ՀՀ ԳԱԱ ԻՆՖՈՐՄԱՏԻԿԱՅԻ ԵՎ ԱՎՏՈՄԱՏԱՑՄԱՆ
ՊՐՈԲԼԵՄՆԵՐԻ ԻՆՍՏԻՏՈՒՏ

Մարտիրոսյան Գևորգ Արթուրի

ԾՐԱԳՐԱՎՈՐՄԱՆ ՖՈՒՆԿՑԻՈՆԱԼ ԼԵԶՈՒՆԵՐԻ ՆԵՐԴՐՎԱԾ ՀԱՍՏԱՏՈՒՆՆԵՐԻ ՄԱՍԻՆ

Ե.13.04 - <<Հաշվողական մեքենաների, համալիրների, համակարգերի և ցանցերի մաթեմատիկական և ծրագրային ապահովում>> մասնագիտությամբ ֆիզիկամաթեմատիկական գիտությունների թեկնածուի գիտական աստիճանի հայցման ատենախոսություն

ՍԵՂՄԱԳԻՐ

Երևան - 2013

ИНСТИТУТ ПРОБЛЕМ ИНФОРМАТИКИ И АВТОМАТИЗАЦИИ НАН РА

Мартиросян Геворг Артурович

О ВСТРОЕННЫХ КОНСТАНТАХ ФУНКЦИОНАЛЬНЫХ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ

АВТОРЕФЕРАТ

Диссертации на соискание ученой степени кандидата физико-математических наук по специальности 05.13.04 – “Математическое и программное обеспечение вычислительных машин, комплексов, систем и сетей”

Ереван – 2013

Ատենախոսության թեման հաստատվել է Երևանի պետական համալսարանում

Գիտական ղեկավար՝	ֆիզ.մաթ.գիտ.դոկտոր Ս.Ա.Նիգիյան
Պաշտոնական ընդդիմախոսներ՝	ֆիզ.մաթ.գիտ.դոկտոր Ի.Դ.Զասլավսկի
	ֆիզ.մաթ.գիտ.դոկտոր Ա.Ա.Չուբարյան
Առաջատար կազմակերպություն՝	Հայաստանի պետական ճարտարագիտական համալսարան

Պաշտպանությունը կայանալու է 2013թ. հունիսի 21-ին, ժ 15:00-ին ՀՀ ԳԱԱ Ինֆորմատիկայի և ավտոմատացման պրոբլեմների ինստիտուտում գործող 037 «Ինֆորմատիկա և հաշվողական համակարգեր» մասնագիտական խորհրդի նիստում հետևյալ հասցեով՝ Երևան, 0014, Պ. Սևակի 1:

Ատենախոսությանը կարելի է ծանոթանալ ՀՀ ԳԱԱ ԻԱՊԻ գրադարանում:

Սեղմագիրը առաքված է 2013թ. մայիսի 21-ին:

Մասնագիտական խորհրդի
գիտական քարտուղար, ֆ.մ.գ.դ. Հ. Գ. Սարուխանյան

Тема диссертации утверждена в Ереванском государственном университете

Научный руководитель:	доктор физ.мат.наук	С.А.Нигиян
Официальные оппоненты:	доктор физ.мат.наук	И.Д.Заславский
	доктор физ.мат.наук	А.А.Чубарян
Ведущая организация:	Государственный инженерный университет Армении	

Защита состоится 21 июня 2013г. в 15:00 на заседании специализированного совета 037 «Информатика и вычислительные системы» Института проблем информатики и автоматизации НАН РА по адресу: 0014, г. Ереван, ул. П. Севака 1.

С диссертацией можно ознакомиться в библиотеке ИПИА НАН РА.

Автореферат разослан 21-го мая 2013г.

Ученый секретарь специализированного
совета, д.ф.м.н. А. Г. Саруханян

ԱՇԽԱՏԱՆՔԻ ԸՆԴՀԱՆՈՒՐ ԲՆՈՒԹԱԳԻՐԸ

Թեմայի արդիականությունը: Աշխատանքը նվիրված է ծրագրավորման ֆունկցիոնալ լեզուների հատկություններին: Հետազոտության առարկա են հանդիսանում ծրագրավորման ֆունկցիոնալ լեզուների ներդրված հաստատունների ենթաբազմությունները, իսկ դիտարկվող հիմնական խնդիրները՝ այդ ենթաբազմությունների լրիվությունը և մինիմալությունը:

