymėjimą id_4, trečias sūnus gauna žymėjimą id_5. Tėvas, turintis uždarus elementus, susijusius su uždaromis ląstelėmis, turinčiomis identifikatorius id_2, id_4, id_5, gauna identifikacija id_1, ir po to yra įvedamas į objekto atmintį.

FIG 10G yra parodyta ląstelė, kurios atmintyje yra įvestas skaidomas reiškinys

par(unify(1 1) unify(par(1) par(1)) unify(2 3))

Pats skaidomas reiškinys yra parodytas FIG 10H. FIG 10G ir 10H viskas yra pavaizduota taip pat, kaip ir FIG 10A ir 10B, todėl aiškios savaime.

FIG 10G taip pat yra parodyta, kad uždaros ląstelės, turinčios tipo kodą unify, gavo nuoroda exec zonoje LAZY. o

uždara lastelė, turinti identifikatorių id_1, gavo nuoroda wait (laukti), kas reiškia, kad uždara lastelė, pažymėta exec, turi būti vykdoma prieš lastelę, pažymėta identifikatoriumi id_1, kai yra atliekamas jų suskaidymas reikšmemis.

Uždaras reiškinys, parodytas FIG 10H, vėliau gali laiko bėgyje būti pakrautas atgal į pagrindinę plokštę tolesniam apdorojimui. Pavyzdžiui, uždara lastelė, turinti identifikatorių id_2, turi reikšmę 1, nes reikšmė 1 ir 1, esantis reikšmės/žymėjimo elementuose, yra tas pats, o uždara lastelė, turinti identifikatorių id_3, pavirs į nothing, nes reikšmės 2 ir 3, esančios jų reikšmės/žymėjimo elementuose, nėra tas pats.

Kiekvienas unifikavimas siūlomame realizavimo variante gali būti atliekamas skaitmeniniame ALU (aritmetinis - loginis įrenginys), kuriame reikšmės sulyginamos komparatoriais, o lyginimo rezultatai perduodami į valdymo bloką CU. Valdymo bloke yra nustatoma ventilinė loginė skleistinė, per kurią informacija yra pateikiama į pirmąjį pagrindinį registrą, esantį pagrindinėje plokštėje. Kai skaidymas baigiasi kanoniniu priskyrimu, paprasta reikšme arba nieku, ši informacija yra globaliai paskirstoma į visas atminties zonas objekto atmintyje, paruošta antro tipo uždarų elementų įvedimui. Tokiu būdu, kiekvienas netiesioginis proskyrimas reiškinio suskaidymui yra pakeičiamas tiesioginiu reišmės žymėjimu.

2 PAVYZDYS

Šis pavyzdys yra aparatinės dalies komandų sarašo pratęsimas - list expansion, reiškiantis, kad uždara lastelė turi įterptą saraša. Šios rūšies komanda sudaro papildoma žingsnį kitų skaidymo operacijų metu.

Įrenginys atlieka skaidyma, pateikta pavyzdyje, vadinamame ex.type, kuris gali būti komandos formoje, į kurią įeina reikšmės ir sarašai, turintys forma

ex.type(1 list(2 3 list(4 5 6))7)

Ši forma yra parodyta FIG 11A, o jos vidaus ląstelės – FIG 11B. FIG 11A ir FIG 11B yra pavaizduota taip pat, kaip ir FIG 10A ir 10B, todėl aiškios savaime.

Kaip parodyta FIG 11C, uždara ląstelė, turinti identifikatorių id_1, yra įvesta į pagrindinį registrą pagrindinėje plokštėje, turinčioje jos identifikaciją ir tipo kodą, esantį identifikavimo registre. Kadangi turinys antrajame pagrindiniame registre yra pažymėtas netiesioginiu elementu open, uždara ląstelė id_2, su kuria jis yra susijęs, yra pakraunamas vertikaliai į bazinius registrus, kaip sūnus, tai yra matyti iš FIG 11D.

Po to aparatinės dalies komanda list expand perkelia diskretinę reikšmę 7 iš trečio pagrindinio registro į padėtį šalia id_4, esančią trečiame baziniame stulpelyje ir perkelia sąrašo dalį iš antrojo stulpelio, esančio virš antrojo pagrindinio registro į trečią stulpelį, patalpinant jo žemiausią elementą (reikšmę 3) į trečią pagrindinį registrą, suteikiant jam tipo kodą list (žr. FIG 11E). Kadangi antrojo pagrindinio registro turinys yra diskretinė reikšmė, jis turi etiketę discr.

Po to sudaromas sąrašo praplėtimas, perkeliant trečio stulpelio turinį, esantį virš pagrindinio registro į ketvirtą stulpelį, nurodant jo tipą, kaip list (sąrašas). Kadangi trečiojo pagrindinio registro turinys yra diskretinė reikšmė, jis turi etiketę discr, kas yra matoma iš FIG 11F.

Po to ketvirtame stulpelyje esantis sąrašas yra įvedamas į objekto atmintį, naudojantis aparatine komanda make. Jis yra įvedamas į atminties ląstelę, turinčią identifikatorių id_2, kadangi ji tapo tuščia, o identifikatorius id_2 yra persiunčiamas atgal į pagrindinę ląstelę, kad būtų įvestas į ketvirto pagrindinio registro atmintį, kaip yra parodyta FIG 11G.

Tokiu būdu yra atliekamas ex.type suskaidymas ir tada, kai yra pasiekiamas kanoninis suskaidymo rezultatas, jis yra pakraunamas atgal į objekto atmintį.

PRAKTINIS PAVYZDYS - PIRMAS MĖGINIMAS

Grįžtant prie FIG 3, pagrindinė išradimo mintis dėl porto darbo yra stebėti aplinkos 7 ir 8 savitarpio ryšį bei aktyvią objekto atmintį 1, 3, kaip unifikavimą, kurio esmė yra tame, kad kiekvienos laiko trukmės metu signalų kiekiai įgauna tik vieną apibrėžtą reikšmę.

Norint suteikti aiškumo dėl darbo eigos, pateikiame sekantį pavyzdį:

Įrenginys, turintis įėjimo ir išėjimo portus, perduoda stačiakampių signalų kiekius į išėjimą. Programa, kuri turi būti įvesta į įrenginį, yra

unify(par port$_{in}$ port$_{out}$) machine behaviour")

unifikuoti(param(portas įėj. portas išėj.) mašinos režimas,

kur "machine behaviour yra operacija alt, kuri pažymi visas sekas, kurios tik gali būti įėjime ir išėjime, t.y., sekančio pobūdžio didelės duomenų struktūros:

```
alt(seq(par(during(1s 1)during(1s 1))
        (par(during(1s 2)during(1s 4))
        (par(during(1s 3)during(1s 9))
        ...)
 (seq(par(during(1s 1)during(1s 1))
     (par(during(1s 3)during(1s 9))
     (par(during(1s 2)during(1s 4))
     ...)
 (seq(par(during(1s 2)during(1s 4))
     (par(during(1s 1)during(1s 1))
     (par(during(1s 3)during(1s 9))
     ...)
 (seq(par(during(1s 2)during(1s 4))
     (par(during(1s 3)during(1s 9))
     (par(during(1s 1)during(1s 1))
     ...)
```

```
(seq(par(during(1s 3)during(1s 9))
    (par(during(1s 1)during(1s 1))
    (par(during(1s 2)during(1s 4))
```

Kai įėjimo portas pradės perduoti reikšmes, suskaidymo operacijos, atliekamos pagrindinėje plokštėje, ištrins visas šakas operacijoje alt, kuri taps nothing, t.y., šakas, kurios netinka.

Pagrindinėje plokštėje yra atliekamas sekantis suskaidymas pagal tokias taisykles:

tokio pavidalo išraiška

unify(par($port_1$....$port_n$)
seq(par(during(q_{11} t_1) ... during(q_{1n} t_1))
.....
(par(during(q_{k1} t_k) ... during(q_{kn} t_k)
...)

yra perrašoma į

nothing

arba į išraiška

(seq(par(during((q_{11} t_1) ... during(q_{1n} t_1))
.....
par(during(q_{k1} t_k) ... (1s 2)during(q_{kn} t_k))
....)

priklausoma nuo signalų iš aplinkos ir pagal išraišką sulyginimo. Visi during, esantys kiekvienoje iš operacijų par, esančių aeq, užima tą patį laiką. Norint kad operacijos during būtų paleistos ir sustotų tuo pačiu laiku, reikia atlikti sinchronizaciją.

Kiatas kelias aukščiau nurodytų išraiškų perrašymui yra jų perrašymas į:

nothing
arba į išraiškas
seq(during(par(q_{11} ... q_{1n}) t_1)
.....

(during(par(q_{k1} ... q_{kn}) t_k)

priklausoma nuo signalų iš aplinkos ir pagal išraišką sulyginimo.

Naudojant pirmąją perrašymo taisyklę, nurodytą aukščiau pateiktame pavyzdyje, yra įvedama sekanti suskaidymo taisyklė:

unify(par(e_1 ... e_n) par(h_1 ... h_n))->
->par(unify e_1 h_1) ... unify e_n h_n)

atitinkanti seq. Kadangi įėjimo reikšmė nėra unifikuota, t.y., nėra ta pati, kaip atitinkama reikšmė, esanti įrenginyje, rezultatas yra nothing. Tai tinka ir kitiems atvejams, iki pat operacijos alt ir visos šakos ištrynimo iš išsireiškimo alt.

Bendravimas su išore

Konstruojant tokį įrenginį kyla keblumų dėl to, kad reikšmės gali būti perduodamos į arba iš įrenginio, priklausomai nuo to, ar išorinis signalas yra nustatomas ar ne, ar mašinos reikšmė yra apibrėžta, ar ne (turint reikšmę $). Įrenginys neapsiriboja vien tik aukščiau aprašyta įėjimo ir išėjimo portų sistema. Tokiu būdu, yra įmanoma panaudoti programą, kurioje portas yra pasirenkamai įrašomas arba ištrinamas. Pavyzdžiu gali būti tokia programa:

```
unify(port "machine behaviour")
turint mašinos režimą
alt(seq(during 1s 1)(during(1s 1)during 1s 2)(during(1s 4)
   (seq(during 1s 1)(during(1s 1)during 1s 3)(during(1s 9)
   (seq(during 1s 2)(during(1s 4)during 1s 1)(during(1s 1)
   (seq(during 1s 3)(during(1s 9)during 1s 2)(during(1s 4)
ir tt.
```

t.y.,

(seq(during input)(during output)(during input)(during output)

Šiame įrenginio režime reikšmė yra perduodama iš aplinkos, o įrenginys atsako, sekančio intervalo metu perduodamas jos kvadratą. Šis metodas pilnai kontroliuoja, kad duomenys eitų paprastu būdu.

Blokas 3 bendrauja su portais ir yra pritaikytas porto įėjimus įvesti į objekto atmintį eilės tvarka.

Tai yra atliekama tokios formos išraiškos pagalba:

unify(port seq(during t_1 q_1) .... seq(during t_k q_k) ...)

Kai ši išraiška yra suskaidoma, centrinio valdymo bloko CU protokolas yra toks, kad jis šią išraišką perrašo į nieką arba į išraišką

seq(during $newt_1$ $newq_1$)... seq(during $newt_k$ $newq_k$) ...

priklausomą nuo signalo, ateinančio iš aplinkos, sulyginimo su išraiška.

Išraiškoje $newt_1$ yra naujos trukmės laikas, o $newq_1$ yra naujas signalo kiekis i-tame intervale, esančiame reikšmių porų sekoje (žr. FIG 2).

