ԵՐԵՎԱՆԻ ՊԵՏԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆ

Ղազարյան Գոռ Արամի

# ԱՌԱՆՑ ՏԻՊԵՐԻ ՖՈՒՆԿՑԻՈՆԱԼ ԾՐԱԳՐԵՐԻ ՁԵՎԱՓՈԽՈՒԹՅՈՒՆՆԵՐԻ ԵՎ ՆՐԱՆՑ ՊՐՈՑԵԴՈՒՐԱՅԻՆ ՍԵՄԱՆՏԻԿԱՆԵՐԻ ՄԱՍԻՆ

Ա.01.09 - «Մաթեմատիկական կիբեռնետիկա և մաթեմատիկական տրամաբանություն» մասնագիտությամբ ֆիզիկամաթեմատիկական գիտությունների թեկնածուի գիտական աստիճանի հայցման ատենախոսության

## ՍԵՂՄԱԳԻՐ

Երևան-2011

---

ЕРЕВАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Казарян Гор Арамович

# О ПРЕОБРАЗОВАНИЯХ БЕСТИПОВЫХ ФУНКЦИОНАЛЬНЫХ ПРОГРАММ И ИХ ПРОЦЕДУРНЫХ СЕМАНТИКАХ

## А В Т О Р Е Ф Е Р А Т

диссертации на соискание ученой степени кандидата физико-математических наук по специальности 01.01.09 – “Математическая кибернетика и математическая логика”

Ереван-2011

Ատենախոսության թեման հաստատվել է Հայ-Ռուսական (Սլավոնական) համալսարանում

| | | |
|---|---|---|
| Գիտական ղեկավար՝ | ֆիզ.մաթ. գիտ. դոկտոր | Ս.Ա. Նիգիյան |
| Պաշտոնական ընդդիմախոսներ՝ | ֆիզ.մաթ. գիտ. դոկտոր | Յու.Մ. Մովսիսյան |
| | ֆիզ.մաթ. գիտ. թեկնածու | Ս.Ա. Ավետիսյան |
| Առաջատար կազմակերպություն՝ | ՀՀ ԳԱԱ Ինֆորմատիկայի և ավտոմատացման պրոբլեմների ինստիտուտ | |

Պաշտպանությունը կայանալու է 2011 թ. մայիսի 20-ին, ժ. $15^{30}$-ին ԵՊՀ-ում գործող ԲՈՀ-ի 044 «Մաթեմատիկական կիբեռնետիկա և մաթեմատիկական տրամաբանություն» մասնագիտական խորհրդի նիստում, հետևյալ հասցեով՝ 0025, Ալեք Մանուկյան 1:

Ատենախոսությանը կարելի է ծանոթանալ ԵՊՀ-ի գրադարանում:

Սեղմագիրն առաքված է 2011 թ. ապրիլի 19–ին:

Մասնագիտական խորհրդի գիտական քարտուղար՝
ֆիզ.մաթ. գիտ. թեկնածու Վ.Ժ. Դումանյան

---

Тема диссертации утверждена в Российско-Армянском (Славянском) университете

| | | |
|---|---|---|
| Научний руководитель: | доктор физ.-мат. наук | С.А. Нигиян |
| Официальные оппоненты: | доктор физ.-мат. наук | Ю.М. Мовсисян |
| | кандидат физ.-мат. наук | С.А. Аветисян |
| Ведущая организация: | Институт проблем информатики и автоматизации НАН РА | |

Защита состоится 20 мая 2011 г. в $15^{30}$ часов на заседании действующего в ЕГУ специализированного совета ВАК 044 “Математическая кибернетика и математическая логика” по адресу: 0025, г. Ереван, ул. Алека Манукяна, 1.

С диссертацией можно ознакомиться в библиотеке ЕГУ.

Автореферат разослан 19 апреля 2011 г.

Ученый секретарь специализированного совета,
кандидат физ.-мат. наук В.Ж. Думанян

**Թեմայի արդիականությունը:** Այս աշխատանքը վերաբերում է ֆունկցիոնալ ծրագրավորման խնդիրներին: Հետազոտության առարկան առանց տիպերի ֆունկցիոնալ ծրագրերն են, որոնք սահմանվում են որպես անջատվող փոփոխականներով հավասարումների համակարգեր առանց տիպերի $\lambda$-հաշվում[1,2]: Ֆունկցիոնալ ծրագրերի անշարժ կետի սեմանտիկան (հիմնական սեմանտիկան) սահմանվում է անշարժ կետի $Y$ կոմբինատորի միջոցով [3,4]: Հիմնականում ֆունկցիոնալ ծրագրերի ինտերպրետատորները հիմնվում են երկու տեսակի գործողությունների վրա՝ տեղադրություն և միաքայլ $\beta$-ռեդուկցիա: Ապացուցված է, որ պրոցեդուրային սեմանտիկաները, հիմնված այդպիսի ինտերպրետացիայի ալգորիթմների վրա, անհակասելի են[2]: Երբ պրոցեդուրային սեմանտիկաները, հիմնված այդ ինտերպրետացիայի ալգորիթմների վրա, համընկնում են ծրագրերի հիմնական սեմանտիկայի հետ, այդ դեպքում նշված պրոցեդուրային սեմանտիկաները համարվում են լրիվ, իսկ հակառակ դեպքում՝ ոչ լրիվ:

[2,5]–ում նկարագրված են ինտերպրետացիայի ալգորիթմների $SNFR$ (տեղադրում և նորմալ ձևի բերում) դասը, նրա կոնկրետ օրինակներ՝ $LSNFR$ ալգորիթմը (ձախ տեղադրում և նորմալ ձևի բերում), $RSNFR$ ալգորիթմը (աջ տեղադրում և նորմալ ձևի բերում), $FSNFR$ ալգորիթմը (լրիվ տեղադրում և նորմալ ձևի բերում), ինչպես նաև $HNFR$ (գլխային նորմալ ձևի բերում), $ACT$(ակտիվ) և $PAS$(պասիվ) ալգորիթմները: Սույն աշխատանքում սահմանվում է $ACT$ ալգորիթմի ձևափոխված տարբերակը՝ $ACT^*$-ը, որը, ի տարբերություն $ACT$-ի, արգումենտի հաշվարկը կատարում է, երբ այն պետք է տեղադրվի առնվազն մեկ փոփոխականի ազատ մուտքի փոխարեն: $LSNFR$, $RSNFR$, $FSNFR$ ալգորիթմները ինչպես նաև $ACT$, $ACT^*$, $PAS$, $HNFR$ կհամարենք հիմնական ինտերպրետացիայի ալգորիթմներ, իսկ պրոցեդուրային սեմանտիկաները, հիմնված այդ ալգորիթմների վրա, հիմնական պրոցեդուրային սեմանտիկաներ: [2,5]–ում ապացուցված է, որ պրոցեդուրային սեմանտիկաները հիմնված $PAS$ և $HNFR$ ալգորիթմների վրա լրիվ են, իսկ $ACT$, $SNFR$ ալգորիթմների վրա հիմնված պրոցեդուրային սեմանտիկաները՝ ոչ լրիվ: Սույն աշխատանքում ցույց է տրվում, որ $ACT^*$-ի վրա հիմնված պրոցեդուրային սեմանտիկան նույնպես ոչ լրիվ է:

[2,5] աշխատանքներում ապացուցված է, որ հիմնական ոչ լրիվ պրոցեդուրային սեմանտիկաները (բացառությամբ $ACT^*$-ի) զույգ առ զույգ անհամեմատելի են: Հետաքրքրություն է առաջացնում այդ խնդրի ուսումնասիրումը մեկ հավասարումից կազմված ծրագրերի դասում:

---

[1] Barendregt H.P. The lambda calculus, its syntax and semantics. Amsterdam: North Holland Pub. Comp., 1981.

[2] Nigiyan S.A., Avetisyan S.A. Semantics of Untyped Functioanal Programs. Programming and Computer Software, Vol. 28, №3, 2002, p.5-14.

[3] Stoy J.E. Denotational semantics: the Scott-Strachey approach to programming language theory. MIT Press, 1977.

[4] Field A.J., Harrison P.G. Functional programming. Addison-Wesley Pub. Co., 1988.

[5] Нигиян С.А., Аветисян С.А. Об алгоритмах интерпретации бестиповых функциональных программ. Вестник РАУ. Серия физико-математические и естественные науки, 2, 2006, с.77-84.