ՄակԿարտիի[1] աշխատության մեջ սահմանվում է *Lisp* լեզուն, որը հանդիսանում է ծրագրավորման առաջին ֆունկցիոնալ լեզուն: *Lisp* լեզվի առարկայական տիրույթը *S-expressions* (ատոմներ և դրանցով կազմված ցուցակներ) բազմությունն է: Այդ աշխատության մեջ սահմանվում են *S-expressions* բազմության վրա որոշված ներդրված տարրական ֆունկցիաներ՝ $car, cdr, cons, atom, eq,$ if_then_else: *Lisp* լեզվի հիմնական հասկացությունների հիման վրա են առաջացել $Scheme^2, Scala^3, ML^4, SASL^5$ և այլ լեզուներ: Այս լեզուների ներդրված ֆունկցիաների բազմությունների մեջ կան վերը նշված տարրական ֆունկցիաները:

Բեկուսի[6] աշխատության մեջ սահմանվում է *FP* ծրագրավորման ֆունկցիոնալ լեզուն, որի առարկայական տիրույթը նույնպես *S-expressions* բազմությունն է: *FP* լեզվում չեն օգտագործվում զրո կարգի փոփոխականներ (առարկայական փոփոխականներ), չի օգտագործվում λ-աբստրակցիա և օգտագործվում են առաջին կարգի փոփոխականներ (ֆունկցիոնալ փոփոխականներ): *FP* լեզվի ներդրված հաստատունների բազմությանն են պատկանում *id* (*Identity*), *hd* (*Head*), *tl* (*Tail*), *apndl* (*Append left*), *eq* (*Equals*) մեկ տեղանի ֆունկցիաները և *comp* (*Composition*), *constr* (*Construction*), *cond* (*Condition*), *const* (*Constant*) ֆունկցիոնալները, ինչպես նաև այլ ֆունկցիաներ և ֆունկցիոնալներ:

Հետաքրքրություն է ներկայացնում տրված ծրագրավորման ֆունկցիոնալ լեզվի ներդրված հաստատուններից ընտրել ենթաբազմություն, որը կլինի լրիվ, այսինքն՝ օգտագործելով միայն այդ հաստատունները հնարավոր կլինի կամայական $f : B \longrightarrow S\text{-}expressions$ $(B \subset S\text{-}expressions^k, k \geq 1)$ հաշվարկելի ֆունկցիա ծրագրավորել այդ լեզվում:

---

[1] *McCarthy J.* Recursive functions of symbolic expressions and their computation by machine. Communications of the ACM, Vol. 3, N 4, 1960, p. 184-195.

[2] *Dybvig R. K.* The Scheme Programming Language. MIT Press. 4 edition, 2009, 504p.

[3] *Wampler D.*, *Payne A.* Programming Scala. O'Reilly Media, 2009, 450p.

[4] *Harper R.* Programming in Standard ML. Carnegie Mellon University, 2005, 297p.

[5] *Tuner D. A.* A New Implementation Technique for Applicative Languages. Software - Practice and Experience - SPE , Vol. 9, N 1, 1979, p. 31-49.

[6] *Backus J. W.* Can programming be liberated from the von Neumann style? A functional style and its algebra of programs. Communications of the ACM, v. 21, N 8, 1978, p. 613□641.

Բոյերի և Մուրի[1] աշխատանքում ցույց է տրվում, որ կամայական $f : B \longrightarrow S\text{-}expressions$ $(B \subset S\text{-}expressions^k, k \geq 1)$ հաշվարկելի ֆունկցիայի համար գոյություն ունի $Lisp$ լեզվով P ծրագիր, որի համար տեղի ունի հետևյալը. Եթե f ֆունկցիային համապատասխանող Թյուրինգի մեքենան կանգ է առնում, ապա ակտիվ ինտերպրետատորը P ծրագրի համար կանգ է առնում նույն պատասխանով, իսկ եթե Թյուրինգի մեքենան անվերջ է աշխատում, ապա ակտիվ ինտերպրետատորը անվերջ է աշխատում: Սակայն Բոյերի և Մուրի աշխատանքում ցույց չի տրվում, որ P ծրագրի անշարժ կետի սեմանտիկան հենց այդ ֆունկցիան է: Ուստի խնդիր է առաջանում ուսումնասիրել այս հարցը:

Վերը նշված խնդրից անմիջապես հետևում է ընտրված ներդրված հաստատունների բազմության մինիմալության հարցը: Այսինքն՝ արդյոք հնարավոր չէ ընտրված բազմությունը փոքրացնել (որևէ ներդրված հաստատուն դուրս հանել այդ բազմությունից՝ պահպանելով բազմության լրիվությունը):

Աշխատանքի նպատակն ու խնդիրները: Այս աշխատանքի հիմնական նպատակն ու խնդիրները հետևյալն են.