Jei signalas porte turi formą, parodytą FIG 2, aukščiau esanti išraiška unify 3 bloke bus perrašyta į nieką, taip pat, jei signalo reikšmė q_1 skirsis nuo atitinkamos reikšmės q_1, esančios išraiškoje unify, arba, jei signalo reikšmė t_1 skirsis nuo atitinkamos reikšmės t_1, esančios išraiškoje unify. Priešingu atveju, išraišką unify yra perrašoma į išraišką seq. Išraiškoje unify, kaip minėta aukščiau, turi būti pažymėtas ypatingas simbolis $, rodantis, kad reikšmė turi būti pasirenkama.

Jei reikšmės t_1 arba q_1, kai i yra laisvai pasirenkamas sveikas skaičius ir kai jos turi žymę $, tada atitinkami $newt_1$ arba $newq_1$ bus perrašomi, įterpiant normuotus dydžius t_1 arba q_1.

Atitinkamai, jei atėjęs iš aplinkos signalas yra neapibrėžtas, o reikšmės t_1 ir q_1 iš išraiškos unify yra tam tikro dydžio, tada visas signalas yra perduodamas, kaip q_1

skaitmeninio - analoginio keitiklio $CONV_{OUT}$ I portą (žr. FIG 6) per laiką t_1, tuo tarpu, kai $newt_1$ ir $newq_1$ iš išraiškos seq yra tas pats, kaip ir reikšmės t_1 ir q_1 išraiškoje unify.

Papildomi H kalbos dariniai

Visi galimi režimai gali būti aprašomi pagal aukščiau nurodytą sistemą. Nelaimei, programos gali būti labai didelės. Tam, kad būtų galima įveikti šitokią situaciją, čia gali panaudoti sekantys kalbos dariniai. Svarbiausi iš jų yra symb, lambda ir apply.

Pilna H kalboje naudojama sintaksė gali turėti sekančias išraiškas:

cont(v), delta, period(v), deltat, par(list), seq(list), 'alt(list), 'con(list), 'pri(list), lambda(list), 'hide(list), 'symb(list), 'unify(list), 'set(list), 'apply(list), alt(list), nothing, con(list), pri(list), unify(list), lambda(list), hide(list), set(list), apply(list), symb(list), discr(n), pulses(n),

kur sintaksės išraiškos yra pateikiamos, kaip operatoriai užkoduotu pavidalu atminties lastelių zonose type.

Tokiu būdu, kiekviena išriška yra operatorius, įvestas į objekto atmintį, kaip ir kiekvienas elementas, esantis (list) yra sąrašo elementas, rodantis režimą, kai n yra sveikas skaičius, o v - realus skaičius. kiekvienas elementas gali būti įvedamas į atminties lastėles, turinčias minėtą išraišką, kuriai priklauso sąrašas.

Išraišką cont(v), kurioje v yra realus skaičius, kuris rodo dydį, turintį neapibrėžtą neapibrėžtumų skaičių atstume, kurio ilgis yra v. Išraiška period(v), kurioje v yra realus skaičius ir žymi laiko trukmę, turinčią neapibrėžtą neapibrėžtų režimų

skaičių per laikotarpį, trunkantį v.

Išraiška delta pažymi režimą, trunkantį atomo eilės trukmę laike ir turintį neapibrėžtą trukmę erdvėje.

Kiekybių, kaip sudėtinių dalių, išraiška

Skaitmenų eilė erdvėje gali būti koduojama, kaip delta parametro par dalis, sekančiu būdu:

0=par()

0=par(delta)

0=par(delta delta)

...

5=par(delta delta delta delta delta)

ir tt.

Skaitmenų trukmė laike gali būti užkoduojama, kaip par iš deltat, tokiu būdu.

Tokiu būdu, skaičius (discr(n)), manoma, esas plokščias darinys, susidedantis iš atominės eilės dydžių delta.

Labai svarbus kiekybių panaudojimas yra jų, kaip elementų, panaudojimas operacijoje during. Šioje operacijoje kiekybės yra naudojamos nurodyti tiek laiką, tiek ir erdvę. Tam yra naudojamas deltat, atitinkantis delta laike.

delta erdvėje užima atominį dydį ir yra neapibrėžtas laike

deltat yra neapibrėžtas erdvėje ir turi atominį dydį laike.

Virš nurodytos išraiškos yra laikomos objekto atmintyje tokiu pačiu būdu, kaip ir sintaksinės išraiškos.

Programos, kaip kalbos; išraiškos, kuriose yra ' .

Irenginys iš karto suskaido visas ıterptas išraiškas ı jų reikšmes. Pavyzdžiui, išraiška

apply(+ par(2 4))

yra tiesiogiai suskaidoma ı jos reikšmę 6. Dėl to yra neımanoma išrinkti programą, kai ji yra tik instaliuota. Neımanoma suskaidyti jos dalimis ir atlikti skaičiavimus su dalimis.

Tuo tikslu, kad būtų ımanoma tokius veiksmus atlikti, šalia bazinės kalbos yra ıvedama pasyvi lygiagreti kalba. Tai yra vadinama programos formomis. Jos susiskaido savitarpyje (yra kanoninės) ir toliau neveikia. Tokiu būdu, jos gali būti suskaidytos ı gabalus.

Funkcija plusrevarg gali būti apibūdinama sekančiu būdu:

unify(symb(plusrevarg)

lambda(par(y x) 'apply(+ par(x y)))

kas programai suteikia pavidalą, kuris atitinka aukščiau nurodytą programą

'apply(+ par(x y))

duoda mums plius argumentą priešinga tvarka, par(4 2).

Norint, kad ıvestas ı atmintı programos pavidalas būtų paverstas ı atitinkamą programą, yra naudojama standartinė funkcija eval. Tokiu būdu, programa yra nukopijuojama ir po to yra pašalinamos dokumentom žymės. Po to programa paprastai yra skaidoma iki jos reikšmių.

apply(symb(eval) 'apply(+ par(x y)))=apply(+ par(2 4)))=6

Programų pavidalo pertvarkymo galimybė supaprastina programų peržiūrėjimą ir supaprastina interpretatorių ir kompiliatorių rašymą kitoms bet kurios rūšies kalboms, naudojantis ırenginiu, kuris kaip bazinę, turi H kalba.

Pagrindinė taisyklė yra tai, kad visos konstrukcijos (pvz., delta, par) turi antrąją formą; vykdoma per ' , kaip viena iš leksikos elementų (pvz., delta, par), nurodančių formą, kaip

programos forma. Svarbu pažymėti, kad vien tik tikra konstrukcija, o ne visas medis yra programos forma. Norint nurodyti visą medį, kaip programos forma, visos konstrukcijos turi būti pažymėtos '.

Jei ι programos formos sąrašą , turintį ', įeina nothing, tada išraiška bus paversta ι nothing.

Programų formavimas

Tokie mechanizmai, reikalingi pažymėjimui, kaip sqr, yra aprašyti aukščiau. Tačiau, yra reikalingas patogus mechanizmas programų formavimui, susidedantis iš mažų pažymėjimui. Mes norime sukurti programą, kaip pagrindinę išraišką, turinčią eilę papildomų apibrėžimų. Tai yra padaroma su con(list).

Con

Konstrukcija con turi vizualią reikšmę v ir keletą nematomų verčių e_i, kur i yra sveikas skaičius tarp 1 ir n.

con(v e_1 e_2 e_3...e_n)

Ši struktūra turi reikšmę, lygią pirmam elementui v. Tačiau, jei sąrašas yra tuščias arba bet kuris iš sąrašo elementų yra nothing, konstrukcija con turi reikšmę nothing. Visi sąrašo elementai yra tos pačios apimties. Nematomi elementai gali būti naudojami laisvų kintamųjų apribojimui arba kaip rišantieji elementai, arba unifikavimui ι reikšmę nothing.

Con perrašymo taisyklės yra tokios:

con()	->nothing
con(e_1...nothing...e_n)	->nothing
con(e_1....e_n)	->e_1

Kaip ir visoms taisyklėms, ši taisyklė yra sudaryta bendravimo tarp aktyvios atminties 1, 3 ir centrinio valdymo bloko CU pamatu.

Pri

Nei viena iš aukščiau aprašytų konstrukcijų negali būti naudojama nurodyti, kad kas nors yra leidžiama tada, kai kas nors neleidžiama (yra nothing). Kadangi negatyvus išraiškos būdas kai kada yra žymiai kompaktiškesnis, negu pozityvus, neigimo būdui yra pridedama konstrukcija pri.

Suskaidymo, naudojant pri taisyklės yra tokios, kad pri reikšmė yra pirmasis elementas, kuris skiriasi nuo nothing. Jeigu tokio elemento nėra, pri bus niekas. Tokiu būdu,

pri(nothing nothing nothing 7 8)=9

Konstrukcija pri(e_1 e_2) gali būti interpretuojama sekančiu būdu: jei e_1 neleidžiama, tai turi būti naudojama e_2.

Pri konstrukcija primena konstrukcija alt. Tai yra režimų rinkinys, tačiau pri atveju režimai yra tam tikroje rikiuotėje. Pirmas režimas sąraše eina pirmuoju, antras - antruoju ir tt.

Svarbu pažymėti, kad konstrukcija pri nieko bendro neturi su prioritetiškumo koncepcija, naudojama įprastinėse kalbose ir įrenginiuose, tvarkant procesą. Vietoj to yra įvedamas neigimas atsisakymo būdu.

Pri perrašymo taisyklės yra tokios:

pri() ->nothing

pri(nothing$_1$...nothing$_n$) ->nothing

pri(nothing$_1$...nothing$_{k-1n}$ e_k...e_n) ->e_1

Kaip ir visoms taisyklėms, ši taisyklė yra sudaryta bendravimo tarp aktyvios atminties 1, 3 ir centrinio valdymo bloko CU pamatu.

Apimties taisyklės

Apimtis yra tekstinė aplinka, kurioje visi su tuo pačiu simboliu susiję įvykiai reiškia tą pačią semantinę sąvoką. Apimties taisyklė sako, kada ir kaip tam tikra konstrukcija veda į naują apimtį. H kalba turi sekančios apimties taisykles:

1. Konstrukcijos par, seq, alt, @ ir unity neįveda naujų apim%dių.Tai reiškia, kad visi su tuo pačiu simboliu susiję įvykiai reiškia tą pačią sąvoką.
2. Konstrukcija lambda (pakeičia pattern) įveda naują priklausomą apimtį. Visi simbolio pasirodymai, kurie nėra paminėti ankstesnėje apimtyje, priklauso naujai apimčiai.
3. Konstrukcija hide (išraiška) įveda papildomą naują apimtį. Nei vienas simbolio pasirodymas konstrukcijoje hide nėra tos pačios sąvokos išraiška toje apimtyje.

Hide

Kai yra ruošiamos praplečiančios programos, reikia panaudoti tuos pačius simbolius skirtingomis prasmėmis. Tai galima padaryti, naudojant konstrukciją (hide), kurios dėka galima įvesti skirtingas apimtis.

Labai svarbi paslėpimo koncepcija yra hide, kaip objekto, panaudojimas ir tam tikro objekto apibūdinimas hide pagalba. Skaičiavimo įrenginio pagal šį išradimą vartotojas visa tai gali rasti programiniame pakete, bibliotekoje arba instaliavimo aprašymuose.

Programinė H kalba apibūdina paslėpimo konstrukciją hide(e). Ji nurodo režimą. Laisvi kintamieji, naudojami konstrukcijos viduje ir išorėje, yra atskirti, t.y., jie nėra matomi vienas iš kito. Ši H kalbos savybė yra skirtinga nuo kitų rūšių kalbų, kurios iš esmės perkelia duomenų tipus ir eilę simbolių pavadinimų. Tokie pavadinimai negali būti panaudoti paketo viduje. Tai reiškia, kad globalinė simbolių sritis yra iš tikrųjų globalinė visiems paketams ir yra matoma jų viduje, pilnai arba ribotu mastu.

Tipinis programinis paketas yra nurodomas sekančiu būdu: lambda(a_pkg $ $).hide(...vietinis instaliavimas...);

Tokiu būdu yra nustatoma taisyklė, turinti režimą, prasidedantį nuo a_pkg, o visa kita nenurodoma.