Երկու ֆունկցիոնալ ծրագրեր համարվում են համարժեք, եթե նրանց հիմնական սեմանտիկաները համընկնում են: Հաճախ օպտիմիզացման նպատակների համար անհրաժեշտ է լինում ծրագիրը ձևափոխել, այսինքն՝ ներկայացնել այն մեկ այլ համարժեք ծրագրով: Ակնհայտ է, որ, օգտվելով ծրագրերի հիմնական սեմանտիկայից, կամայական ծրագիր կարելի է ձևափոխել մեկ հավասարումից կազմված ծրագրի: Բնականաբար հարց է առաջանում, թե արդյոք կարելի է ցանկացած ծրագիր ձևափոխել մեկ հավասարումից կազմված ծրագրի այնպես, որ պահպանվեն բոլոր պրոցեդուրային սեմանտիկաները:

Հիմնական օգտագործվող ձևախոխություններից է տեղադրման ձևախոխությունը, որն իրենից ներկայացնում է հավասարումների աջ մասերում փոփոխականների փոխարեն համապատասխան թերմերի տեղադրություն: Հարց է առաջանում. արդյոք պահպանվում են, թե ոչ ծրագրերի հիմնական ոչ լրիվ պրոցեդուրային սեմանտիկաները այդպիսի ձևախոխություններից հետո: Հետաքրքրության է արժանի նաև ֆունկցիոնալ ծրագրերի հավասարումների քանակի մինիմիզացման խնդիրը՝ հիմնված տեղադրման ձևախոխության վրա:

**Աշխատանքի նպատակն ու խնդիրները:** Այս աշխատանքի հիմնական նպատակն ու խնդիրները հետևյալներն են.

1. Հետազոտել հիմնական ոչ լրիվ պրոցեդուրային սեմանտիկաների համեմատելիության խնդիրը մեկ հավասարումից կազմված ծրագրերի համար:
2. Պարզել, թե արդյոք կարելի է կամայական ծրագիր ձևափոխել մեկ հավասարումից կազմված ծրագրի այնպես, որ պահպանվեն նրա բոլոր հիմնական պրոցեդուրային սեմանտիկաները:
3. Հետազոտել այն ձևափոխությունները, որոնք հիմնված են տեղադրումների վրա, պարզել՝ պահպանում են այդ ձևափոխությունները բոլոր հիմնական պրոցեդուրային սեմանտիկաները, թե ոչ, և հետազոտել այն ձևափոխությունների բարդությունը, որոնք կամայական ծրագիր ձևափոխում են մինիմալ քանակությամբ հավասարումներից կազմված ծրագրի:

**Հետազոտության մեթոդները:** Այս աշխատանքում հետազոտության համար օգտագործվում են առանց տիպերի λ-հաշվի, հանրահաշվի, ալգորիթմների տեսության, մաթեմատիկական տրամաբանության և գրաֆների տեսության մեթոդները:

**Արդյունքների նորությունը:** Ատենախոսության հիմնական արդյունքներն են.

1. Ապացուցվել է, որ պրոցեդուրային սեմանտիկաները, որոնցից մեկը հիմնված է $ACT$ ալգորիթմի վրա, իսկ մյուսը՝ $SNFR$ դասի ալգորիթմներից որևէ մեկի վրա, համեմատելի են մեկ հավասարումից կազմված ծրագրերի դասում: Ապացուցվել է, որ պրոցեդուրային սեմանտիկաները, հիմնված $LSNFR$, $RSNFR$, $FSNFR$ ալգորիթմների վրա, զույգ առ զույգ անհամեմատելի են մեկ հավասարումից կազմված ծրագրերի դասում: Ապացուցվել է նաև, որ պրոցեդուրային սեմանտիկաները, որոնցից մեկը օգտագործում է ինտերպրետացիայի $ACT^*$ ալգորիթմը, իսկ մյուսը՝ ցանկացած ալգորիթմ $SNFR$ դասից, անհամեմատելի են մեկ հավասարումից կազմված ծրագրերի դասում:
2. Ապացուցվել է, որ գոյություն ունի այնպիսի ծրագիր, որը չունի իրեն համարժեք մեկ հավասարումից կազմված ծրագիր այնպես, որ համընկնեն նրանց բոլոր հիմնական պրոցեդուրային սեմանտիկաները: Նկարագրվել է ձևափոխություն, որը

կամայական ծրագիր ձևափոխում է երկու հավասարումից կազմված ծրագրի՝ պահպանելով $SNFR$ դասի այն ալգորիթմների վրա հիմնված պրոցեդուրային սեմանտիկաները, որոնք բավարարում են որոշակի պայմանի:

3. Ապացուցվել է, որ այն ձևափոխությունները, որոնք հիմնված են տեղադրումների վրա, ընդհանուր առմամբ, չեն պահպանում ծրագրերի հիմնական պրոցեդուրային սեմանտիկաները և, որ այդ ձևափոխություններով կամայական ծրագրի ձևափոխումը մինիմալ քանակությամբ հավասարումներից կազմված ծրագրի, NP դժվար խնդիր է: Ապացուցվել է նաև, որ երկու տարբեր մինիմալ ծրագրերի հիմնական պրոցեդուրային սեմանտիկաները, ընդհանուր առմամբ, չեն համընկնում:

**Տեսական և կիրառական նշանակությունը:** Այս աշխատանքում ստացված արդյունքներն ունեն ինչպես տեսական, այնպես էլ կիրառական նշանակություն: Դրանք կարող են օգտագործվել ֆունկցիոնալ ծրագրերի օպտիմիզացման խնդիրներում, ինչպես նաև նոր ֆունկցիոնալ ծրագրավորման համակարգերի ստեղծման ժամանակ:

**Ստացված արդյունքների ապրոբացիան:** Ատենախոսության հիմնական արդյունքները զեկուցվել են ԵՊՀ ծրագրավորման և ինֆորմացիոն տեխնոլոգիաների ամբիոնի սեմինարում, ԵՊՀ ինֆորմատիկայի և կիրառական մաթեմատիկայի ֆակուլտետի ընդհանուր սեմինարում, Հայ-Ռուսական (Սլավոնական) համալսարանի տարեկան գիտաժողովում (Երևան, 2009) և միջազգային գիտաժողովներում (CSIT-07, Երևան, 2007 և CSIT-09, Երևան, 2009):

Աշխատանքի հիմնական արդյունքները ներառված են հրատարակված չորս աշխատանքներում:

**Աշխատանքի կառուցվածքն ու ծավալը:** Ատենախոսությունը բաղկացած է ներածությունից, երեք գլխից, եզրակացությունից և օգտագործված գրականության ցանկից (30 անուն): Ատենախոսության ծավալը 82 էջ է:

## ԱՇԽԱՏԱՆՔԻ ԲՈՎԱՆԴԱԿՈՒԹՅՈՒՆԸ

**Ներածությունում** նկարագրված է ատենախոսական աշխատանքի հետազոտության հիմնական խնդիրը, հիմնավորված է ատենախոսության թեմայի արդիականությունը և նորությունը: Համառոտ կերպով ներկայացված է նաև աշխատանքի բովանդակությունը:

**Գլուխ 1-ում** բերված են հիմնական սահմանումները, որոնք վերցված են [1,2]-ից: Հետազոտված է ոչ լրիվ պրոցեդուրային սեմանտիկաների համեմատելիության խնդիրը մեկ հավասարումից կազմված ծրագրերի դասում: Առաջին գլուխը բաղկացած է երկու բաժնից՝ 1.1-1.2:

**1.1 բաժնում** ներկայացված են հիմնական սահմանումները և ձևակերպված է թեորեմ 1.1.1-ը[2]:

---

[1] Barendregt H.P. The lambda calculus, its syntax and semantics. Amsterdam: North Holland Pub. Comp., 1981.

[2] Nigiyan S.A., Avetisyan S.A. Semantics of Untyped Functioanal Programs. Programming and Computer Software, Vol. 28, №3, 2002, p.5-14.

Փոփոխականների հաշվելի բազմությունը նշանակենք $V$-ով:

**Սահմանում 1.1.1** Սահմանենք թերմերի $\Lambda$ բազմությունը.

1) Եթե $x \in V$, ապա $x \in \Lambda$;
2) Եթե $t_1, t_2 \in \Lambda$, ապա $(t_1 t_2) \in \Lambda$: Այս դեպքում կասենք, որ $(t_1 t_2)$ թերմը ստացվել է $t_1, t_2$ թերմերից ապլիկացիայի գործողության միջոցով:
3) Եթե $x \in V$ և $t \in \Lambda$, ապա $(\lambda x t) \in \Lambda$: Այս դեպքում կասենք, որ $(\lambda x t)$ թերմը ստացվել է $t$ թերմից և $x$ փոփոխականից աբստրակցիայի գործողության միջոցով, $\lambda x$-ը կանվանենք աբստրակտոր, իսկ $t$-ն՝ աբստրակտորի ազդեցության տիրույթ:

**Սահմանում 1.1.2** Դիցուք $t \in \Lambda$: $x$ փոփոխականի ֆիքսած մուտքը $t$ թերմում կանվանենք կապված, եթե այն կամ պատկանում է որևէ աբստրակտորի, կամ գտնվում է այդ փոփոխականն օգտագործող աբստրակտորի ազդեցության տիրույթում: $x$ փոփոխականի մուտքը $t$ թերմում կանվանենք ազատ, եթե այն կապված չէ: Կասենք, որ $x$ փոփոխականը ազատ է $t$ թերմում, եթե այն ունի առնվազն մեկ ազատ մուտք:

$t$ թերմի բոլոր ազատ փոփոխականները կնշանակենք $fv(t)$–ով:

Այն թերմը, որը չունի ազատ փոփոխականներ, կանվանենք փակ:

Բոլոր փակ թերմերի բազմությունը կնշանակենք $\Lambda^o$–ով:

Պայմանավորվենք $t[x_1, \dots, x_m]$–ով նշանակել $t$ թերմը՝ նշելով նրանում մեզ հետաքրքրող իրարից տարբեր փոփոխականները՝ $x_1, x_2 \dots, x_m, m > 0$:

$t[t_1, \dots, t_m]$-ով կամ $t[x_1 := t_1, \dots, x_m := t_m]$-ով կնշանակենք այն թերմը, որը ստացվում է $t$-ից՝ $x_1, x_2 \dots, x_m$ փոփոխականների բոլոր ազատ մուտքերի փոխարեն միաժամանակ տեղադրելով համապատասխանաբար $t_1, t_2, \dots, t_m$ թերմերը:

Պայմանավորվենք $t\langle x \rangle$-ով նշանակել $t$ թերմը՝ նշելով $x$ փոփոխականի ֆիքսած ազատ մուտքը, և $t\langle \tau \rangle$-ով այն թերմը, որը ստացվում է $t$-ից՝ $x$ ֆիքսած մուտքի փոխարեն տեղադրելով $\tau$ թերմը:

Տեղադրումը կանվանենք թույլատրելի, եթե այդ գործողությունից հետո տեղադրվող թերմերի ոչ մի ազատ մուտք չի կապվում: Մենք կդիտարկենք միայն թույլատրելի տեղադրումներ:

**Սահմանում 1.1.3** $t_1$ և $t_2$ թերմերը կանվանենք կոնգրուենտ, եթե նրանցից մեկը կարելի է ստանալ մյուսից՝ վերանվանելով կապված փոփոխականները, և կգրենք այսպես՝ $t_1 \equiv t_2$:

Այսուհետև մենք չենք տարբերակի կոնգրուենտ թերմերը:

Ներմուծենք թերմերի կարճ գրելաձևեր.

$(\dots(t_1 t_2) \dots t_k)$ թերմը, որտեղ $t_i \in \Lambda$, $i = 1, \dots, k, k > 1$, կնշանակենք $t_1 t_2 \dots t_k$-ով, $(\lambda x_1 (\lambda x_2 (\dots (\lambda x_m t) \dots)$ թերմը, որտեղ $x_j \in V, t \in \Lambda$, կնշանակենք $\lambda x_1 x_2 \dots x_m . t$ -ով,

$j = 1, \dots, m, m > 0$,

$I \equiv \lambda x . x$,

$T \equiv \lambda xy . x$,

$F \equiv \lambda xy.y$,

$\Omega \equiv (\lambda x.xx)(\lambda x.xx)$,

$if\ t_1\ then\ t_2\ else\ t_3\ \equiv\ t_1t_2t_3$, որտեղ $t_1, t_2, t_3\ \in \Lambda$,

$< t_1, \dots, t_m > \equiv \lambda x.xt_1 \dots t_m$, որտեղ $x \in V, t_i \in \Lambda, x \notin fv(t_i), i = 1, \dots, m, m \geq 1$,

$U_i^m \equiv\ \lambda x_1 \dots x_m.x_i$, որտեղ $x_i \in V, k \neq j \Longrightarrow x_k \not\equiv x_j, k, j = 1, \dots, m, 1 \leq i \leq m,\ m \geq 1$,

$P_i^m \equiv\ \lambda x.xU_i^m$, որտեղ $x \in V, 1 \leq i \leq m, m \geq 1$,

$Y \equiv\ \lambda h.\big(\lambda x.h(xx)\big)\big(\lambda x.h(xx)\big)$, որտեղ $x, h \in V$: $Y$-ը կանվանենք անշարժ կետի կոմբինատոր:

**Սահմանում 1.1.4** β-ռեդուկցիայի գաղափարը իրենից ներկայացնում է հետևյալ զույգերի բազմությունը՝ $\beta = \{((\lambda x.t[x])t',\ t[x \coloneqq t']) \mid t, t' \in \Lambda, x \in V\}$, որտեղ $(\lambda x.t[x])t'$ թերմը կոչվում է $\beta$-ռեդեքս կամ ուղղակի ռեդեքս, իսկ $t[x \coloneqq t']$ թերմը՝ նրա փաթեթ:

Միաքայլ β-ռեդուկցիան ($\rightarrow_\beta$), β-ռեդուկցիան ($\rightarrow\rightarrow_\beta$) և $\beta$ հավասարությունը ($=_\beta$) սահմանվում են սովորական ձևով: Այսուհետև պայմանավորվենք $\beta$ սիմվոլը չգրել:

Այն թերմը, որը չի պարունակում ռեդեքս, կոչվում է նորմալ ձև: Բոլոր նորմալ ձևերի բազմությունը նշանակենք $NF$-ով, իսկ փակ նորմալ ձևերի բազմությունը՝ $NF^o$-ով:

Կասենք, որ $t$ թերմն ունի նորմալ ձև, եթե գոյություն ունի $t' \in NF$, այնպես որ $t = t'$:

$\lambda x_1 \dots x_k.xt_1 \dots t_m$ տեսքի թերմերը, որտեղ $x, x_i \in V,\ t_j \in \Lambda, i = 1, \dots, k, j = 1, \dots, m, k \geq 0, m \geq 0$, կոչվում են գլխային նորմալ ձևեր: Բոլոր գլխային նորմալ ձևերի բազմությունը նշանակենք $HNF$-ով: Կասենք, որ $t$ թերմն ունի գլխային նորմալ ձև, եթե գոյություն ունի $t' \in HNF$, այնպես որ $t = t'$: Հայտնի է, որ $NF \subset HNF$, և $HNF \not\subset NF$:

**Սահմանում 1.1.5** Առանց տիպերի ֆունկցիոնալ ծրագիրն իրենից ներկայացնում է (1) հավասարումների համակարգը՝

$$\begin{aligned} f_1 &= t_1[f_1 \dots f_m] \\ &\dots \\ f_m &= t_m[f_1 \dots f_m] \end{aligned} \qquad (1)$$

որտեղ $f_i \in V, i \neq j \Longrightarrow f_i \not\equiv f_j,\ t_i[f_1 \dots f_m] \in \Lambda, fv(t_i[f_1 \dots f_m]) \subset \{f_1, \dots, f_m\}$,

$i, j = 1, \dots, m, m \geq 1$: (1)-ում առաջին հավասարումը կհամարենք ծրագրի գլխավոր հավասարում:

Բոլոր (1) տիպի ծրագրերի բազմությունը նշանակենք $\mathcal{P}$-ով:

Դիտարկենք (1) ծրագրի լուծումը՝ $(\tau_1, \dots, \tau_m)$, որտեղ

$\tau_i \equiv P_i^m(Y(\lambda x.< t_1[P_1^m x, \dots, P_m^m x], \dots, t_m[P_1^m x, \dots, P_m^m x] >)), i = 1, \dots, m$:

$\tau_1$ թերմը կանվանենք լուծման գլխավոր կոմպոնենտ, որը անվանում են (1) ծրագրի անշարժ կետի սեմանտիկա (հիմնական սեմանտիկա):

**Սահմանում 1.1.6** Դիցուք $P$-ն (1) ծրագիրն է, իսկ $\tau_1$-ը՝ $P$ ծրագրի անշարժ կետի սեմանտիկան, այդ դեպքում $Fix(P)$ բազմությունը սահմանվում է հետևյալ կերպ՝

$Fix(P) = \{(v_1, \dots, v_k, t_0) | \tau_1 v_1 \dots v_k \rightarrow\rightarrow t_0,\ v_1, \dots, v_k, t_0 \in NF^0, k \geq 0\}$:

$Fix(P)$ կանվանենք $P$ ծրագրի անշարժ կետի սեմանտիկային համապատասխանող բազմություն:

**Թեորեմ 1.1.1 (տեղադրումների մասին):** Դիցուք տրված է (1) տիպի $P$ ծրագիր և $v_1, \dots, v_k, t_0 \in NF^0$ թերմերը, որտեղ $k \geq 0$: Այդ դեպքում՝

$(v_1, \dots, v_k, t_0) \in Fix(P) \Leftrightarrow \exists\, n \geq 1; t_1^n[f_1, \dots, f_m]v_1 \dots v_k \to\to t_0$, որտեղ $t_i^0[f_1, \dots, f_m] \equiv f_i$, $t_i^s[f_1, \dots, f_m] \equiv t_i\left[t_1^{s-1}[f_1, \dots, f_m], \dots, t_m^{s-1}[f_1, \dots, f_m]\right], s \geq 1, i = 1, \dots, m$:

**1.2 բաժնում** սահմանված են հիմնական պրոցեդուրային սեմանտիկաները և ներմուծված են անհակասելիության, լրիվության և անհամեմատելիության գաղափարները: Հետազոտված է հիմնական ոչ լրիվ պրոցեդուրային սեմանտիկաների համեմատելիության խնդիրը մեկ հավասարումից կազմված ծրագրերի համար:

Բերենք ինտերպրետացիայի ալգորիթմի սահմանումը:

**Սահմանում 1.2.1** $A$ ինտերպրետացիայի ալգորիթմը, ստանալով (1) տեսքի $P$ ծրագիր և $X$ թերմ՝ որպես մուտքային տվյալներ, կամ ավարտում է աշխատանքը և վերադարձնում $X' \in NF,\ fv(X') \cap \{f_1, \dots, f_m\} = \emptyset$, կամ աշխատում է անվերջ:

$A$ ալգորիթմը օգտագործում է հետևյալ գործողությունները՝

ա) $t_1[f_1 \dots f_m], \dots, t_m[f_1 \dots f_m]$ թերմերի տեղադրում համապատասխանաբար $f_1, \dots, f_m$ փոփոխականների որոշ ազատ մուտքերի փոխարեն:

բ) միաքայլ β-ռեդուկցիա:

Այժմ սահմանենք $Proc_A(P)$ բազմությունը, որը համապատասխանում է $A$ ինտերպրետացիայի ալգորիթմն օգտագործող պրոցեդուրային սեմանտիկային:

**Սահմանում 1.2.2** Դիցուք $P$-ն (1) ծրագիրն է, և $A$-ն՝ որևիցե ինտերպրետացիայի ալգորիթմ: Այդ դեպքում՝ $Proc_A(P)$ բազմությունը սահմանվում է հետևյալ կերպ.