1. $Lisp$ ծրագրավորման լեզվի ներդրված ֆունկցիաների բազմությունից ընտրել լրիվ և մինիմալ ենթաբազմություն:

2. FP ծրագրավորման լեզվի ներդրված ֆունկցիաների և ֆունկցիոնալների բազմությունից ընտրել լրիվ և մինիմալ ենթաբազմություն:

3. Դիտարկել FP ծրագրավորման լեզվի մոդիֆիկացիաներ: Այդ լեզուների ներդրված ֆունկցիաների և ֆունկցիոնալների բազմություններից ընտրել լրիվ և մինիմալ ենթաբազմություններ:

Հետազոտության մեթոդները: Աշխատանքում օգտագործված հետազոտության մեթոդները ներառում են ֆունկցիոնալ ծրագրավորման, տիպիզացված λ-հաշվի, ալգորիթմների տեսության և հանրահաշվի մեթոդները:

Արդյունքների նորությունը: Աշխատանքում կատարված հետազոտությունների արդյունքում ստացվել են հետևյալ հիմնական արդյունքները.

1. Ներդրված ֆունկցիաների $\Phi = \{car, cdr, cons, atom, eq, if\_then\_else\}$ բազմության համար ստացվել է հետևյալը.
   - Ցույց է տրվել, որ կամայական $f : B \longrightarrow S\text{-}expressions$ $(B \subset S\text{-}expressions^k, k \geq 1)$ հաշվարկելի ֆունկցիա ծրագրավորելի է ծրագրավորման ֆունկցիոնալ լեզվում, որն օգտագործում է Φ բազմության ֆունկցիաները: Ցույց է տրվել, որ երկուսից ավելի ատոմների դեպքում Φ բազմությունը մինիմալ է:
   - Ցույց է տրվել, որ երկու ատոմների դեպքում կամայական $f : B \longrightarrow S\text{-}expressions$ $(B \subset S\text{-}expressions^k, k \geq 1)$ հաշվարկելի ֆունկցիա ծրագրավորելի է ծրագրավորման ֆունկցիոնալ լեզվում, որն օգտագործում է $\Phi \setminus \{eq\}$ բազմության ֆունկցիաները: Ցույց է տրվել, որ $\Phi \setminus$

[1] *Boyer R. S., Moore J. S.* A Mechanical Proof of the Turing Completeness of Pure Lisp. Automated Theorem Proving: After 25 Years, American Mathematical Soc. 1984, p. 133-167.

$\{eq\}$ բազմությունը մինիմալ է: Ապացուցվել է նաև, որ կամայական $\varphi \in \Phi \setminus \{eq\}$ համար ոչ բոլոր $f : B \longrightarrow S\text{-}expressions$ $(B \subset S\text{-}expressions^k, k \geq 1)$ հաշվարկելի ֆունկցիաներն են ծրագրավորելի լեզվում, որն օգտագործում է $\Phi \setminus \{\varphi\}$ բազմության ֆունկցիաները:

2. Ներդրված հաստատունների $\Phi = \{id, hd, tl, apndl, eq\} \cup \{comp, constr, const, cond\}$ բազմության համար ստացվել է հետևյալը.
   - Ցույց է տրվել, որ կամայական $f : B \longrightarrow S\text{-}expressions$ $(B \subset S\text{-}expressions)$ հաշվարկելի ֆունկցիա ծրագրավորելի է FP լեզվում, որն օգտագործում է Φ բազմության հաստատունները: Ցույց է տրվել, որ երեքից ավելի ատոմների դեպքում Φ բազմությունը մինիմալ է:
   - Ցույց է տրվել, որ երեք ատոմների դեպքում կամայական $f : B \longrightarrow S\text{-}expressions$ $(B \subset S\text{-}expressions)$ հաշվարկելի ֆունկցիա ծրագրավորելի է FP լեզվում, որն օգտագործում է $\Phi \setminus \{const\}$ բազմության հաստատունները: Ցույց է տրվել, որ $\Phi \setminus \{const\}$ բազմությունը մինիմալ է: Ապացուցվել է նաև, որ կամայական $\varphi \in \Phi \setminus \{const\}$ համար ոչ բոլոր $f : B \longrightarrow S\text{-}expressions$ $(B \subset S\text{-}expressions)$ հաշվարկելի ֆունկցիաներն են ծրագրավորելի FP լեզվում, որն օգտագործում է $\Phi \setminus \{\varphi\}$ բազմության հաստատունները:
3. Դիտարկվել են FP լեզվի մոդիֆիկացիաներ: Ցույց է տրվել, որ կամայական $f : B \longrightarrow S\text{-}expressions$ $(B \subset S\text{-}expressions)$ հաշվարկելի ֆունկցիա ծրագրավորելի է այդ լեզուներում: Ցույց է տրվել, որ օգտագործված ներդրված հաստատունների բազմությունները մինիմալ են:

Տեսական և կիրառական նշանակությունը: Այս աշխատանքում ստացված արդյունքներն ունեն ինչպես տեսական, այնպես էլ կիրառական նշանակություն: Դրանք կարող են օգտագործվել ֆունկցիոնալ ծրագավորման նոր 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, *Computer Science and Information Technologies* – 2011 (*CSIT*, Երևան, 2011) միջազգային գիտաժողովում: Ատենախոսության հիմնական արդյունքներն ընդգրկված են հրատարակված երեք աշխատանքներում:

Աշխատանքի կառուցվածքն ու համառոտ բովանդակությունը: Ատենախոսությունը բաղկացած է ներածությունից, երեք գլխից, եզրակացությունից և օգտագործված գրականության ցանկից (33 անուն): Ատենախոսության ծավալը 120 էջ է:

ԱՇԽԱՏԱՆՔԻ ԲՈՎԱՆԴԱԿՈՒԹՅՈՒՆԸ

Ներածությունում նկարագրվում է ատենախուսության հետազոտության հիմնական նպատակն ու խնդիրները, հիմնավորվում է ատենախոսության թեմայի արդիականությունը և նորությունը: Համառոտ կերպով ներկայացվում է աշխատանքի բովանդակությունը:

Գլուխ 1-ում բերվում են անհրաժեշտ սահմանումներ և արդյունքներ ֆունկցիոնալ ծրագրավորման վերաբերյալ, որոնք հիմնականում վերցված են [1,2,3,4] աշխատանքներից: Առաջին գլուխը բաղկացած է երեք բաժիններից:

1.1 բաժնում ներմուծված են տիպի, թերմի, թերմի արժեքի, β և δ-ռեդուկցիաների, $\beta\delta$-ռեդուկցիայի, ծրագրի, ծրագրի անշարժ կետի և պրոցեդուրային սեմանտիկաների գաղափարները:

Դիցուք M-ը մասնակի կարգավորված բազմություն է, որտեղ $\perp \in M$ և ցանկացած $m \in M$ համար $\perp \sqsubseteq m$ և $m \sqsubseteq m$:

Սահմանում 1.1.4: Սահմանենք տիպերի $Types$ բազմությունը:

1. $M \in Types$:

2. Եթե $\alpha_1, \dots, \alpha_n, \beta \in Types$, $n \geq 1$, ապա $[\alpha_1 \times \dots \times \alpha_n \to \beta] \in Types$, որտեղ $[\alpha_1 \times \dots \times \alpha_n \to \beta]$-ով նշանակված է $\alpha_1 \times \dots \times \alpha_n$-ից β բոլոր մոնոտոն արտապատկերումների բազմությունը:

Սահմանում 1.1.5: Դիցուք $\alpha \in Types$: α տիպին համապատասխանեցնենք բնական թիվ հետևյալ կերպ, որը կանվանենք α տիպի կարգ (կնշանակենք $ord(\alpha)$).

1. Եթե $\alpha = M$, ապա $ord(\alpha) = 0$:

2. Եթե $\alpha_1, \dots, \alpha_n, \beta \in Types, n \geq 1$, ապա $ord([\alpha_1 \times \dots \times \alpha_n \to \beta]) = max(ord(\alpha_1), \dots, ord(\alpha_n), ord(\beta)) + 1$:

Սահմանում 1.1.6: Սահմանենք թերմ հասկացությունը: V_α-ով նշանակենք α տիպի փոփոխականների հաշվելի բազմությունը: Բոլոր փոփոխականների բազմությունը նշանակենք V-ով՝ $V = \bigcup_{\alpha \in Types} V_\alpha$: Եթե $c \in \alpha, x \in V_\alpha, ord(\alpha) = n, n \geq 0$, ապա կասենք, որ c-ն n-րդ կարգի հաստատուն է, x-ը n-րդ կարգի փոփոխական է: Բոլոր թերմերի բազմությունը նշանակենք Λ-ով՝ $\Lambda = \bigcup_{\alpha \in Types} \Lambda_\alpha$, որտեղ Λ_α-ով նշանակված է α տիպի թերմերի բազմությունը:

---

[1] *Budaghyan L. E.* Formalizing the notion of □-reduction in monotonic models of typed □-calculus. Algebra, Geometry & Their Applicatiions, Vol. 1, YSU Press, 2002, p. 48-57.

[2] *Нигиян С.А.* Функциональные языки программирования. Программирование, № 5, 1991, с. 77-86.

[3] *Нигиян С.А.* Об интерпретации функциональных языков программирования. Программирование, № 2, 1993, с. 58-68.