Tai yra tik taisyklės nurodymas, kur simboliai, neapimami tos taisyklės, yra nežinomi.

Biblioteka yra paketas, esantis H kalbos koncepcijoje, naudojant parametrą, parenkantį vieną arba keletą taisyklių. Jis gali būti instaliuojamas, kaip

```
hide(
     alt(lambda "rule a".a-spec
         lambda "rule b".b-spec
          .....
         lambda "rule n".n-spec))
```

Biblioteka yra alternatyva, talpinanti taisykles. Modelis "rule a" yra naudojamas, norint surinkti programą a spec. Tai gali būti apibūdinama, kaip pasirenkama konstrukcija , tačiau yra bendrai abstraktus modelis arba taisyklė.

Kaip buvo parodyta aukščiau, H kalba turi elegantišką metodą, pasirenkant programinę įrangą pagal reikiamą inžinerinį poreikį. Kalbai nereikia įvesti naujų kalbos konstrukcijų dėl šio tikslo.

Reikia pastebėti, kad paslėpimas daro įtaką tik operacijoms su apimtimi.

lambda

Išraiška lambda(e_m e_{p1}...e_{pn}) yra taisyklė, išreiškianti tai, kad elementas e_m pakeičia e_{p1} ...e_{pn}; čia e_r yra taisyklė lambda(e_m e_{p1}...e_{pn}) ir minėta išraiška apply(e_r e_{arg1}...e_{arg2}) yra pritaikymas, kuris suskaidymo operacijos metu yra pakeičiamas e_m, jei kiekvienas e_{p1} yra unifikuojamas su atitinkamu e_{arg1}, arba, priešingai, pakeičiamas nieku.

apply

Toliau bus pateiktas pavyzdys, kaip yra įvedama taisyklė apply.

PAVYZDYS

Turi būti atliekama skaitmeninė komanda. Skaitmeninė komanda gali būti +, −, *, /, during, ir tt. Po komandos seka argumentai. Šiame pavyzdyje yra atliekamas sudėties veiksmas tarp skaičių, esančių sąraše. Įrenginys atlieka apply (pritaikymas) suskaidymą pagal formą

apply(+ list(1 2))

Pritaikymas yra parodytas FIG 12A, o jo uždaros lasteles – FIG 10A ir 10B, kuriose schemos paaiškina viena kitą.

Kaip yra parodyta FIG 12C, į pagrindinį registrą, turintį savo identifikatorių ir tipo kodą, yra pakraunamos uždaros lastelės, turinčios identifikatorių id_1. Kaip komanda, yra duodama nuoroda (+). Kadangi antrame pagrindiniame registre esantis turinys turi etiketę, kaip netiesioginis elementas open, uždara lastelė, su kuria jis yra susijęs, yra pakraunama vertikaliai, kaip sūnus, į bazinius registrus, kaip matoma iš FIG 12D.

Po to yra atliekamas sąrašo praplėtimas, uždedant etiketę discr ant diskretinės reikšmės, esančios antrame pagrindiniame registre, ir sąrašo praplėtimo reikšmę 2 žymint list kodo tipo lauke. Tai yra daroma todėl, kad įrenginys atlieka tą pačią operaciją, nepaisant to, ar sąrašas, turintis identifikatorių id_2 turi du, tris ar keturis elementus. Kadangi naujame sąraše yra tiktai vienas elementas, įrenginys perkelia žymę list su nuoroda, kad pagrindinis registras turi reikšmę, kuri yra discr, kas yra matoma iš FIG 12F.

Tada į pagrindinį registrą įeina komandos žymė (+) ir dvi diskretinės reikšmės, o tai priverčia valdymo bloką, kuriame yra saugomos komandos, valdyti skaitmeninį ALU, kad būtų atlikta komanda (sudėtis) ir pirmame pagrindiniame registre būtų gautas skaitmeninės operacijos rezultatas, kaip kanoninė reikšmė, matoma

FIG 126. Reikia pastebeti, kad nuoroda apply, esanti kodo tipo lauke yra žymėjimas, kad turi būti atlikta pritaikymo funkcija.

Rezultato reikšmė, šiuo atveju, paprasta reikšmė 3, po to yra perskirstoma globaliai tuo tikslu, kad būtų išvengtas bet koks identifikatoriaus id_1 atsiradimas ties ta reikšme.

Perrašymo taisyklės

Perrašymo taisyklėse nurodomos išraiškos, kurios yra lygios. Perrašymo kryptis yra susijusi su šiomis taisyklėmis. Prisilaikant perrašymo krypties, gaunamas paprastesnis reiškinys. Kai reiškinys nebegali būti perrašomas į paprastesnį, jis yra kanoninis. Tokiu būdu, skaičiavimo įrenginys pagal šį išradimną naudoja perrašymo taisykles reškinių perrašymui į kanonines formas.

Be aukščiau minėtų perrašymo taisyklių, centrinis valdymo įrenginys CU įgyvendina ir sekančias perrašymo taisykles:

type(e_1...nothing...e_n) -> nothing, kai type rodo išraišką, esančią saraše virš par(list) iki unify(list), o e_1...e_n yra sarašo elementai.

type(e_1...alt(c_1...c_k)...e_n) ->
alt(type(e_1...c_1...e_n)...type(e_1...c_k...e_n))
kur type rodo išraišką, esančią saraše virš par(list) iki unify(list) (except alt(list)), o e_1...e_n, e_1...e_n yra sarašo elementai.

set() ->nothing

set(a) -> 'alt(a), jei a yra išraiška iš par(list) iki apply(list), virš esančių išraiškų saraše.

set(alt(e_1...e_n)) -> alt(e_m...e_n), kai e_1...e_n yra elementai, nepateikti kokia nors tvarka, o e_m...e_n yra surikiuoti.

set(e

$_1 \ldots e_n$) -> set(con($e_1 \ldots e_n$))

lambda(alt($e_1 \ldots e_n$)) -> set(alt($e_1 \ldots e_n$))

lambda($e_1 \ldots e_n$) -> par($e_1 \ldots e_n$)

hide(alt($e_1 \ldots e_n$)) -> set(alt($e_1 \ldots e_n$))

hide($e_1 \ldots e_n$) -> par($e_1 \ldots e_n$)

apply() -> nothing

apply(e) -> $eval_{meta}$(e)

apply('alt ($b_1 \ldots b_n$)$a_2 \ldots a_m$) -> apply($eval_{meta}$ 'alt($b_1 \ldots b_n$)$a_2 \ldots a_m$)

apply(par($b_1 \ldots b_n$)$a_2 \ldots a_m$) -> nothing, jei visos b_i išraiškos yra iš par(list) iki 'apply(list), esančio virš, o n#m arba bet kuriam iš a_i#b_i tinka i>1.

apply(par($b_1 \ldots b_n$)$a_2 \ldots a_m$) -> b_1, jei visos b_i ir a_i išraiškos yra iš par(list) iki 'apply(list), esančio virš, o n=m arba bet kuriam iš i>1 a_i=b_i.

apply(seq($b_1 \ldots b_n$)$a_2 \ldots a_m$) -> nothing, jei visi b_i ir a_i yra iš par(list) iki 'apply(list), esančio virš, o n#m arba bet kuriam iš a_i#b_i tinka i>1.

apply(seq($b_1 \ldots b_n$)$a_2 \ldots a_m$) -> b_1, jei visos b_i ir a_i išraiškos yra iš par(list) iki 'apply(list), esančio virš, o n=m arba bet kuriam iš i>1 a_i=b_i.

Funkcija $eval_{meta}$ išraiškas iš programinės formos (pvz., suteikia ') paverčia į išraiškos formą (pvz., be '). Vien tik aukščiausia struktūra turi pašalintą '.

$eval_{meta}$'type(a) -> type(a), kai type rodo išraišką, esančią saraše virš par(list) iki unify(list) išskyrus alt(list).

symb

Išraiška symb(s), į kurią įeina elementas s, kuris, skaidant į kanoninę išraišką, yra identifikatoriumi, minėtas elementas turi galimybę būti įvestas į atmintį atminties zonose, priklausančiose įvairioms atminties ląstelėms minėtoje objekto atmintyje, minėtos atminties ląstelės surikiuojamos jų apimtyje, visi identifikatoriai s kiekvienoje apimtyje žymi tą patį režimą.

Discr, pulses

Išraiška discr(n), kurioje n yra sveikas skaičius, pažymi lygiagretų režimą, turintį daugelį n atominės eilės režimų. Išraiška pulses(n), kurioje n yra sveikas skaičius, pažymi n atominės eilės režimų laiką.

Galioja sekančios perrašymo taisyklės:

par(delta$_1$....delta$_n$) -> discr(n)

seq(deltat$_1$....deltat$_n$) -> pulses(n)

PRAKTINIS PAVYZDYS - ANTRASIS BANDYMAS

Tuo tikslu, kad būtų pateiktas pavyzdys, kaip gali būti panaudota išplėstinė sintaksė įrašymui arba nuskaitymui tame pačiame porte, gali būti pateikta kompaktiškesnė programa. Pirmiausiai yra nurodoma pastaba square (sqr) kas diskretinėmis reikšmėmis 1, 2 ir 3, naudojantis sekančia trumpa programa:

unify(symb(agr)alt(lambda(1 1) lambda(4 2)lambda(9 3))

saugoma objekto atmintyje 1, uždarų reiškinių medyje. Tada mašinos režimas gali būti parašytas vien tik kaip viena operacija seq:

```
seq(during(1s x)during(1s apply(symb(sqr) x)
   (during(1s x)during(1s apply(symb(sqr) x)...)
```

Ši operacija seq vykdymo metu bus perrašyta į operaciją

alt parodytą aukščiau. Net ir ši programa yra ant tiek ilga, kaip ir duomenų įvedimas. Galimybė šį perocesą valdyti elegantiškesniu būdu yra aparatinėje dalyje sudaryti eilę funkcijų ir standartinių alternatyvų, arba jas pakrauti drauge su dominančia programa.

Pavyzdžiu gali būti sandauga (x) ir šia standartine operacija atliekamas a reikšmės pakėlimas kvadratu. Tai gali būti parašyta, kaip paprasta programa:

```
unify(symb(sqr)  lambda(symb(a)  apply(x
par(symb(a)symb(a))))
```

Šios programos dėka galima pakelti kvadratu visus apatinėje dalyje esančius skaičius.

Ši taisyklė gali būti naudojama mažų rekursyvinių programų rašymui, kurios, jei jas parašius pagal suskaidymo taisykles, yra išplečiamos į plačias alt konstrukcijas, nurodytas aukščiau. Aparatinė dalis yra tokia, kad nereikalingas išplėtimas yra praleidžiamas. Tai paprasčiausiai yra padaroma, naudojant pagrindinę plokštę tokios formos, kokia yra aprašyta pateiktoje paraiškoje Nr PCT/SE91/00514.

Skaičiavimo įrenginio, turinčio galimybę atlikti nurodytas perrašymo taisykles, režimas, naudojant režimų unifikavimą, yra apibūdinamas, tiksliai nurodant užimamą laiką. Laikinas aprašymas yra panašus į aprašomąją kalbą.

Programų nuskaitymo gerinimas

Abstrakčioji sintaksė yra labai paprasta. Dėl to ji, deja, darosi nelabai lengvai skaitoma. Tuo tikslu, kad programos būtų lengviau skaitomos, yra įvedama labiau komplikuota sintaksė, nes žmogus, kaip individas, nelabai mėgsta naudotis per daug paprastais dalykais.

Ši transformacija gali būti atlikta daugeliu įvairių būdų. Tačiau yra natūralu turėti tekstinę ir grafinę formą.