$Proc_A(P) = \{(v_1, \dots, v_k, t_0) \mid A(P, t_1[f_1, \dots, f_m]v_1 \dots v_k)$ որոշված է և հավասար $t_0,\ v_1, \dots, v_k, t_0 \in NF^0, k \geq 0\}$:

**Սահմանում 1.2.3** $A$ ինտերպրետացիայի ալգորիթմին համապատասխանող պրոցեդուրային սեմանտիկան անհակասելի է, եթե կամայական $P \in \mathcal{P}$ ծրագրի համար $Proc_A(P) \subset Fix(P)$:

Աշխատություն [1]-ում ապացուցված է թեորեմ 1.2.1-ը:

**Թեորեմ 1.2.1 (անհակասելիության մասին):** Կամայական $P \in \mathcal{P}$ ծրագրի և ինտերպրետացիայի $A$ ալգորիթմի համար $Proc_A(P) \subset Fix(P)$:

**Սահմանում 1.2.4** $A$ ինտերպրետացիայի ալգորիթմին համապատասխանող պրոցեդուրային սեմանտիկան լրիվ է, եթե կամայական $P \in \mathcal{P}$ ծրագրի համար

$Fix(P) \subset Proc_A(P)$:

[1] Nigiyan S.A., Avetisyan S.A. Semantics of Untyped Functioanal Programs. Programming and Computer Software, Vol. 28, №3, 2002, p.5-14.

**Սահմանում 1.2.5** $A$ և $B$ ինտերպրետացիայի ալգորիթմներին համապատասխանող պրոցեդուրային սեմանտիկաները անհամեմատելի են, եթե բավարարվում են հետևյալ պայմանները՝

1) գոյություն ունի $P \in \mathcal{P}$ ծրագիր, այնպես որ $Proc_A(P) \not\subset Proc_B(P)$

2) գոյություն ունի $P \in \mathcal{P}$ ծրագիր, այնպես որ $Proc_B(P) \not\subset Proc_A(P)$

Եթե $X \in NF$, ապա $L(X)$-ով նշանակենք $X$ թերմը, իսկ հակառակ դեպքում $L(X)$-ով նշանակենք $X'$, որտեղ $X'$ ստացվում է $X$-ից միաքայլ ձախ ռեդուկցիայով (այն միաքայլ ռեդուկցիայով, որը համապատասխանում է ամենաձախ ռեդեքսին):

Նկարագրենք $N$ ալգորիթմը, որը տրված $X$ թերմի համար կառուցում է նրա նորմալ ձևը եթե այն գոյություն ունի և աշխատում անվերջ հակառակ դեպքում.

<u>$N$ ալգորիթմ</u>

Մուտք՝ X թերմը

Ելք՝ $N(X)$ թերմը, եթե $N$-ը որոշված է $X$-ի վրա:

1. Եթե $X \in NF$, ապա վերադարձնել $X$, հակառակ դեպքում՝ $N(L(X))$:

<u>$H$ ալգորիթմ</u>

Մուտք՝ $X$ թերմը

Ելք՝ $H(X)$ թերմը, եթե $H$-ը որոշված է $X$-ի վրա:

1. Եթե $X \in HNF$, ապա վերադարձնել $X$, հակառակ դեպքում՝ $H(L(X))$

**<u>Ինտերպրետացիայի ալգորիթմների $SNFR$ դասը.</u>**

Կամայական $A \in SNFR$ ալգորիթմ ունի հետևյալ տեսքը՝

Մուտք՝ (1) տիպի $P$ ծրագիր և $X$ թերմ այնպես, որ $fv(X) \subset \{f_1, \dots, f_m\}$,

Ելք՝ $A(P, X)$ թերմը, եթե $A$-ն որոշված է $P$-ի և $X$-ի վրա:

1. Եթե $N(X)$ որոշված չէ, ապա $A$ ալգորիթմը ևս որոշված չէ, և $N$ ալգորիթմի անվերջ աշխատանքը համապատասխանում է $A$ ալգորիթմի անվերջ աշխատանքին, հակառակ դեպքում անցնել 2-րդ կետին:
2. Եթե $N(X) \equiv t \in NF \setminus NF^0$, ապա $A(P, t')$, որտեղ $t'$ ստացվում է $t$-ից՝ $t_1[f_1 \dots f_m], \dots, t_m[f_1 \dots f_m]$ թերմերի տեղադրումով համապատասխանաբար $f_1, \dots, f_m$ փոփոխականների որոշ ազատ մուտքերի փոխարեն, հակառակ դեպքում վերադարձնել $N(X)$:

Ձևակերպենք $SNFR$ դասի կոնկրետ ալգորիթմներ՝ $LSNFR$, $RSNFR$, $FSNFR$:

<u>Ալգորիթմ $LSNFR$</u>

Մուտք՝ (1) տիպի $P$ ծրագիր և $X$ թերմ այնպես, որ $fv(X) \subset \{f_1, \dots, f_m\}$,
Ելք՝ $LSNFR(P, X)$ թերմը, եթե $LSNFR$-ը որոշված է $P$-ի և $X$-ի վրա:

1. Եթե $N(X)$ որոշված չէ, ապա $LSNFR$ ալգորիթմը ևս որոշված չէ, և $N$ ալգորիթմի անվերջ աշխատանքը համապատասխանում է $LSNFR$ ալգորիթմի անվերջ աշխատանքին, հակառակ դեպքում անցնել 2-րդ կետին:

2. Եթե $N(X) \equiv t\langle f_i\rangle \in NF \setminus NF^0$, որտեղ $f_i$-ն $\{f_1, \dots, f_m\}$ փոփոխականների ամենաձախ ազատ մուտքն է, ապա $LSNFR\ (P,\ t\langle t_i\rangle)$, հակառակ դեպքում վերադարձնել $N(X)$:

Ալգորիթմ *RSNFR*

Մուտք՝ (1) տիպի $P$ ծրագիր և $X$ թերմ այնպես, որ $fv(X) \subset \{f_1, \dots, f_m\}$,

Ելք՝ $RSNFR(P, X)$ թերմը, եթե $RSNFR$-ը որոշված է $P$-ի և $X$-ի վրա:

1. Եթե $N(X)$ որոշված չէ, ապա $RSNFR$ ալգորիթմը ևս որոշված չէ, և $N$ ալգորիթմի անվերջ աշխատանքը համապատասխանում է $RSNFR$ ալգորիթմի անվերջ աշխատանքին, հակառակ դեպքում անցնել 2-րդ կետին:
2. Եթե $N(X) \equiv t\langle f_i\rangle \in NF \setminus NF^0$, որտեղ $f_i$-ն $\{f_1, \dots, f_m\}$ փոփոխականների ամենաաջ ազատ մուտքն է, ապա $RSNFR\ (P,\ t\langle t_i\rangle)$, հակառակ դեպքում վերադարձնել $N(X)$:

Ալգորիթմ *FSNFR*

Մուտք՝ (1) տիպի $P$ ծրագիր և $X$ թերմ այնպես, որ $fv(X) \subset \{f_1, \dots, f_m\}$,

Ելք՝ $FSNFR(P, X)$ թերմը, եթե $FSNFR$-ը որոշված է $P$-ի և $X$-ի վրա:

1. Եթե $N(X)$ որոշված չէ, ապա $FSNFR$ ալգորիթմը ևս որոշված չէ, և $N$ ալգորիթմի անվերջ աշխատանքը համապատասխանում է $FSNFR$ ալգորիթմի անվերջ աշխատանքին, հակառակ դեպքում անցնել 2-րդ կետին:
2. Եթե $N(X) \equiv t[f_1, \dots, f_m] \in NF \setminus NF^0$, ապա $FSNFR\ (P,\ t[t_1, \dots, t_m])$, հակառակ դեպքում վերադարձնել $N(X)$:

Ձևակերպենք $HNFR$, $PAS$, $ACT$ ալգորիթմները:

Ալգորիթմ *HNFR*

Մուտք՝ (1) տիպի $P$ ծրագիր և $X$ թերմ,

Ելք՝ $HNFR(P, X)$ թերմը, եթե $HNFR$-ը որոշված է $P$-ի և $X$-ի վրա:

1. Եթե $H(X)$ որոշված չէ, ապա $HNFR$ ալգորիթմը ևս որոշված չէ, և $H$ ալգորիթմի անվերջ աշխատանքը համապատասխանում է $HNFR$ ալգորիթմի անվերջ աշխատանքին, հակառակ դեպքում անցնել 2-րդ կետին:
2. Եթե $H(X) \equiv \lambda x_1 \dots x_u.\, x\theta_1[f_1, \dots, f_m] \dots \theta_v[f_1, \dots, f_m] \in NF$ և