Klaida

Klaidos indikacija yra pateikiama tada, kai aparatinė dalis aptinka, kad kokia nors reikšmė yra neteisinga, pavyzdžiui, tikrinant lyginius bitus skaičiavimų metu arba panašiai. Po to yra naudojama išraiška backup.

Suskaidymo taisyklės backup atvejui yra tokios, kad backup reikšmė yra pirmas elementas, kuris skiriasi nuo fault. Jei tokio elemento nėra, backup bus niekas.

Perrašymo taisyklės backup atveju yra sekančios:

backup() -> fault

backup($fault_1 \ldots fault_n$) -> fault

backup($fault_1 \ldots fault_{k-1}\ e_k \ldots e_n$) -> e_k

Kaip ir visose taisyklėse, šios taisyklės yra instaliuotos ir sudaro ryšį tarp aktyvios atminties 1, 3 ir centrinio valdymo bloko.

Keletas kalbos pavyzdžių

Kaip aiškėja iš aukščiau pateikto aprašymo, H kalba turi keletą variantų. Kalba turi savo pagrindinę dalį H_{abs}, kuri yra abstrakti kalba, o jos pamatu yra sudaryta kalbų šeima. Pagrindiniai kalbos variantai yra šie:

H_{abs} Šis kalbos variantas yra labiausiai bendras. Jis yra aprašomojo pobūdžio, tačiau ši kalba neturi sintaksės, kuri būtų prieinama vartotojui.

H_{fault} Šis kalbos variantas yra praplėsta H_{abs} kalba, turinti konstrukcijas aparatinės dalies klaidų modeliavimui skaičiavimo įrenginyje. Šis kalbos variantas nėra aprašomojo pobūdžio. Vietoj rezultato, ji pateikia statistinę reikšmę. Šis variantas yra pagrindinės kalbos variantas, kuria remiasi visos likusios

kalbos. Ši kalba neturi sintaksės, kuri būtų prieinama vartotojui.

H_{user} Šis kalbos variantas yra H_{fault} praplėtimas, turintis instaliuotus derinius veiksmams su skaičiais ir operatorius skaitmeninei ir struktūrinei aritmetikai atlikti. Į ją taip pat įeina neapibrėžti reiškiniai. Kalba neturi sintaksės, kuri būtų prieinama vartotojui. Šis kalbos variantas yra abstrakti kalba, vizualiai prieinama vartotojui.

H_{ascii} Tai kalbos variantas, naudojantis normalius ascii ženklus ir prefikso žymėjimą visoje sintaksėje, išskyrus labiausiai bendras išraiškas, naudojant sintaksę su intarpais.

Kadangi išradimas yra aprašytas, duodant nuorodas į tam tikrą realizavimo variantą, kvalifikuotam asmeniui reikia suprasti, kad gali būti padaryti įvairūs pakeitimai ir padaryti ekvivalentiški pakeitimai kai kurių elementų, nenutolstant nuo tikros išradimo idėjos ir požiūrio. Be to, gali būti atlikti patobulinimai, nenutolstant nuo išradimo pagrindinių teiginių.

A P I B R Ė Ž T I S

1. Skaičiavimo įrenginys skaidymo tipo, skirtas programų, įvestų į atminties elementus (Cu, 1), apdorojimui, į kurį įeina daug atmintiers ląstelių (2), skaičiavimo įrenginys, taip pat turintis ne mažiau, kaip vieną porto elementą (4, 5, 6, 11) ir ne mažiau, kaip vieną išorinės aplinkos priemonę, c h a r a k t e r i z u o j a m a s tuo, kad

a) atminties priemonės (Cu, 1) yra aktyvios, taip, kad kiekviena iš daugelio atminties ląstelių (2) turi informaciją, kuri įgalina pradėti operacijos vykdymą,

b) ne mažiau, kaip vienas porto elementas (4, 5, 6, 11) yra prijungtas prie aktyvių atminties priemonių (Cu, 1) ir turi sulyginimo priemones (SEL_1, SEL_2, Cu_1, Cu_2), skirtas sulyginti unifikavimo metu duomenis, pateikiamus iš aplinkos, vienoje minėto porto elementų pusėje, turinčio atminties ląsteles su įvestais į vieną iš minėtų atminties elementų (2) duomenimis, kurie sulyginimo metu yra pateikiami į kitą minėto porto elementų pusę, ir

c) duomenys, esą bet kurioje minėtų porto elementų pusėje gali turėti nenustatytus duomenų elementus ($) arba dalinai nustatytus duomenų elementus (alt).

2. Skaičiavimo įrenginys pagal požymį 1, c h a r a k t e r i z u o j a m a s tuo, kad minėti porto elementai yra pritaikyti signalų eilės sulyginimui ir turi priemones (CU, 3; Cu_1, Cu_2), skirtas sulyginamos sekos perrašymui į ką nors, kas rodo defektą, jei sulyginimas parodo aišku skirtumą, arba kita, skirta lyginamos sekos perrašymui į tam tikrą seką.

3. Skaičiavimo įrenginys pagal požymius 1 ir 2, c h a r a k t e r i z u o j a m a s tuo, kad sulyginimo elementai (SEL_1, SEL_2; Cu_1, Cu_2) yra pritaikyti sulyginimui grupėmis, susidedančiomis iš ne mažiau, kaip dviejų elementų (V_{DU}, t_{DU}), turinčių nustatytą skaičių sąrašo elementų.

4. Skaičiavimo įrenginys pagal požymį 2, c h a r a k t e r i z u o j a m a s tuo, kad turi priemones (7, 8, 9; T_{COMP}, SEL_1, T_{COUNT}, SEL_2, $CONV_{IN}$, $CONV_{OUT}$), skirtas išrinktos signalų sekos pakeitimui laike, turinčiame individualius išrinkimo periodus, o signalų seką sudaro elementų grupių sąrašas, kur į kiekvieną grupę įeina trukmės laikas (t_1, t_2 ir tt.) ir ne mažiau, kaip vienas signalo kiekybinis požymis (q_1, q_2 ir tt.) tame laikotarpyje.

5. Skaičiavimo įrenginys pagal požymį 4, c h a r a k t e r i z u o j a m a s tuo, kad nustatytas sąrašo elementų skaičius kiekvienoje grupėje, pateiktoje į sulyginimo priemones, yra poromis, kiekviena iš kurių turi laiko ir signalo kiekio derinį (t_1, q_1; t_2, q_2 ir tt.).

6. Skaičiavimo įrenginys pagal požymius nuo 1 iki 5, c h a r a k t e r i z u o j a m a s tuo, kad turi priemones sinchronizuoti vieną iš porto elementų su ne mažiau vienu iš kitų porto elementų ir perduoti lygiagretų signalų kiekio portuose indikavimą kiekvienos laiko trukmės metu.

7. Skaičiavimo įrenginys pagal požymius nuo 1iki 6, c h a r a k t e r i z u o j a m a s tuo, kad atminties elementų priemonės yra patalpintos aktyvioje dalyje ir yra porto elementai yra pritaikytos įvesti į atmintį kompiuterinę programą, esančią aiškioje arba numanomoje kodavimo formoje abstrakčiaja sintakse, kuri aprašo eilę abstrakčių objektų išraiškų pagal-

ba, kiekviena atminties ląstelė turi galimybę vienu metu saugoti dalį iš vienos sintaksinės išraiškos tinkamų duomenų ir/arba programinės struktūros pavidalu.

8. Skaičiavimo įrenginys pagal požymį 7, c h a r a k t e r i z u o j a m a s tuo, kad atminties ląstelių priemonėse kiekviena sintaksės reiškinių rūšis taip pat turi atitinkamos rūšies sintaksinių išraiškų, rodančių, kad tai yra programos forma, o visos programų formos yra sintaksinės išraiškos, kurios yra pritaikytos suskaidymui savitarpyje ir tokiu būdu užima savo vietą.

9. Skaičiavimo įrenginys pagal požymį 9, c h a r a k t e r i z u o j a m a s tuo, kad turi perrašymo priemones (CU), esančias saveikoje su atminties ląstelėmis, įskaitant sintaksės išraiškas jų perrašymui atitinkamai nustatytoms perrašymo taisyklėms.

10. Skaičiavimo įrenginys pagal požymį 9, c h a r a k t e r i z u o j a m a s tuo, kad į sintaksę įeina bent keletas sekančių sintaksinių išraiškų:
port, nothing, alt(list), par(list), seq(list), unify (list), during(list), cont(v), period(v), kur (list) yra duomenų elementų $(e_1,...e_n)$ sarašas, kiekvienas iš kurių yra įvestas į vienos iš ląstelių atmintį, turinčią išraišką, kuriai priklauso sarašas, jei atminties ląstelėse yra vietos sarašui arba atminties ląstelėse, turinčiose ryšį su atminties ląstelėmis, laikančiomis tas išraiškas, kur v yra realus skaičius, o į saraša, esantį during(list) gali įeiti sekantys elementai: cont(v), period(v) ir pasirenkama reikšmė ($), ir kur išraiška port reiškia vieną iš fizinių portų ir yra speciali rūšis netiesioginių elementų, nothing yra speciali rūšis reikšmės, reiškiančios

neatitikimą, alt, kad į atminties ląsteles, turinčias šią išraišką, įeina saraše esantys diskretiniai elementai, laikomi alternatyviais elementais, par, kad į atminties ląsteles, turinčias šią išraišką, įeina saraše esantys diskretiniai elementai, laikomi lygiagrečiais elementais, seq, kad į atminties ląsteles, turinčias šią išraišką, įeina saraše esantys diskretiniai elementai, laikomi nuosekliais elementais, unify, kad į atminties ląsteles, turinčias šią išraišką, įeina saraše esantys diskretiniai elementai, laikomi unifikuojančiais elementais, during, kad į atminties ląsteles, turinčias šią išraišką, įeina pora, turinti laiko trukmę ir signalo kiekį, cont(v) pažymi erdvės kiekį, turintį neapibrėžtą neapibrėžtų režimų skaičių atkarpos dalyje, turinčioje ilgį v, o period pažymi laiko trukmę, turinčią neapibrėžtą skaičių neapibrėžtai trumpų režimų periodo, turinčio trukmę v.

11. Skaičiavimo įrenginys pagal požymius 9 ir 11, c h a r a k t e r i z u o j a m a s tuo, kad perrašymo taisyklės yra parenkamos iš sekančių tarpo:

nothing	-> nothing
alt(e)	-> e
alt(e1...nothing...en)	-> alt(e1...en) (t.y., reikšmė nothing yra praleista)
alt(e1...alt(c1...ck)...en)	-> alt(e1...c1... ...ck... en)
par(e1...en)	-> niekas, jei bet kuri iš reikšmių nuo e1 iki en yra niekas
par(e1..alt(c1..ck)..en)	-> alt(par(e1..c1..en) ..par(e1..ck..en))

par(e1..en)	-> par(e1...en) kitu atveju
seq(e1..en)	-> niekas, jei bet kuri iš reikšmių nuo e1 iki en yra niekas
seq(e1...alt(c1...ck)...en)	-> alt(seq(e1..c1..en) ..seq(e1..ck..en))
seq(e1..en)	-> seq(e1...en) kitu atveju
unify()	-> niekas
unify(e)	-> e
unify(e1 e2)	-> e2, jei e1=e2
unify(e1 e2)	-> niekas, jei e1 skiriasi nuo e2
unify(e1 e2 e3....en)	-> unify(unify(e1 e2)e3....en)
unify(e1...en)	-> niekas, jei bet kuri iš reikšmių nuo e1 iki en yra niekas
unify(e1...alt(c1...ck)...en)	-> alt(unify(e1...c1.. ..en).. unify(e1..ck..en))

12. Skaičiavimo įrenginys pagal požymį 8, c h a r a k t e- r i z u o j ą m a s tuo, kad į sintaksę įeina ne mažiau kaip viena iš sintaksės išraiškų, parinktų iš sekančių:

cont(v), period(v), deltat, par(list), seq(list), 'alt(list), 'con(list), 'pri(list), lambda(list), 'hide(list), 'symb(list), 'unify(list), 'set(list), 'apply(list), alt(list), nothing, con(list), pri(list), unify(list), lambda(list), hide(list), set(list), apply(list), symb(list), discr(n), pulses(n),

kur sintaksės išraiškos yra operatorius, saugomas kaip išraiška vienoje iš objekto atminties ląstelių, o kiekvienas

elementas, esantis (list) yra sąrašo elementas, rodantis režimą, kai n yra sveikas skaičius, o v - realus skaičius. kiekvienas elementas gali būti įvedamas į atminties ląsteles, turinčias minėtą išraišką, kuriai priklauso sąrašas; kur išraiška cont(v) nurodo erdvės kiekį, turintį neapibrėžtą neapibrėžtumų skaičių atstume, kurio ilgis yra v; kur išraiška period(v) pažymi laiko trukmę, turinčią neapibrėžtą neapibrėžtai trumpų režimų skaičių per laikotarpį, trunkantį v; kada išraiška delta pažymi režimą, trunkantį mažiausią trukmę erdvėje, turintį neapibrėžtą laiko trukmę; kur išraiška deltat pažymi režimą, turintį mažiausią trukmę laike ir neapibrėžtą trukmę erdvėje; kada išraiška par(list) pažymi tai, kad į atminties ląsteles, turinčias šią išraišką, įeina sąraše esantys diskretiniai elementai, laikomi lygiagrečiais elementais; kur išraiška seq(list) pažymi tai, kad į atminties ląsteles, turinčias šią išraišką, įeina sąraše esantys diskretiniai elementai, laikomi nuosekliais elementais, kur visos išraiškos, turinčios leksikos elementą ', nurodo programos formą; kur išraiška alt(list) rodo, kad į atminties ląsteles, turinčias šią išraišką, įeina sąraše esantys diskretiniai elementai, laikomi alternatyviais elementais; kur išraiška nothing yra specialios rūšies reikšmė, reiškianti neatitikimą; kur į išraišką con(list) įeina sąrašas, turintis nulį ir dar keletą režimo elementų ir yra identiškas pirmam elementui sąraše, jei visi elementai sąraše yra kanoniniai ir skiriasi nuo nothing; kur į išraišką pri(list) įeina sąrašas, turintis nulį ir dar keletą režimo elementų ir yra identiškas pirmam kanoniniam elementui sąraše ir skiriasi nuo nothing; kur išraiška unify(list) rodo, kad į atminties ląsteles, turinčias šią išraišką, įeina sąraše esantys diskretiniai elementai, laikomi unifikuojančiais elementais; kur išraiška lambda(e_m e_{p1}...epn) yra taisyklė, reiškianti, kad elementas e_m yra pakeičiamas e_{p1}...epn; kur išraiška apply(list) yra naudojama skaitmeninės matematikos veiksmams pagal funkcijos komandą

(+-*/ ir tt.), kaip jos pirma elementą, o kiti elementai yra argumentais; jei e_r yra taisykle lambda(e_m e_{p1}...epn), išraiška apply(e_r e_{arg1}...eargn) yra pritaikymas, kuris suskaidymo operacijoje yra pakeičiamas e_m, arba, kitu atveju, yra pakeičiamas nieku; kur išraiška hide(list) pažymi režimą, kur pilni kintamieji, naudojami konstrukcijos viduje arba išorėje yra atskirti, t.y., nematomi vienas kito atžvilgiu; išraiška set(list) pažymi, kad sarašo turinys bus pervestas į programos formą; kur išraiška symb(s), į kurią įeina elementas s, kuris, suskaidytas į kanoninę išraišką, tampa identifikatoriumi, elementas s turi galimybę būti įvestas į atmintises skiltis, priklausančias skirtingoms atminties lastelems objekto atmintyje, atminties lasteles apibūdinamos taip, kad būtų galima sutvarkyti jų apimtį, visi identifikatoriai s kiekvienoje iš apimčių pažymi tą patį režimą; kur išraiška discr(n), kurioje n yra sveikas skaičius, pažymi erdvės kokybę, turinčią eilę atominės eilės režimų; kur išraiška pulses(n), kurioje n yra sveikas skaičius, pažymi n atominės eilės režimų laiko trukmę.

13. Skaičiavimo įrenginys pagal požymius 9 ir 11,

c h a r a k t e r i z u o j a m a s tuo, kad perrašymo taisyklės yra praplečiamos sekančiomis taisyklėmis:

type(e_1...nothing...e_n) -> nothing, kai type rodo išraišką, esančią saraše virš par(list) iki unify(list), o e_1...e_n yra sarašo elementai.

type(e_1...alt(c_1...c_k)...e_n) ->
alt(type(e_1...c_1...e_n)...type(e_1...c_k...e_n))
kur type rodo išraišką, esančią saraše virš par(list) iki unify(list) (except alt(list)), o e_1...e_n, e_1...e_n yra sarašo elementai.

set() ->nothing

set(a) -> alt(a), jei a yra išraiška iš par(list) iki apply(list),

	virš esančių išraiškų saraše.
set(alt($e_1 \ldots e_n$))	-> alt($e_m \ldots e_n$), kai $e_1 \ldots e_n$ yra elementai, nepateikti kokia nors tvarka, o $e_m \ldots e_m$ yra surikiuoti.
set($e_1 \ldots e_n$)	-> set(con($e_1 \ldots e_n$))
lambda(alt($e_1 \ldots e_n$))	-> set(alt($e_1 \ldots e_n$))
lambda($e_1 \ldots e_n$)	-> par($e_1 \ldots e_n$)
hide(alt($e_1 \ldots e_n$))	-> set(alt($e_1 \ldots e_n$))
hide($e_1 \ldots e_n$)	-> par($e_1 \ldots e_n$)
apply()	-> nothing
apply(e)	-> $eval_{meta}$(e)
apply('alt ($b_1 \ldots b_n$)$a_2 \ldots a_m$)	-> apply($eval_{meta}$ 'alt($b_1 \ldots b_n$)$a_2 \ldots a_m$)
apply(par($b_1 \ldots b_n$)$a_2 \ldots a_m$)	-> nothing, jei visos b_i išraiškos yra iš par(list) iki 'apply(list), esančio virš, o n#m arba bet kuriam iš $a_i \# b_i$ tinka i>1.
apply(par($b_1 \ldots b_n$)$a_2 \ldots a_m$)	-> b_1, jei visos b_i ir a_i išraiškos yra iš par(list) iki 'apply(list), esančio virš, o n=m arba bet kuriam iš i>1 $a_i = b_i$.
apply(seq($b_1 \ldots b_n$)$a_2 \ldots a_m$)	-> nothing, jei visi b_i ir a_i yra iš par(list) iki 'apply(list), esančio virš, o n#m arba bet kuriam iš $a_i \# b_i$ tinka i>1.
apply(seq($b_1 \ldots b_n$)$a_2 \ldots a_m$)	-> b_1, jei visos b_i ir a_i išraiškos yra iš par(list) iki 'apply(list), esančio virš, o n=m arba bet kuriam iš i>1 $a_i = b_i$.

Funkcija $\underline{eval}_{meta}$ išraiškas iš programinės formos (pvz., turinčias ') paverčia į išraiškos formą (pvz., be '). Vien tik aukščiausia struktūra turi pašalinta '.

$\underline{eval}_{meta}$'type(a)	-> type(a), kai <u>type</u> rodo išraišką, esančią saraše virš <u>par</u>(list) iki <u>unify</u>(list) išskyrus <u>alt</u>(list)
con()	-> nothing
con(e1...nothing...en)	-> nothing
con(e1...en)	-> en
pri()	-> nothing
pri(nothing1...nothingn)	-> nothing
pri(nothing$_1$...nothing$_{k-1}$ ek...en)	-> $\underline{ek}$
<u>par</u>delta$_1$....delta$_n$	->discr(n)
<u>seq</u>(deltat$_1$....deltat$_n$)	->pulses(n)

14. Skaičiavimo įrenginys pagal požymius nuo 1 iki 13, c h a r a k t e r i z u o j a m a s tuo, kad kiekviena atminties ląstelė yra pritaikyta įvedimui į atmintį bent kai kurias savybes, nurodytas sekančiai:

žymes, rodančias, ar ląsteleje esanti išraiška turi būti suskaidyta,

žymes, rodančias, ar išraiška yra medžio dalis bei išraiškos savybes,

žymes, rodančias, kaip yra sudaryta išraiška,

žymes, rodančias, ar išraiška sudaro eilę pasikartojančių režimų,

žymes, rodančias, ar išraiška yra tiktai sarašo dalis , turinti kitus sąrašo narius, įvestus į kitas atminties ląsteles.

15. Skaičiavimo įrenginys pagal požymius nuo 1 iki 14, c h a r a k t e r i z u o j a m a s tuo, kad jį įeina priemonės,

esančios atminties elementuose, objekto atmintyje, tam, kad būtų įmanoma įvesti į aminti kompiuterines programas, esančias aiškioje formoje ir užkoduotas abstrakčiaja sintakse, virš jos esant suskaidymo taisyklių semantikai.

16. Skaičiavimo įrenginys pagal požymį 15, c h a r a k t e r i z u o j a m a s tuo, kad į semantiką įeina eilė perrašymo taisyklių, kiekviena iš kurių turi išraišką ir reikšmių grupę, reiškiančias, kad kiekviena reikšmė grupėje yra laikoma ekvivalentine kitai reikšmei grupėje ir gali būti sukeista su ja.

17. Skaičiavimo įrenginys pagal požymius nuo 1 iki 16, c h a r a k t e r i z u o j a m a s priemonėmis, skirtomis atlikti pastovų klaidų tikrinimą ir turintis rezervinį veikimą pagal sekančias perrašymo taisykles:

backup() -> fault (klaida)

backup(fault1...faultn) -> fault (klaida)

backup(fault1...faultk-1 ek...en) -> ek.