   $fv(H(X)) \cap \{f_1, \dots, f_m\} = \emptyset$, ապա $H(X)$, հակառակ դեպքում, եթե $x \equiv f_i$, ապա $HNFR(P, \lambda x_1 \dots x_u.\, t_i[f_1, \dots, f_m]\theta_1[f_1, \dots, f_m] \dots \theta_v[f_1, \dots, f_m])$, հակառակ դեպքում՝ $\lambda x_1 \dots x_u.\, xHNFR(P, \theta_1[f_1, \dots, f_m]) \dots \dots HNFR(P, \theta_v[f_1, \dots, f_m])$, որտեղ $x, x_j \in V,\ \theta_s \in \Lambda,\ j = 1, \dots, u, s = 1, \dots, v$ և $v \geq 0, 1 \leq i \leq m$:

Դիցուք տրված է $t_{\tau_1}$ թերմը, որտեղ ֆիքսված է $\tau_1$ ենթաթերմի որևէ մուտք: $t_{\tau_2}$-ով պայմանավորվենք նշանակել այն թերմը, որը ստացվում է $t_{\tau_1}$-ից՝ $\tau_1$-ի ֆիքսված մուտքը փոխարինելով $\tau_2$ թերմով:

Ալգորիթմ *PAS*

Մուտք՝ (1) տիպի $P$ ծրագիր և $X$ թերմ այնպես, որ $fv(X) \subset \{f_1, \dots, f_m\}$,

Ելք՝ $PAS(P,X)$ թերմը, եթե $PAS$-ը որոշված է $P$-ի և $X$-ի վրա:

1. Եթե $X \in NF$ և $fv(X) \cap \{f_1, \dots, f_m\} = \emptyset$, ապա վերադարձնել $X$, հակառակ դեպքում անցնել 2-րդ կետին:
2. Եթե $X \equiv X\langle f_i \rangle$, որտեղ $f_i$-ն $\{f_1, \dots, f_m\}$ փոփոխականների ամենաձախ ազատ մուտքն է և այդ մուտքը գտնվում է $X$ թերմի ամենաձախ ռեդեքսից ձախ, ապա վերադարձնել $PAS(P, X\langle t_i \rangle)$, հակառակ դեպքում անցնել 3-րդ կետին:
3. Եթե $X \equiv X_{(\lambda x.t)\tau}$, որտեղ $x \in V, t, \tau \in \Lambda$, և $X$ թերմի ամենաձախ ռեդեքսը $(\lambda x.t)\tau$ թերմն է, ապա վերադարձնել $PAS(P, X_{t[x:=\tau]})$:

Ալգորիթմ $ACT$

Մուտք՝ (1) տիպի $P$ ծրագիր և $X$ թերմ,

Ելք՝ $ACT(P,X)$ թերմը, եթե $ACT$-ը որոշված է $P$-ի և $X$-ի վրա:

1. Եթե $X \in NF$ և $fv(X) \cap \{f_1, \dots, f_m\} = \emptyset$, ապա վերադարձնել $X$, հակառակ դեպքում անցնել 2-րդ կետին:
2. Եթե $X \equiv X\langle f_i \rangle$, որտեղ $f_i$-ն $\{f_1, \dots, f_m\}$ փոփոխականների ամենաձախ ազատ մուտքն է և այդ մուտքը գտնվում է $X$ թերմի ամենաձախ ռեդեքսից ձախ, ապա վերադարձնել $ACT(P, X\langle t_i \rangle)$, հակառակ դեպքում անցնել 3-րդ կետին:
3. Եթե $X \equiv X_{(\lambda x.t)\tau}$, որտեղ $x \in V, t, \tau \in \Lambda$, և $X$ թերմի ամենաձախ ռեդեքսը $(\lambda x.t)\tau$ թերմն է, ապա վերադարձնել $ACT(P, X_{t[x:=ACT(P,\tau)]})$:

**Լեմմա 1.2.1** Դիցուք տրված է $P$ ծրագիրը՝ $f = t$, որտեղ $t \in \Lambda, fv(t) = \{f\}$:

Այդ դեպքում՝ $Proc_{ACT}(P) = \emptyset$:

**Դիտողություն 1.2.1** Հաշվի առնելով այն, որ $ACT$ ալգորիթմը, ըստ լեմմա 1.2.1–ի, աշխատում է անվերջ մեկ հավասարումից կազմված ռեկուրսիվ ծրագրերի համար, մենք կնկարագրենք $ACT$ ալգորիթմի ձևափոխված տարբերակը՝ $ACT^*$:

$ACT$ ալգորիթմը $ACT^*$-ից տարբեվում է միայն 3-րդ կետում:

Ալգորիթմ $ACT^*$

Մուտք՝ (1) տիպի $P$ ծրագիր և $X$ թերմ,

Ելք՝ $ACT^*(P,X)$ թերմը, եթե $ACT^*$-ը որոշված է $P$-ի և $X$-ի վրա:

1. Եթե $X \in NF$ և $fv(X) \cap \{f_1, \dots, f_m\} = \emptyset$, ապա վերադարձնել $X$, հակառակ դեպքում անցնել 2-րդ կետին:
2. Եթե $X \equiv X\langle f_i \rangle$, որտեղ $f_i$-ն $\{f_1, \dots, f_m\}$ փոփոխականների ամենաձախ ազատ մուտքն է և այդ մուտքը գտնվում է $X$ թերմի ամենաձախ ռեդեքսից ձախ, ապա վերադարձնել $ACT^*(P, X\langle t_i \rangle)$, հակառակ դեպքում անցնել 3-րդ կետին:
3. Եթե $X \equiv X_{(\lambda x.t)\tau}$, որտեղ $x \in V, t, \tau \in \Lambda$, և $X$ թերմի ամենաձախ ռեդեքսը $(\lambda x.t)\tau$ թերմն է, ապա վերադարձնել $ACT^*(P, X_t)$ , եթե $x \notin fv(t)$, և $ACT^*(P, X_{t[x:=ACT^*(P,\tau)]})$ հակառակ դեպքում:

Չնակերպենք թեորեմներ 1.2.2 – 1.2.5-ը, որոնք վերցված են [1,2]-ից:

**Թեորեմ 1.2.2** ($SNFR$ դասի ալգորիթմներն օգտագործող պրոցեդուրային սեմանտիկաների ոչ լրիվության մասին): Գոյություն ունի $P \in \mathcal{P}$ ծրագիր այնպես, որ կամայական $A \in SNFR$ ալգորիթմի համար $Fix(P) \not\subset Proc_A(P)$:

**Թեորեմ 1.2.3** ($HNFR$ ալգորիթմն օգտագործող պրոցեդուրային սեմանտիկայի լրիվության մասին): Կամայական $P \in \mathcal{P}$ ծրագրի համար $Fix(P) \subset Proc_{HNFR}(P)$:

**Թեորեմ 1.2.4** ($PAS$ ալգորիթմն օգտագործող պրոցեդուրային սեմանտիկայի լրիվության մասին): Կամայական $P \in \mathcal{P}$ ծրագրի համար $Fix(P) \subset Proc_{PAS}(P)$:

**Թեորեմ 1.2.5** ($FSNFR, LSNFR, RSNFR, ACT$ ալգորիթմներն օգտագործող պրոցեդուրային սեմանտիկաների անհամեմատելիության մասին):

ա) պրոցեդուրային սեմանտիկաները, որոնք օգտագործում են ինտերպրետացիայի $FSNFR, LSNFR, RSNFR$ ալգորիթմները, զույգ առ զույգ անհամեմատելի են;

բ) պրոցեդուրային սեմանտիկաները, որոնցից մեկը օգտագործում է ինտերպրետացիայի $ACT$ ալգորիթմը, իսկ մյուսը՝ ցանկացած ալգորիթմ $SNFR$ դասից, անհամեմատելի են:

Ատենախոսության մեջ ապացուցվում են թեորեմներ 1.2.6 – 1.2.8-ը:

$\mathcal{P}_1$-ով նշանակենք այն $P \in \mathcal{P}$ ծրագրերը, որոնք բաղկացած են մեկ հավասարումից:

**Թեորեմ 1.2.6** ($ACT$ և $ACT^*$ ալգորիթմներն օգտագործող պրոցեդուրային սեմանտիկաների համեմատելիության մասին):

ա) կամայական $P \in \mathcal{P}$ ծրագրի համար $Proc_{ACT}(P) \subset Proc_{ACT^*}(P)$

բ) գոյություն ունի $P \in \mathcal{P}_1$ ծրագիր այնպես, որ $Proc_{ACT^*}(P) \not\subset Proc_{ACT}(P)$

**Թեորեմ 1.2.7** ($ACT$ և $SNFR$ ալգորիթմներն օգտագործող պրոցեդուրային սեմանտիկաների համեմատելիության մասին մեկ հավասարումից կազմված ծրագրերի դասում):

ա) կամայական $P \in \mathcal{P}_1$ ծրագրի և $A \in SNFR$ ալգորիթմի համար

$Proc_{ACT}(P) \subset Proc_A(P)$;

բ) գոյություն ունի $P \in \mathcal{P}_1$ ծրագիր այնպես, որ կամայական $A \in SNFR$ ալգորիթմի համար $Proc_A(P) \not\subset Proc_{ACT}(P)$:

**Թեորեմ 1.2.8** ($FSNFR, LSNFR, RSNFR, ACT^*$ ալգորիթմներն օգտագործող պրոցեդուրային սեմանտիկաների անհամեմատելիության մասին մեկ հավասարումից կազմված ծրագրերի դասում):

[1] Nigiyan S.A., Avetisyan S.A. Semantics of Untyped Functioanal Programs. Programming and Computer Software, Vol. 28, №3, 2002, p.5-14.