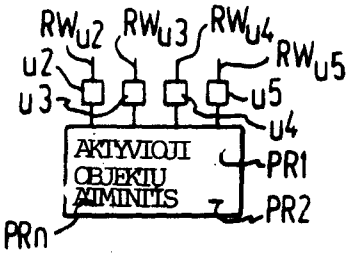

FIG. 1

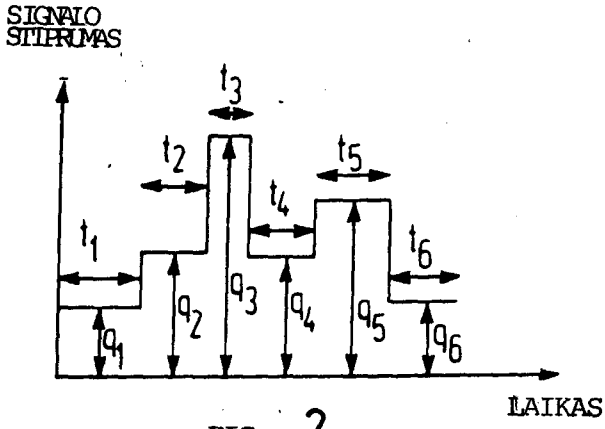

FIG. 2

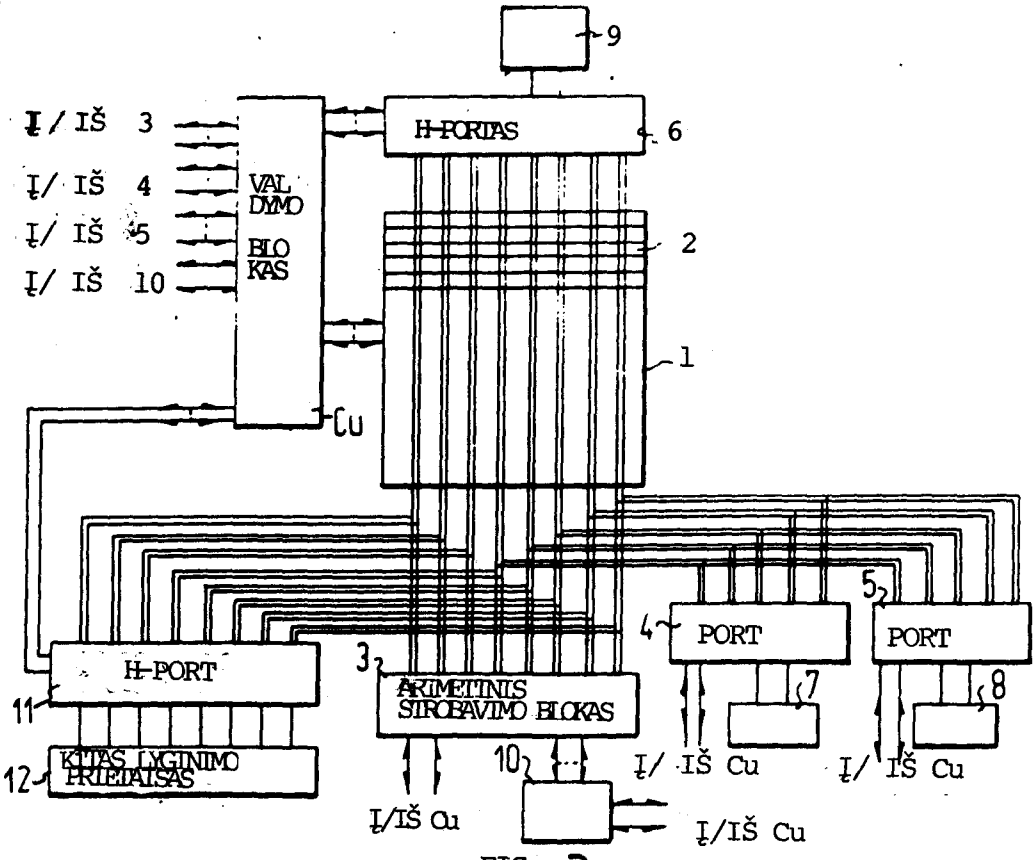

FIG. 3

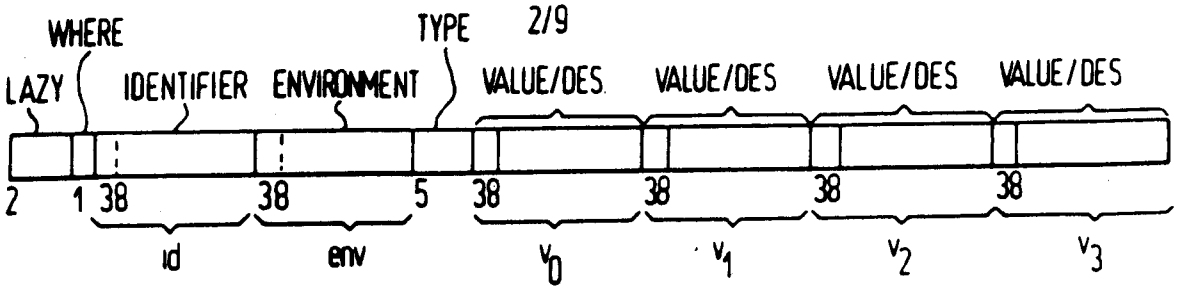

FIG.4

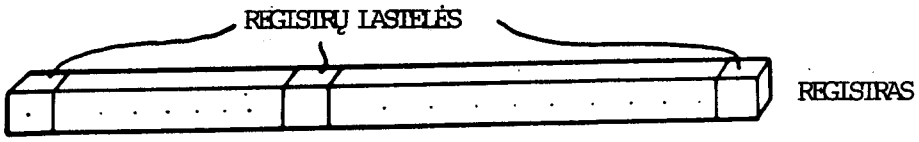

FIG.8A

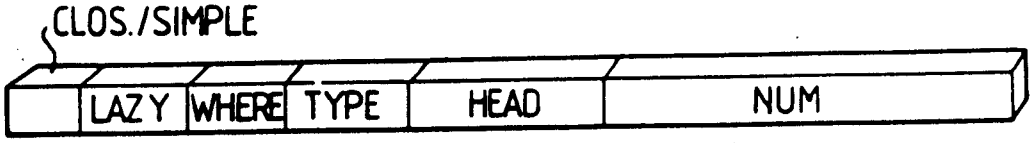

FIG. 8B

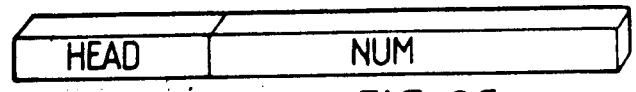

FIG. 8C

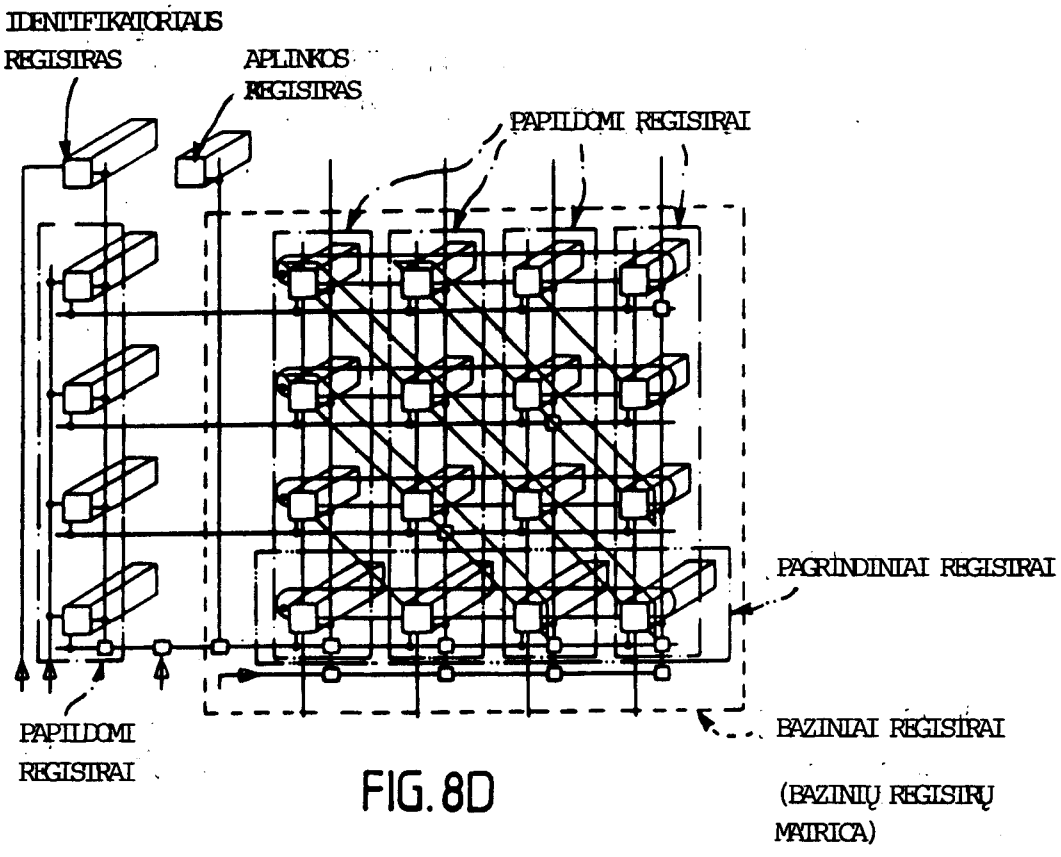

FIG. 8D

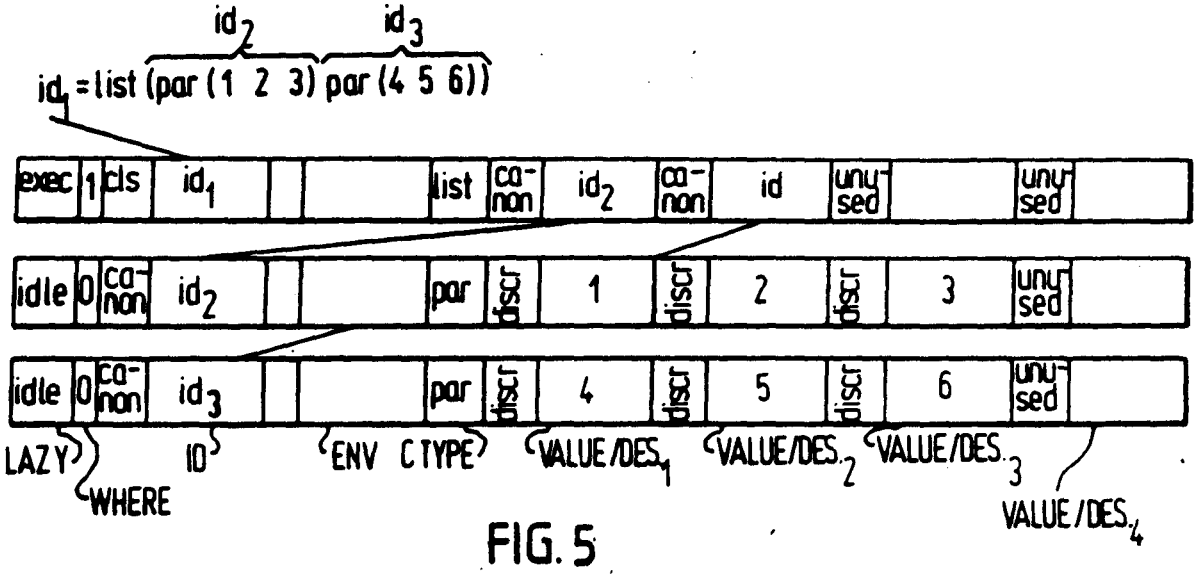

FIG. 5

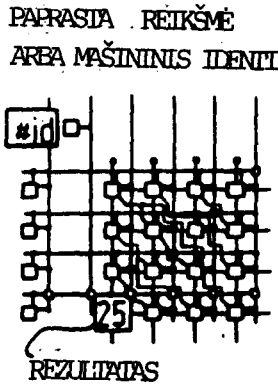

FIG. 9A

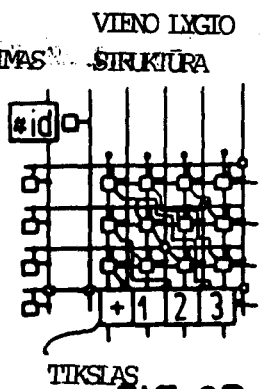

FIG. 9B

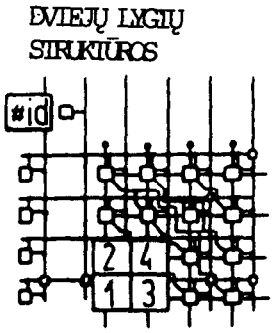

FIG. 9C

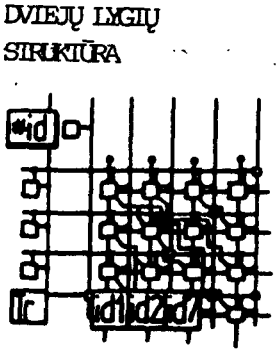

FIG. 9D

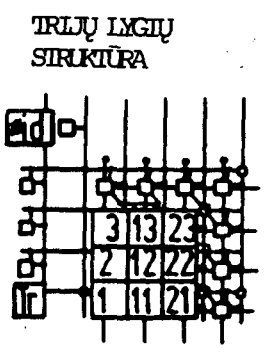

FIG. 9E

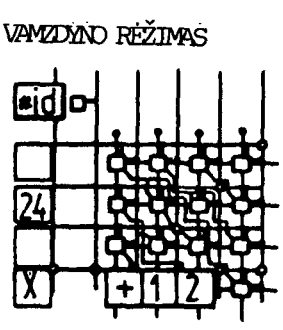

FIG. 9F

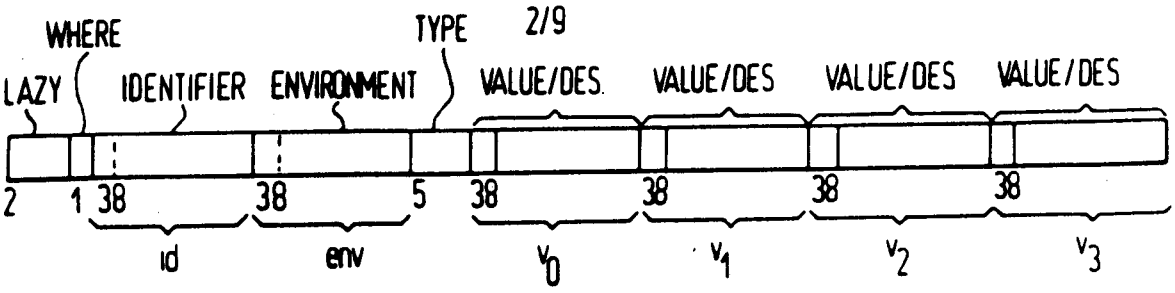

FIG.4

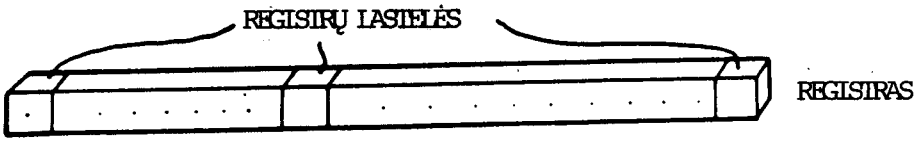

FIG.8A

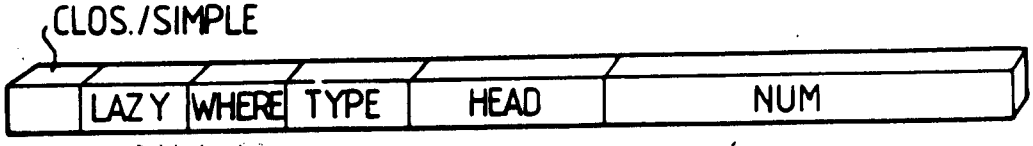

FIG. 8B

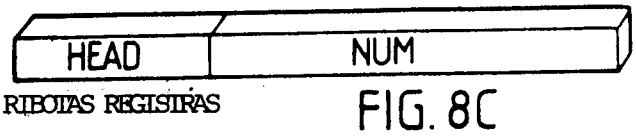

FIG. 8C

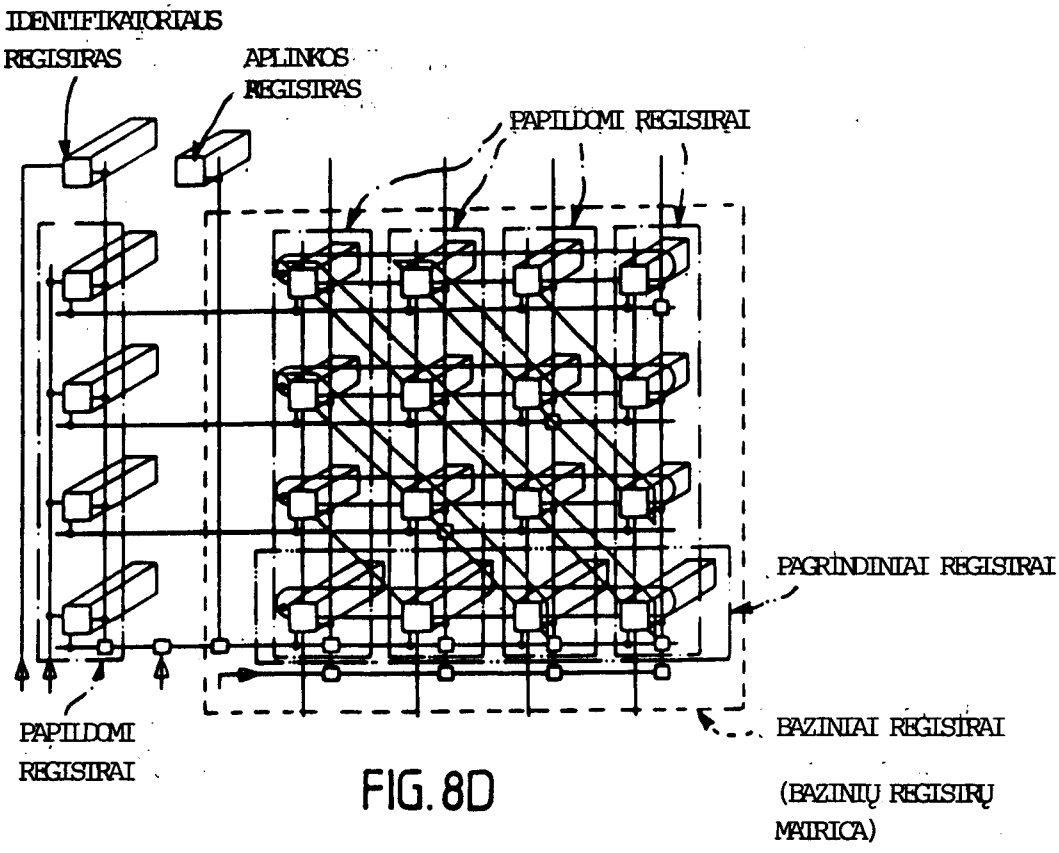

FIG. 8D

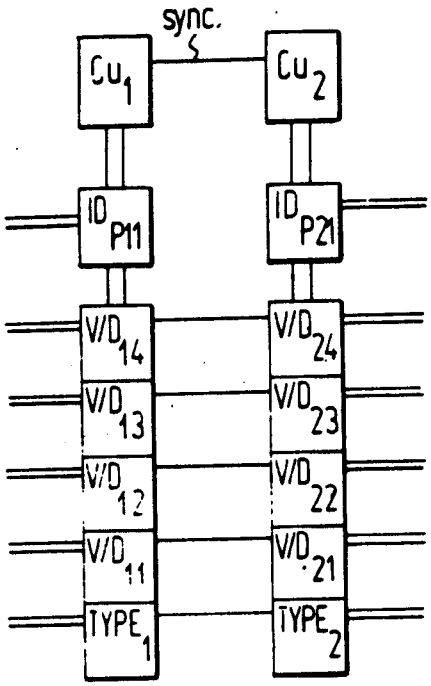
sync.
Cu1
Cu2
ID P11
ID P21
V/D14
V/D13
V/D12
V/D11
TYPE1
V/D24
V/D23
V/D22
V/D21
TYPE2