[2] Нигиян С.А., Аветисян С.А. Об алгоритмах интерпретации бестиповых функциональных программ. Вестник РАУ. Серия физико-математические и естественные науки, 2, 2006, с.77-84.

ա) պրոցեդուրային սեմանտիկաները, որոնք օգտագործում են ինտերպրետացիայի $FSNFR$, $LSNFR$, $RSNFR$ ալգորիթմները, զույգ առ զույգ անհամեմատելի են մեկ հավասարումից կազմված ծրագրերի դասում;

բ) պրոցեդուրային սեմանտիկաները, որոնցից մեկը օգտագործում է ինտերպրետացիայի $ACT^*$ ալգորիթմը, իսկ մյուսը՝ ցանկացած ալգորիթմ $SNFR$ դասից, անհամեմատելի են մեկ հավասարումից կազմված ծրագրերի դասում:

**Սահմանում 1.2.6** Կասենք, որ ինտերպրետացիայի $A$ ալգորիթմը բավարարում է (*) հատկությանը, եթե տեղի ունի հետևյալ պայմանը.

կամայական (1) տիպի $P$ ծրագրի համար, եթե $t\langle f\rangle \in NF\backslash NF^0$, $f \in \{f_1, \dots, f_m\}$ և $t' \equiv t\langle\tau\rangle, \tau \in NF$, և $\tau$ պարունակում է $f_1, \dots, f_m$ փոփոխականների միայն մեկ ազատ մուտք, ապա $A$ ալգորիթմը (ա) գործողությունն իրականացնելու ժամանակ ընտրում է փոփոխականների միևնույն ազատ մուտքերը $t$ և $t'$ թերմերի համար:

**Պնդում 1.2.1** $PAS$, $ACT$, $ACT^*$, $HNFR$, $LSNFR$, $RSNFR$, $FSNFR$ ալգորիթմները բավարարում են (*) հատկությանը:

Պայմանավորվենք $SNFR^*$-ով նշանակել այն $A \in SNFR$ ալգորիթմների բազմությունը, որոնք բավարարում են սահմանում 1.2.6-ում ներմուծված (*) հատկությանը:

**Գլուխ 2-ը** նվիրված է այն հարցին, թե արդյոք կարելի է կամայական ծրագիր ձևափոխել մեկ հավասարումից կազմված ծրագրի այնպես, որ պահպանվեն նրա հիմնական ոչ լրիվ պրոցեդուրային սեմանտիկաները: Երկրորդ գլուխը բաղկացած է երկու բաժնից՝ 2.1-2.2:

**2.1 բաժնում** ներմուծված է ծրագրերի համարժեքության գաղափարը և ապացուցված է թեորեմ 2.1.1-ը:

**Սահմանում 2.1.1** Կասենք որ $P$ ծրագիրը համարժեք է $P'$ ծրագրին, եթե $Fix(P) = Fix(P')$, և կգրենք $P{\sim}P'$:

Բոլոր հիմնական ինտերպրետացիայի ալգորիթմների բազմությունը նշանակենք $\mathcal{A}^{(0)}$-ով՝ $\mathcal{A}^{(0)} = \{LSNFR, RSNFR, FSNFR, ACT, ACT^*, PAS, HNFR\}$:

**Թեորեմ 2.1.1** Գոյություն ունի $P_0 \in \mathcal{P}$ ծրագիր, որի համար գոյություն չունի այնպիսի $P \in \mathcal{P}_1$ ծրագիր, որ $P{\sim}P_0$, և բավարարվի հետևյալ պայմանը՝ $Proc_A(P) = Proc_A(P_0)$, կամայական $A \in \mathcal{A}^{(0)}$ ալգորիթմի համար:

**2.2 բաժնում** ներմուծված է ծրագրերի ձևափոխության գաղափարը և նկարագրված է ձևափոխություն, որը կամայական ծրագիր ձևափոխում է երկու հավասարումից կազմված ծրագրի՝ պահպանելով նրանց $SNFR^*$ ինտերպրետացիայի ալգորիթմներին համապատասխանող պրոցեդուրային սեմանտիկաները:

**Սահմանում 2.2.1** $T: \mathcal{P} \to \mathcal{P}$ հաշվարկելի արտապատկերումը կոչվում է $\mathcal{P}$ ծրագրերի ձևափոխություն, եթե ցանկացած $P \in \mathcal{P}$ ծրագրի համար $T(P){\sim}P$:

**Դիտողություն 2.2.1** Ակնհայտ է, որ գոյություն ունի այնպիսի $T_1$ ձևափոխություն, որը կամայական (1) տիպի $P$ ծրագիր կձևափոխի մեկ հավասարումից կազմված ծրագրի: Իրոք, դիտարկենք հետևյալ $T_1 : \mathcal{P} \to \mathcal{P}_1$ արտապատկերումը.

(1) տիպի $P \in \mathcal{P}$ ծրագրին համապատասխանեցնենք հետևյալ $P' \in \mathcal{P}_1$ ծրագիրը՝ $f = \tau_1$, որտեղ $f \in V$, իսկ $\tau_1$-ն $P$ ծրագրի հիմնական սեմանտիկան է: Թեորեմ 1.1.1-ից անմիջապես հետևում է, որ $Fix(P) = Fix(P')$: Հետևաբար $T_1$-ը ձևափոխություն է:

Նշանակենք $t^{[n]}$-ով $t\underbrace{F \dots F}_{n}T$ թերմը, որտեղ $t \in \Lambda, n \geq 0$:

Դիցուք $P$-ն (1) ծրագիրն է, և $t \in \Lambda, fv(t) \subset \{f_1, \dots, f_m\}$: $\bar{t}$-ով կնշանակենք այն թերմը, որը ստացվում է $t$-ից՝ $f_j, j = 1, \dots, m$ փոփոխականների ամեն մի ազատ մուտք փոխարինելով $f_2^{[j-1]}$ թերմով, $j = 1, \dots, m$:

<u>$C$ ալգորիթմ</u>

Մուտք՝ $t_1, \dots, t_m, m > 1$ թերմերը:

Ելք՝ $C(t_1 \dots, t_m)$ թերմը:

1. Եթե $m = 1$, ապա վերադարձնել $\lambda x.\, x\, t_1 I$ թերմը, հակառակ դեպքում վերադարձնել $\lambda x.\, x\, t_1 C(t_2, \dots, t_m)$ թերմը:

<u>$Converter$ ալգորիթմ</u>

Մուտք՝ (1) տիպի $P$ ծրագիրը:

Ելք՝ $Converter(P)$ ծրագիրը:

1. Վերադարձնել (2) ծրագիրը՝

$$f_1 = t_1'[f_2] \equiv f_2 T \tag{2}$$
$$f_2 = t_2'[f_2] \equiv \overline{C(t_1, \dots, t_m)}$$

**Թեորեմ 2.2.1** Կամայական (1) տիպի $P$ ծրագրի համար $Converter(P) \sim P$:

**Թեորեմ 2.2.2** Կամայական (1) տիպի $P$ ծրագրի համար $Proc_A\big(Converter(P)\big) = Proc_A(P)$ ցանկացած $A \in SNFR^*$ ինտերպրետացիայի ալգորիթմի համար:

**Գլուխ 3-ը** նվիրված է ծրագրերի այն ձևափոխություններին, որոնք հիմնված են տեղադրման գործողության վրա: Հետազոտված են ծրագրերի հիմնական պրոցեդուրային սեմանտիկաները այդպիսի ձևափոխությունների ժամանակ: Ուսումնասիրված է ծրագրերի հավասարումների քանակի մինիմիզացման խնդիրը հիմնված տեղադրման ձևափոխությունների վրա: Գլուխ 3-ը բաղկացած է չորս բաժնից՝ 3.1-3.4:

**3.1 բաժնում** սահմանված է տեղադրման գործողությունը ֆունկցիոնալ ծրագրերի համար, ապացուցված է, որ այն ձևափոխություն է: Հետազոտված են հիմնական ոչ լրիվ պրոցեդուրային սեմանտիկաները այդ ձևափոխությունների ժամանակ:

**Սահմանում 3.1.1** $f_i$ փոփոխականը կանվանենք ռեկուրսիվ (1) ծրագրում, եթե $f_i \in fv(t_i),\ i = 1, \dots, m$:

**Սահմանում 3.1.2 (տեղադրման գործողություն):** Դիտարկենք (3), (4) ծրագրերը՝

$$f_1 = t_1$$
$$\dots$$
$$f_{i-1} = t_{i-1}$$

$$
\begin{aligned}
&f_i = t_i \\
&f_{i+1} = t_{i+1} \\
&\quad \dots \\
&f_m = t_m
\end{aligned}
\tag{3}
$$

$$
\begin{aligned}
&f_1 = t_1[f_i \coloneqq t_i] \\
&\quad \dots \\
&f_{i-1} = t_{i-1}[f_i \coloneqq t_i] \\
&f_{i+1} = t_{i+1}[f_i \coloneqq t_i] \\
&\quad \dots \\
&f_m = t_m[f_i \coloneqq t_i]
\end{aligned}
\tag{4}
$$