FIG.7

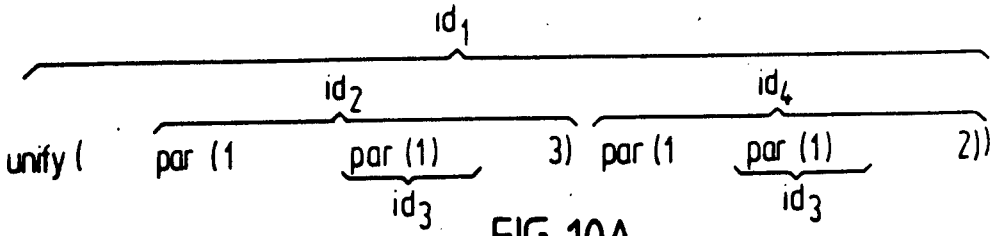

FIG. 10A

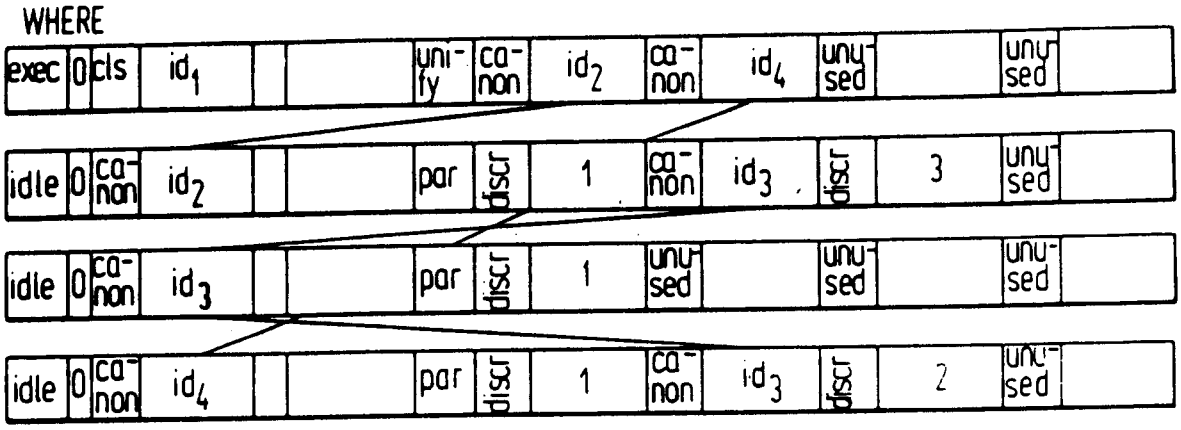

FIG. 10B

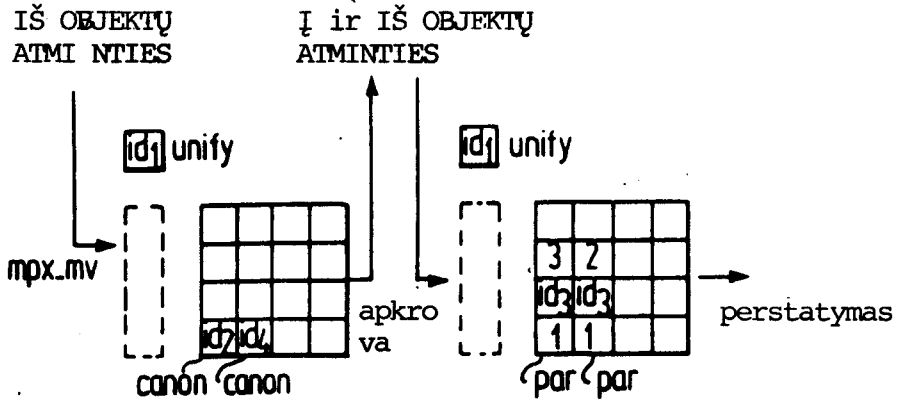

FIG. 10C

FIG. 10D

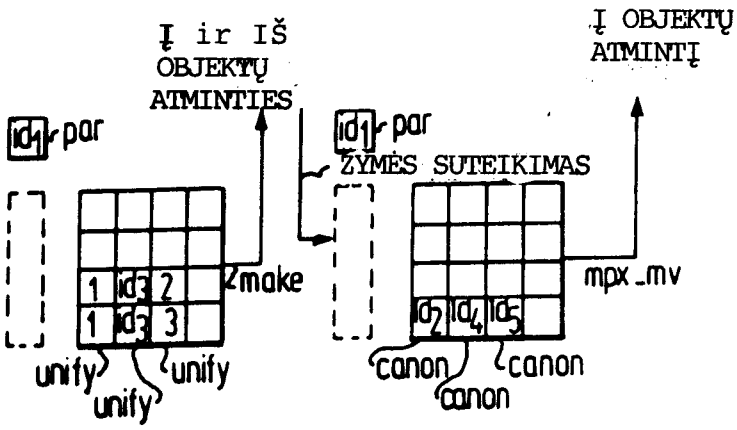

FIG.10E FIG.10F

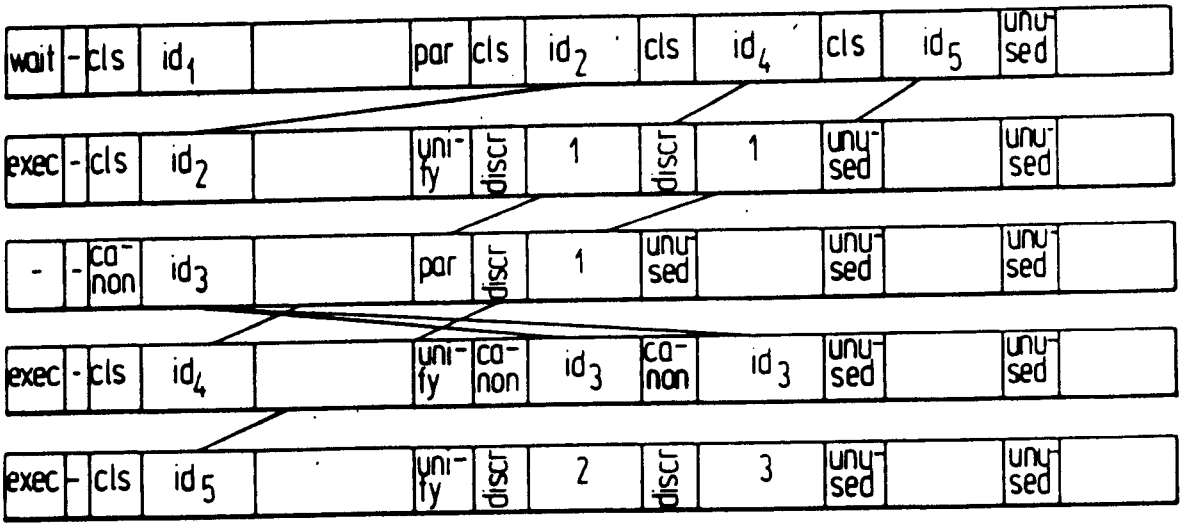

FIG. 10G

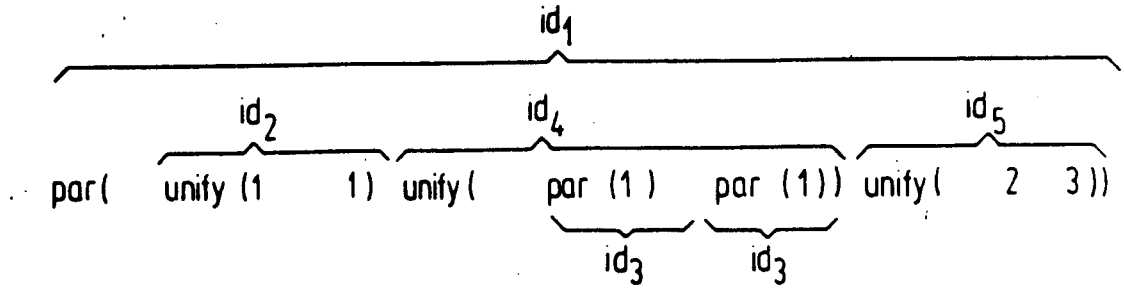

FIG.10H

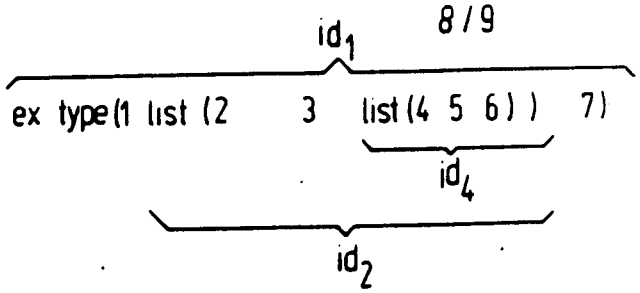

FIG. 11A

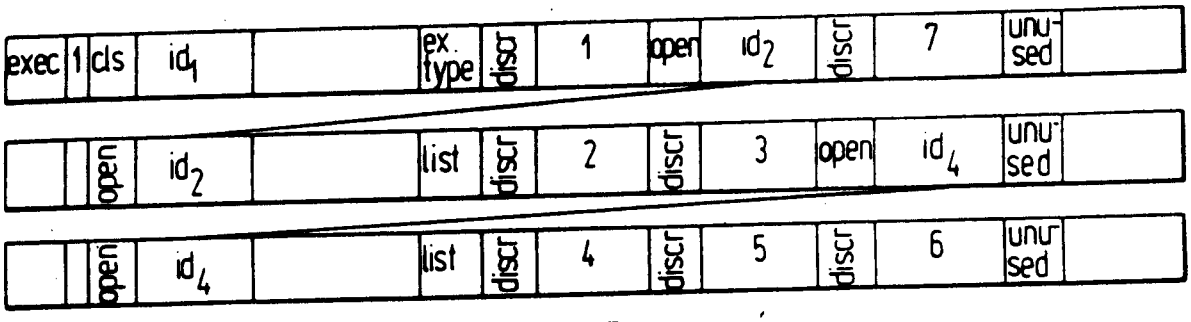

FIG. 11B

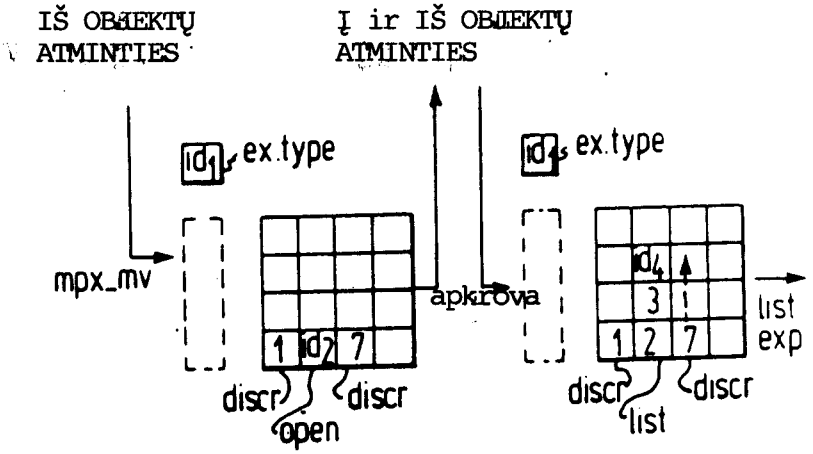

FIG. 11C FIG. 11D

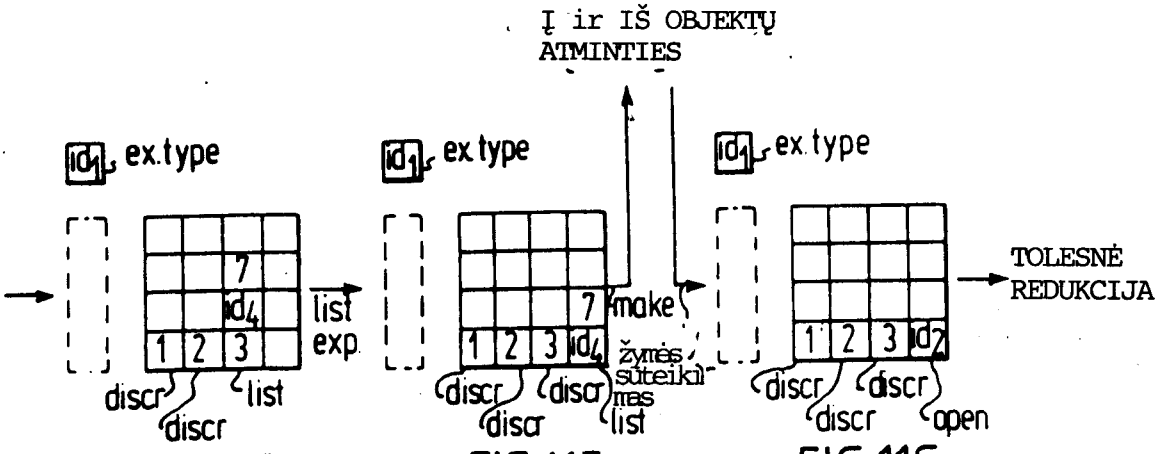

FIG. 11E FIG. 11F FIG. 11G

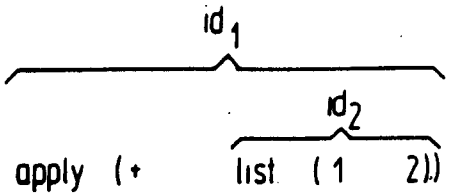

FIG. 12A

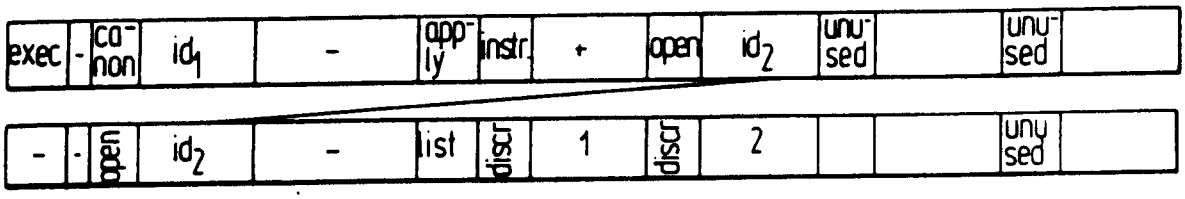

FIG. 12B

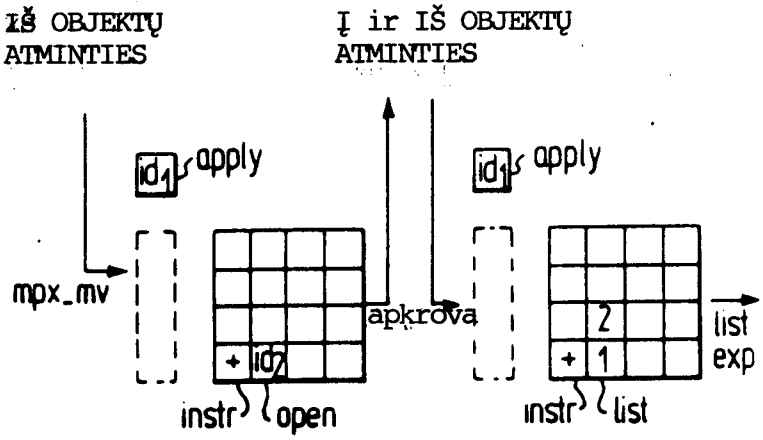

FIG. 12C FIG. 12D

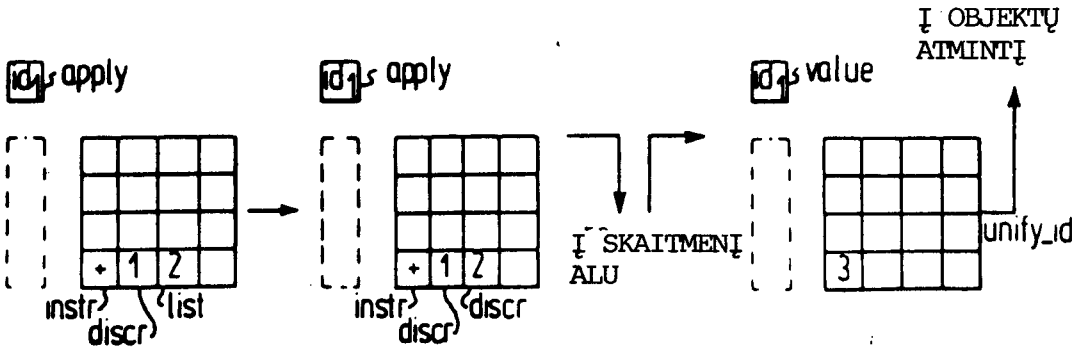

FIG. 12E FIG. 12F FIG. 12G