որտեղ $f_i \in V, i \neq j \Longrightarrow f_i \not\equiv f_j, t_i[f_1 \dots f_m] \in \Lambda, fv(t_i[f_1 \dots f_m]) \subset \{f_1, \dots, f_m\}$,

$i, j = 1, \dots, m, m \geq 1$:

Կասենք, որ (4) ծրագիրը ստացվել է (3) ծրագրից կիրառելով տեղադրման գործողություն ոչ ռեկուրսիվ $f_i$ փոփոխականի համար, $i = 1, \dots, m$:

Դիցուք տրված է (1) տիպի $P$ ծրագիր: Կասենք, որ $P'$ ծրագիրը ստացվում է $P$ ծրագրից տեղադրման գործողության միջոցով, եթե $P'$-ը ստացվում է $P$-ից կիրառելով տեղադրման գործողություն որևէ $f_i,\ i = 1, \dots, m$ փոփոխականի համար:

Այսուհետև կպայմանավորվենք չդիտարկել $f_1$-ի տեղադրումները:

**Թեորեմ 3.1.1** Դիցուք ունենք (1) տիպի $P$ ծրագիր, և $P'$ ծրագիրը ստացվել է $P$-ից՝ տեղադրման գործողության միջոցով: Այդ դեպքում $P \sim P'$:

**Թեորեմ 3.1.2** Դիցուք ունենք (1) տիպի $P$ ծրագիր, և $P'$ ստացվել է $P$-ից՝ տեղադրման գործողության միջոցով: Այդ դեպքում $Proc_{ACT}(P) = Proc_{ACT}(P'),\ Proc_{ACT^*}(P) = Proc_{ACT^*}(P')$:

**Սահմանում 3.1.3** Կասենք որ, $T$ ձևափոխությունը հիմնված է տեղադրման գործողության վրա, եթե կամայական (1) տիպի $P$ ծրագրի համար գոյություն ունեն $P_1, P_2, \dots, P_k$ ծրագրեր այնպես, որ $P_1 = P,\ T(P) = P_k,$ և $P_i$ ստացվում է $P_{i-1}$-ից՝ տեղադրման գործողության միջոցով, $i = 2, \dots, k,\ k \geq 2$:

Տեղադրման գործողության վրա հիմնված բոլոր ձևափոխությունների բազմությունը կնշանակենք $\mathcal{T}$-ով:

$LSNFR, RSNFR, FSNFR$ ինտերպրետացիայի ալգորիթմների բազմությունը նշանակենք $\mathcal{A}^{(1)}$-ով՝ $\mathcal{A}^{(1)} = \{LSNFR, RSNFR, FSNFR\}$:

**Թեորեմ 3.1.3** Կամայական $A \in \mathcal{A}^{(1)}$ ինտերպրետացիայի ալգորիթմի համար գոյություն ունի $P_A$ ծրագիր և $T_A \in \mathcal{T}$ ձևափոխություն այնպես, որ $Proc_A(P_A) \neq Proc_A(T_A(P_A))$:

**3.2 բաժնում** ձևակերպվում է հավասարումների քանակի մինիմիզացման խնդիրը ֆունկցիոնալ ծրագրերի համար՝ *MES* և ուսումնասիրվում են այդ խնդրի իմաստով մինիմալ քանակությամբ հավասարումներից կազմված ծրագրերի պրոցեդուրային սեմանտիկաները:

Այժմ սահմանենք *MES* (Minimal Equation System) խնդիրը ֆունկցիոնալ ծրագրերի համար:

**Խնդիր *MES*** Դիցուք տրված է (1) տիպի $P$ ծրագիր: Գտնել մինիմալ քանակությամբ հավասարումներ պարունակող $P'$ ծրագիր այնպես, որ $P' = T(P)$, որտեղ $T \in \mathcal{T}$:

**Թեորեմ 3.2.1** Գոյություն ունի $P$ ծրագիր այնպես, որ $Proc_A(P_1) \neq Proc_A(P_2)$ կամայական $A \in SNFR$ ալգորիթմի համար, որտեղ $P_1, P_2$-ը $P$ ծրագրով որոշվող *MES* խնդրի տարբեր լուծումներն են:

**3.3 բաժնում** բերվում են գրաֆների տեսությունից օգտագործված սահմանումները, և նկարագրվում է *MFVS* NP դժվար խնդիրը:

Դիցուք ունենք $G = (V, A)$ կողմնորոշված գրաֆը, որտեղ

$V$-ն գագաթների բազմությունն է,

$A$ -ն աղեղների բազմությունն է:

**Սահմանում 3.3.1** $V_0 V_1 \dots V_{k-1}$ գագաթների հաջորդականությունը, որտեղ $V_0 = V_{k-1}, V_i \neq V_j, i \neq j\ i, j = 1, \dots, k-2,\ k \geq 2, (V_{p-1}, V_p) \in A, p = 1, \dots, k-1$, կոչվում է ցիկլ:

**Սահմանում 3.3.2** $G = (V, A)$ կողմնորոշված գրաֆը կանվանենք ացիկլիկ, եթե այն ցիկլ չի պարունակում:

**Սահմանում 3.3.3** $V' \subset V$ բազմությունը կանվանենք հատույթ $G = (V, A)$ կողմնորոշված գրաֆի համար, եթե $G'' = (V'', A'')$ գրաֆը ացիկլիկ է, որտեղ $V'' = V \setminus V'$, $A'' = \{(x, y) | (x, y) \in A \text{ և } x, y \in V''\}$:

**Խնդիր *MFVS*** Գտնել մինիմալ քանակությամբ գագաթներ պարունակող հատույթ տրված $G = (V, A)$ գրաֆի համար:

Հայտնի է, որ *MFVS*-ը NP դժվար խնդիր է[1]:

**Թեորեմ 3.3.1 (*MFVS* խնդրի NP դժվարության մասին):** *MFVS* խնդիրը հանդիսանում է NP դժվար խնդիր:

**3.4 բաժնում** ցույց է տրվում *MFVS* և *MES* խնդիրների համարժեքությունը:

**Թեորեմ 3.4.1** *MFVS*-ը համարժեք է *MES* -ին:

Թեորեմ 3.4.1-ից հետևում է թեորեմ 3.4.2-ը:

**Թեորեմ 3.4.2 (*MES* խնդրի NP դժվարության մասին):** *MES* խնդիրը հանդիսանում է *NP* դժվար խնդիր:

[1] Karp R.M., Miller R.E., Thatcher J.W. Reducibility among combinatorial problems. Complexity of Computer Computations. Plenum Press, New York, 1972.

## ԱՇԽԱՏԱՆՔԻ ՀԻՄՆԱԿԱՆ ԱՐԴՅՈՒՆՔՆԵՐԸ

Ատենախոսությունում ստացվել են հետևյալ հիմնական արդյունքները.

1. Ապացուցվել է, որ պրոցեդուրային սեմանտիկաները, որոնցից մեկը հիմնված է *ACT* ալգորիթմի վրա, իսկ մյուսը՝ *SNFR* դասի ալգորիթմներից որևէ մեկի վրա, համեմատելի են մեկ հավասարումից կազմված ծրագրերի դասում: Ապացուցվել է, որ պրոցեդուրային սեմանտիկաները, հիմնված *LSNFR*, *RSNFR*, *FSNFR* ալգորիթմների վրա, զույգ առ զույգ անհամեմատելի են մեկ հավասարումից կազմված ծրագրերի դասում: Ապացուցվել է նաև, որ պրոցեդուրային սեմանտիկաները, որոնցից մեկը օգտագործում է ինտերպրետացիայի *ACT** ալգորիթմը, իսկ մյուսը՝ ցանկացած ալգորիթմ *SNFR* դասից, անհամեմատելի են մեկ հավասարումից կազմված ծրագրերի դասում, [2]:
2. Ապացուցվել է, որ գոյություն ունի այնպիսի ծրագիր, որը չունի իրեն համարժեք մեկ հավասարումից կազմված ծրագիր այնպես, որ համընկնեն նրանց բոլոր հիմնական պրոցեդուրային սեմանտիկաները: Նկարագրվել է ձևափոխություն, որը կամայական ծրագիր ձևափոխում է երկու հավասարումից կազմված ծրագրի՝ պահպանելով *SNFR** դասի ալգորիթմների վրա հիմնված պրոցեդուրային սեմանտիկաները, [3], [4]:
3. Ապացուցվել է, որ ձևափոխությունները, որոնք հիմնված են տեղադրումների վրա, պահպանում են *ACT* և *ACT** ալգորիթմների վրա հիմնված պրոցեդուրային սեմանտիկաները, իսկ մնացած հիմնական ոչ լրիվ պրոցեդուրային սեմանտիկաները չեն պահպանում: Ապացուցվել է նաև, որ վերը նշված ձևափոխություններով կամայական ծրագրի ձևափոխումը մինիմալ քանակությամբ հավասարումներից կազմված ծրագրի NP դժվար խնդիր է, և որ երկու տարբեր մինիմալ ծրագրերի հիմնական պրոցեդուրային սեմանտիկաները, ընդհանուր առմամբ, չեն համընկնում, [1]:

## ԱՏԵՆԱԽՈՍՈՒԹՅԱՆ ՇՐՋԱՆԱԿՆԵՐՈՒՄ ՀՐԱՏԱՐԱԿՎԱԾ ԱՇԽԱՏԱՆՔՆԵՐԻ ՑԱՆԿԸ


1. Ghazaryan G.A. Minimization of number of equations in functional program is NP hard problem. *Proceedings of the Conference on Computer Science and Information Technologies (CSIT-2007), Publishing House of NAS of RA, Yerevan, 2007, p.41-42.*
2. Ghazaryan G.A. On incomparability of some procedural semantics of untyped functional programs. *Proceedings of the Conference on Computer Science and Information Technologies (CSIT-2009), Publishing House of NAS of RA, Yerevan, 2009, p.35-37.*
3. Ghazaryan G.A. Some representation of untyped functional programs. *Четвертая Годичная Научная Конференция, сб. научных статей: физико-математические и естественные науки, РАУ, 2009, с.164-170.*
4. Ghazaryan G.A. On transformations of untyped functional programs and their procedural semantics. *Proceedings of the Yerevan State University, Physical and Mathematical Sciences, Yerevan State University Press, Yerevan, 2011, №1, p.36-43.*

# РЕЗЮМЕ

Казарян Гор Арамович

## “О преобразованиях бестиповых функциональных программ и их процедурных семантиках”

Диссертационная работа посвящена проблемам функционального программирования. Объектом исследования является бестиповая функциональная программа, которая представляет собой систему уравнений (с отделяющимися переменными) в бестиповом λ-исчислении. Семантика неподвижной точки (основная семантика) бестиповой функциональной программы определяется с помощью комбинатора неподвижной точки Y. Процедурные семантики бестиповых функциональных программ основаны на алгоритмах интерпретации, использующих два вида операции: подстановку и одношаговую β-редукцию. Известно, что процедурные семантики, основанные на таких алгоритмах интерпретации являются непротиворечивыми. Если процедурная семантика совпадает с основной семантикой программ, то она называется полной, в противном случае – не полной.

Из известных алгоритмов интерпретации рассматриваются $ACT$(активный), $PAS$ (пассивный), $HNFR$ (подстановка и редукция к головной нормальной форме), а так же класс алгоритмов $SNFR$ (подстановка и редукция к нормальной форме), и конкретные примеры из этого класса – $LSNFR$ (левая подстановка и редукция к нормальной форме), $RSNFR$ (правая подстановка и редукция к нормальной форме), $FSNFR$ (полная подстановка и редукция к нормальной форме). В данной работе описывается преобразованный вариант алгоритма $ACT - ACT^*$, который в отличие от $ACT$ вычисляет аргумент только в том случае, когда его значение подставляется вместо по меньшей мере одного свободного вхождения переменной. Известно, что процедурные семантики, основанные на алгоритмах $PAS$ и $HNFR$ являются полными, а процедурные семантики, основанные на алгоритмах класса $SNFR$ и алгоритме $ACT$ не являются полными. Процедурные семантики, основанные на алгоритмах интерпретации $LSNFR$, $RSNFR$, $FSNFR$, $ACT$, $ACT^*$, $PAS$, $HNFR$ мы называем основными процедурными семантиками.

Известно, что основные неполные процедурные семантики (за исключением $ACT^*$) попарно не сравнимы. Представляет интерес исследование этой проблемы в классе программ состоящих из одного уравнения.

В цельях оптимизации, часто бывает необходимо преобразовывать программу, т.е. представлять ее другой, эквивалентной ей программой. Очевидно, что можно преобразовать произвольную программу в эквивалентную ей программу, состоящую из одного уравнения. Соответсвенно возникает вопрос: возможно ли преобразовать произволную программу в программу состоящую из одного уравнения, которая бы сохраняла все основные процедурные семантики?

В работе рассматриваются преобразования, основанные на постановках. Возникает вопрос: сохраняются ли все основные неполные процедурные семантики после таких

приобразований? Представляет так же интерес задача минимизации числа уравнений функциональных программ, с помощью этих преобразований.

В работе получены следующие основные результаты:

1. Доказано, что процедурные семантики, одна из которых основана на алгоритме $ACT$, а другая - на одном из алгоритмов класса $SNFR$, сравнимы в классе программ, состоящих из одного уравнения. Доказано, что процедурные семантики, основанные на алгоритмах $LSNFR$, $FSNFR$, попарно несравнимы в классе программ, состоящих из одного уравнения. Доказано также, что процедурные семантики, одна из которых использует алгоритм интерпретации $ACT^*$, а другая – любой алгоритм класса $SNFR$, несравнимы в классе программ, состоящих из одного уравнения, [2].
2. Доказано, что существует такая программа, которая не имеет эквивалентной ей программы, состоящей из одного уравнения, которая сохраняет все основные процедурные семантики. Описывается преобразование, которое по каждой программе строит эквивалентную ей программу, состоящую из двух уравнений, сохраняющую процедурные семантики, основанные на алгоритмах класса $SNFR^*$ (алгоритмы класса SNFR, которые удовлетворяют некоторому условию), [3], [4].
3. Доказано, что те преобразования, которые основаны на подстановках, в общем случае не сохраняют основные процедурные семантики, и преобразование с помощью этих преобразований произвольной программы в программу, состоящую из минимального количества уравнений, является NP трудной задачей. Доказано также, что основные процедурные семантики двух разных минимальных программ, в общем случае не совпадают, [1].

ABSTRACT

Gor A. Ghazaryan

"On transformations of untyped functional programs and their procedural semantics"

The thesis is devoted to the problems of functional programming. The main object of the research is the untyped functional program, which is defined as a system of equations (with separated variables) in the untyped λ-calculus. The fixed-point semantics (main semantics) of the untyped functional program is defined by the fixed-point combinator Y. The procedural semantics of untyped functional programs are based on interpretation algorithms that use two types of operations: substitution and one-step β-reduction. It is known that procedural semantics based on such interpretation algorithms are consistent. If procedural semantics match with the main semantics of the programs, then these procedural semantics are called complete, and in the opposite case, these procedural semantics are called incomplete.

Interpretation algorithms $ACT$ (active), $PAS$ (passive), $HNFR$ (head normal form reduction), and also the class of algorithms $SNFR$ (substitution and normal form reduction) and its examples: $LSNFR$ (left substitution and normal form reduction), $RSNFR$ (right substitution and normal form reduction), $FSNFR$ (full substitution and normal form reduction) are regarded. In the thesis, the modification of algorithm $ACT$ is introduced, i.e. the algorithm $ACT^*$, which as opposed to the algorithm $ACT$, calculates the argument only when it should be substituted for at least one of the free occurrences of the variables. It is known that procedural semantics that are based on algorithms $PAS$ and $HNFR$ are complete, and the procedural semantics that are based on algorithms $ACT$ and $SNFR$ are incomplete. In the thesis the procedural semantics based on interpretation algorithms $LSNFR, RSNFR, FSNFR, HNFR, PAS, ACT, ACT^*$ are called main procedural semantics.

It is known that the main incomplete procedural semantics (except $ACT^*$) are pairwise incomparable. It is interesting to investigate this problem within the class of programs composed of one equation.

Often for optimization purposes, it is necessary to transform the program, i.e. to represent a program by another equivalent program. It is evident that any program can be transformed to an equivalent program composed of one equation. Consequently, a question arises: whether it is possible to transform any program to a program composed of one equation, which preserves the main procedural semantics?

In this work transformations that are based on substitutions are regarded. A question arises: whether the main incomplete procedural semantics remain the same after such transformations? It is also interesting to investigate the minimization problem of the number of equations of functional programs based on such transformations.

The main results of this work are the followings:

1. It was proved that procedural semantics, the one based on algorithm $ACT$ (active), the other based on any algorithm of class $SNFR$ (substitution and normal form reduction), are comparable within the class of programs composed of one equation. It was proved that

procedural semantics that are based on algorithms $LSNFR$ (left substitution and normal form reduction), $RSNFR$ (right substitution and normal form reduction), $FSNFR$ (full substitution and normal form reduction) are pairwise incomparable within the class of programs composed of one equation. It was proved also that procedural semantics, the one using interpretation algorithm $ACT^*$, the other using any algorithm of class $SNFR$, are incomparable within the class of programs composed of one equation, [2].

2. It was proved that there exists such a program that has not an equivalent program composed of one equation, such that the main procedural semantics remain the same. The transformation which represents programs by two equations, such that procedural semantics using algorithms of class $SNFR^*$ (algorithms of class $SNFR$ that satisfy to a certain condition) remain the same, was introduced, [3], [4].
3. It was proved that transformations that are based on substitutions, in general do not preserve the main procedural semantics of programs, and that the transformation of any program into a program composed of minimal number of equations using this type of transformations, is NP-hard problem. It was proved also that the main procedural semantics of two different minimal programs in general do not match, [1].