[4] *Նիգիյան Ս.Ա., Բուդաղյան Լ.Է.* Ծրագրավորման ֆունկցիոնալ համակարգեր: ԵՊՀ Հրատարակչություն, 2006, 56 էջ:

1. Եթե $c \in \alpha, \alpha \in Types$, ապա $c \in \Lambda_\alpha$:
2. Եթե $x \in V_\alpha, \alpha \in Types$, ապա $x \in \Lambda_\alpha$:
3. Եթե $\tau \in \Lambda_{[\alpha_1 \times \ldots \times \alpha_k \to \beta]}$, $t_i \in \Lambda_{\alpha_i}$, $\alpha_i, \beta \in Types$, $i = 1, .., k$, $k \geq 1$, ապա $\tau(t_1, \ldots, t_k) \in \Lambda_\beta$:
4. Եթե $\tau \in \Lambda_\beta$, $x_i \in V_{\alpha_i}, \alpha_i, \beta \in Types, i \neq j \Rightarrow x_i \neq x_j, i, j = 1, \ldots, k, k \geq 1$, ապա $\lambda x_1 \ldots x_k[\tau] \in \Lambda_{[\alpha_1 \times \ldots \times \alpha_k \to \beta]}$: $\lambda x_1 \ldots x_k$-ն կանվանենք աբստրակտոր, իսկ τ-ն՝ աբստրակտորի ազդեցության տիրույթ:

Սահմանում 1.1.7: Դիցուք $t \in \Lambda$: x փոփոխականի ֆիքսված մուտքը t թերմում կանվանենք կապված, եթե այն կամ պատկանում է որևէ աբստրակտորին, կամ գտնվում է այդ փոփոխականն օգտագործող աբստրակտորի ազդեցության տիրույթում: Հակառակ դեպքում x փոփոխականի մուտքը կանվանենք ազատ: Կասենք, որ x փոփոխականը ազատ է t թերմում, եթե այն ունի առնվազն մեկ ազատ մուտք: t թերմի ազատ փոփոխականների բազմությունը կնշանակենք $FV(t)$-ով:

Սահմանում 1.1.8: Դիցուք $t \in \Lambda$ և $\sigma = \{t_1/x_1, \ldots, t_n/x_n\}$ որևէ տեղադրում է: $t\sigma$-ով կնշանակենք այն t' թերմը, որը ստացվում է t թերմում $x_1, \ldots, x_n$ փոփոխականների բոլոր ազատ մուտքերը միաժամանակ փոխարինելով համապատասխանաբար $t_1, \ldots, t_n$ թերմերով: Այդ դեպքում կասենք, որ t' թերմը ստացվել է t թերմից σ տեղադրման կիրառման արդյունքում:

Սահմանում 1.1.9: Դիցուք $t \in \Lambda$ և $\sigma = \{t_1/x_1, \ldots, t_n/x_n\}$ որևէ տեղադրում է: Կասենք, որ σ տեղադրության կիրառումը t թերմի վրա թույլատրելի է, եթե ցանկացած $i = 1, \ldots, n$ համար x_i փոփոխականի ոչ մի ազատ մուտք t թերմում չի գտնվում t_i թերմի ազատ փոփոխականներն օգտագործող որևէ աբստրակտորի ազդեցության տիրույթում:

Մենք կդիտարկենք միայն տեղադրումների թույլատրելի կիրառումներ:

Սահմանում 1.1.11: t_1 և t_2 թերմերը կանվանենք կոնգրուենտ (կնշանակենք $t_1 \equiv t_2$), եթե t_2 թերմը ստացվում է t_1 թերմից $n \geq 0$ անգամ կապված փոփոխականի վերանվանումով: Պայմանավորվենք հետագայում կոնգրուենտ թերմերն իրարից չտարբերել:

Սահմանում 1.1.12: Յուրաքանչյուր թերմին համապատասխանության մեջ դնենք $Val_{\bar{y}_0}(t)$ հաստատունը, որտեղ $t \in \Lambda_\alpha, \alpha \in Types, FV(t) \subset \{y_1, \ldots, y_n\}, y_i \in V_{\alpha_i}, \bar{y}_0 =< y_1^0, \ldots, y_n^0 >, y_i^0 \in \alpha_i, \alpha_i \in Types, i = 1, \ldots, n, n \geq 0$:

1. Եթե $t \equiv c, c \in \alpha, \alpha \in Types$, ապա $Val_{\bar{y}_0}(t) = c$:
2. Եթե $t \equiv x, x \in V$, ապա $Val_{\bar{y}_0}(t) = y_i^0$, այն դեպքում, երբ $x = y_i, 1 \leq i \leq n$:
3. Եթե $t \equiv \tau(t_1, \ldots, t_k) \in \Lambda_\beta$, $\tau \in \Lambda_{[\alpha_1 \times \ldots \times \alpha_k \to \beta]}$, $t_i \in \Lambda_{\alpha_i}$, $\alpha_1, \ldots, \alpha_k, \beta \in Types, i = 1, .., k, k \geq 1$,ապա $Val_{\bar{y}_0}(\tau(t_1, \ldots, t_k)) = Val_{\bar{y}_0}(\tau)\left(Val_{\bar{y}_0}(t_1), \ldots, Val_{\bar{y}_0}(t_k)\right)$
4. Եթե $t \equiv \lambda x_1 \ldots x_k[\tau] \in \Lambda_{[\alpha_1 \times \ldots \times \alpha_k \to \beta]}, \tau \in \Lambda_\beta$, $x_i \in V_{\alpha_i}, i = 1, .., k, k \geq 1, \alpha_1, \ldots, \alpha_k, \beta \in Types, i \neq j \Rightarrow x_i \neq x_j$, $i, j = 1, \ldots, k$, ապա $Val_{\bar{y}_0}(\lambda x_1 \ldots x_k[\tau]) \in [\alpha_1 \times \ldots \times \alpha_k \to \beta]$ և կամայական $\bar{x}_0 =< x_1^0, \ldots, x_k^0 >, x_j^0 \in \alpha_j, j = 1, \ldots, k$ համար ունենք $Val_{\bar{y}_0}(\lambda x_1 \ldots x_k[\tau])(\bar{x}_0) = Val_{\bar{x}_0\, \bar{y}_0'}(\tau)$, որտեղ $FV(\lambda x_1 \ldots x_k[\tau]) = \{y_{i_1}, \ldots, y_{i_s}\}$, $s \geq 0$, $\bar{y}_0' =< y_{i_1}^0, \ldots, y_{i_s}^0 >$:

Սահմանում 1.1.13: Դիցուք $t_1, t_2 \in \Lambda$ և $FV(t_1) \cup FV(t_2) = \{y_1, \dots, y_n\}, y_i \in V_{\alpha_i}, \alpha_i \in Types, i = 1, \dots, n,\ n \geq 0$: Կասենք, որ t_1 և t_2 թերմերը համարժեք են (կնշանակենք $t_1 \sim t_2$), եթե կամայական $\bar{y}_0 =< y_1^0, \dots, y_n^0 >$ համար, որտեղ $y_i^0 \in \alpha_i, i = 1, \dots, n$, տեղի ունի հետևյալը. $Val_{\bar{y}_0}(t_1) = Val_{\bar{y}_0}(t_2)$:

Դիտարկենք հետևյալ հավասարումների համակարգը՝

$$\begin{cases} F_1 = \tau_1 \\ \dots \\ F_n = \tau_n \end{cases} \qquad (1)$$

որտեղ $F_i \in V_{\alpha_i}, i \neq j \Rightarrow F_i \neq F_j, \tau_i \in \Lambda_{\alpha_i}, FV(\tau_i) \subset \{F_1, \dots, F_n\}, \alpha_i \in Types, i, j = 1, \dots, n, n \geq 1$:

Սահմանում 1.1.15: $\Psi_P: \alpha_1 \times \dots \times \alpha_n \to \alpha_1 \times \dots \times \alpha_n$ արտապակերումը, որը սահմանվում է հետևյալ կերպ $\Psi_P(\bar{g}) =< Val_{\bar{g}}(\tau_1), \dots, Val_{\bar{g}}(\tau_n) >$, որտեղ $\bar{g} \in \alpha_1 \times \dots \times \alpha_n$, կանվանենք (1) հավասարումների համակարգին համապատասխանող արտապատկերում: $(\bar{g})_i$-ով նշանակենք $\bar{g}$-ի i-րդ կոմպոնենտը$(1 \leq i \leq n)$: Կասենք, որ $\bar{f} \in \alpha_1 \times \dots \times \alpha_n$ հանդիսանում է (1) համակարգի լուծում, եթե $\Psi_P(\bar{f}) = \bar{f}$:

[1]Աշխատանքում ցույց է տրված, որ (1) տեսքի կամայական համակարգ ունի փոքրագույն լուծում:

Սահմանում 1.1.16: Դիցուք $\xi \in \alpha, \alpha \in Types, ord(\alpha) \geq 2$: ξ արտապատկերումը կանվանենք էֆֆեկտիվ ներկայացվող հաստատուն, եթե գոյություն ունի (1) տեսքի P համակարգ, որտեղ $F_1 \in V_\alpha$ և որը չի օգտագործում > 1 կարգի հաստատուններ, իսկ օգտագործվող 1 կարգի հաստատունները հաշվարկելի ֆունկցիաներ են և, եթե $< f_1, \dots, f_n > P$ համակարգի փոքրագույն լուծումն է, ապա $f_1 = \xi$:

Սահմանում 1.1.17: (1) տեսքի հավասարումների համակարգը կոչվում է ծրագիր, եթե նրանում օգտագործված 1 կարգի հաստատունները հաշվարկելի ֆունկցիաներ են, ≥ 2 կարգի հաստատունները էֆֆեկտիվ ներկայացվող են և $F_1 \in V_{[M^k \to M]}, (k \geq 1)$: $F_1 = \tau_1$ հավասարումը կոչվում է P ծրագրի գլխավոր հավասարում, իսկ $f_p = (\bar{f})_1$ կոչվում է P ծրագրի անշարժ կետի սեմանտիկա, որտեղ $\bar{f}$-ն P ծրագրի փոքրագույն լուծումն է:

Նշանակենք $\Psi_P^0(\bar{\Omega}) = \bar{\Omega},\ \Psi_P^{j+1}(\bar{\Omega}) = \Psi_P(\Psi_P^j(\bar{\Omega}))$, որտեղ $\bar{\Omega}$-ը $\alpha_1 \times \dots \times \alpha_n$ բազմության փոքրագույն էլեմենտն է, $j \geq 0$:

[2]Աշխատանքից հետևում է, որ եթե տրված է P ծրագիր և $\bar{f} =< f_1, \dots, f_n >$-ն P ծրագրի փոքրագույն լուծումն է, ապա $f_i = sup\{(\Psi_P^k(\bar{\Omega}))_i \mid k \geq 0\}$, եթե $ord(f_i) \leq 2, i = 1, \dots, n, n \geq 1$:

Պայմանավորվենք t_τ-ով նշանակել t թերմը, որում ֆիքսված է τ ենթաթերմի մի որևէ մուտք, իսկ $t_{\tau'}$-ով նշանակենք այն թերմը, որը ստացվում է t թերմից ֆիքսված τ ենթաթերմի փոխարինմամբ τ' թերմով:

---

[1] *Нигиян С.А.* Функциональные языки программирования. Программирование, № 5, 1991, с. 77-86.

[2] *Нигиян С.А.* Об интерпретации функциональных языков программирования. Программирование, № 2, 1993, с. 58-68.

Սահմանում 1.1.18: β-ռեդուկցիայի գաղափար կոչվում է զույգերի հետևյալ բազմությունը.

$$\beta = \{< \lambda x_1 \dots x_k[\tau](t_1, \dots, t_k), \tau\{t_1/x_1, \dots, t_k/x_k\} > | \\ x_i \in V_{\alpha_i}, t_i \in \Lambda_{\alpha_i}, \tau \in \Lambda, \alpha_i \in Types, i = 1, .., k, k \geq 1\}$$

ընդ որում $\lambda x_1 \dots x_k[\tau](t_1, \dots, t_k)$-ը կոչվում է β-ռեդեքս, իսկ $\tau\{t_1/x_1, \dots, t_k/x_k\}$-ը կոչվում է այդ ռեդեքսի փաթեթ:

Այսուհետ այս բաժնում կդիտարկենք միայն այնպիսի թերմեր, որոնք չեն պարունակում ≥ 2 կարգի հաստատուններ:

δ-ռեդուկցիայի և իրական δ-ռեդուկցիայի գաղափարները վերցված են [1]աշխատանքից: Այսուհետ կօգտագործենք իրական δ-ռեդուկցիայի գաղափարը և դրա հետ կապված արդյունքները:

δ-ռեդեքսը ունի $f(t_1, \dots, t_k)$ տեսքը, որտեղ $f \in [M^k \to M], t_i \in \Lambda_M, i = 1, \dots, k, k \geq 1$: Նրա փաթեթն կամ $m \in M$ հաստատունը և այդ դեպքում $f(t_1, \dots, t_k) \sim m$, կամ t_i ենթաթերմը, $1 \leq i \leq k$ և այդ դեպքում $f(t_1, \dots, t_k) \sim t_i$:

$\beta \cup \delta$-ռեդուկցիայի գաղափարը նշանակենք $\beta\delta$-ով: Պայմանավորվենք $\beta\delta$-ռեդուկցիան անվանել ռեդուկցիա, $\beta\delta$-ռեդեքսը անվանել ռեդեքս:

Սահմանում 1.1.24: Կասենք, որ t թերմը նորմալ ձև է, եթե նրանում չկա ռեդեքս հանդիսացող ենթաթերմ: Նորմալ ձևերի բազմությունը կնշանակենք NF-ով:

Սահմանում 1.1.34: Սահմանենք FS լրիվ տեղադրման ինտերպրետացիայի ալգորիթմը:

Մուտք: (1) տեսքի P ծրագիր և t թերմ, $FV(t) \subset \{F_1, \dots, F_n\}$:

Ելք: $FS(P, t) \in NF, FV(FS(P, t)) \cap \{F_1, \dots, F_n\} = \emptyset$, եթե FS-ը որոշված է P-ի և t-ի համար:

1. <u>եթե</u> $t \in NF$ և $FV(t) \cap \{F_1, \dots, F_n\} = \emptyset$, <u>ապա</u> t, <u>այլապես եթե</u> $t \notin NF$, <u>ապա</u> անցնել քայլ 2-ին, <u>այլապես</u> անցնել քայլ 3-ին;
2. <u>եթե</u> $t \equiv t_\tau$ և τ-ն t թերմի ամենաձախ ռեդեքսն է, <u>ապա</u> $FS(P, t_{\tau_0})$, որտեղ τ_0-ն τ ռեդեքսի փաթեթն է;
3. $FS(P, t\{\tau_1/F_1, \dots, \tau_n/F_n\})$:

[1]Աշխատանքից հետևում է, որ FS ալգորիթմը լրիվ է և անհակասելի:

1.2 բաժնում սահմանվում են S-*expressions* բազմությունը, S-ֆունկցիա, հաշվարկելի S-ֆունկցիա հասկացությունները:

Սահմանում 1.2.1: Դիցուք $Atoms = \{a_1, \dots, a_n\}$ $(n \geq 2)$ ատոմների բազմություն է: Սահմանենք S-*expressions* բազմությունը:

1. Եթե $t \in Atoms$, ապա $t \in S$-*expressions*:
2. Եթե $t_1, \dots, t_n \in S$-*expressions* $(n \geq 0)$, ապա $(t_1 \dots t_n) \in S$-*expressions*:

Երկրորդ կետով սահմանված էլեմենտներին անվանում են ցուցակներ: Եթե $(t_1 \dots t_n)$ ցուցակ է, ապա t_1-ը կոչվում է ցուցակի գլուխ, $(t_2 \dots t_n)$-ը՝ ցուցակի պոչ, որտեղ $t_1, \dots, t_n \in S$-*expressions* $(n > 0)$:

[1] *Budaghyan L. E.* A Necessary and sufficient condition of completeness of computation rule for strong typed functional programs. Proceedings of the Conference of Computer Science and Information Technologies (CSIT-2005), Publishing House of NAS of RA, 2005, p. 16-19.

Սահմանում 1.2.2: Դիցուք $B \subset S\text{-}expressions^k$ $(k \geq 1)$: $f : B \longrightarrow S\text{-}expressions$ $(k \geq 1)$ ֆունկցիային կանվանենք S-ֆունկցիա:

f S-ֆունկցիան կլինի հաշվարկելի ֆունկցիա, եթե գոյություն ունի Թյուրինգի մեքենա, որն հաշվարկում է այդ ֆունկցիան:

1.3 բաժնում սահմանվում են տրված ֆունկցիոնալ լեզվի ներդրված հաստատունների բազմության լրիվության և մինիմալության հասկացությունները:

Մենք կդիտարկենք ծրագրավորման ֆունկցիոնալ լեզուներ, որոնք որոշվում են $L = (M, C, X, T)$ քառյակով, որտեղ՝

$M = S\text{-}expressions \cup \{\bot\}$ մասնակի կարգավորված բազմությունն է, որտեղ ցանկացած $m \in M$ համար $\bot \sqsubseteq m$ և $m \sqsubseteq m$:

$C = M \cup \Psi$, որտեղ Ψ-ն ≥ 1 կարգի հաստատուններ են, որոնք օգտագործվում են L լեզվի ծրագրերում:

X-ը փոփոխականների բազմությունն է:

$T \subset \Lambda(C, X)$, որտեղ $\Lambda(C, X)$-ը թերմեր են, որոնք կառուցվում են օգտագործելով հաստատուններ և փոփոխականներ միայն C և X բազմություններից:

$\wp(L)$-ով նշանակենք L լեզվի ծրագրերի բազմությունը, որտեղ $F_i \in X, \tau_i \in T, i = 1, \dots, n, n \geq 1$:

Սահմանում 1.3.1: Կասենք, որ $f : B \longrightarrow S\text{-}expressions$ $(B \subset S\text{-}expressions^k, k \geq 1)$ S-ֆունկցիան ծրագրավորելի է $L = (M, C, X, T)$ լեզվում, եթե գոյություն ունի $P \in \wp(L)$ ծրագիր, այնպիսին, որ տեղի ունի հետևյալը.

- եթե $\overline{m} \in B$, ապա $f_P(\overline{m}) = f(\overline{m})$;
- եթե $\overline{m} \in S\text{-}expressions^k \setminus B$, ապա $f_P(\overline{m}) = \bot$;

որտեղ f_P-ն P ծրագրի անշարժ կետի սեմանտիկան է:

Սահմանում 1.3.2: Կասենք, որ ներդրված հաստատունների Ψ բազմությունը լրիվ է $L = (M, C, X, T)$ լեզվի համար, որտեղ $C = M \cup \Psi$, եթե կամայական $f : B \longrightarrow S\text{-}expressions$ $(B \subset S\text{-}expressions^k, k \geq 1)$ հաշվարկելի ֆունկցիա ծրագրավորելի է L լեզվում:

Սահմանում 1.3.3: Կասենք, որ ներդրված հաստատունների Ψ լրիվ բազմությունը մինիմալ է $L = (M, C, X, T)$ լեզվի համար, որտեղ $C = M \cup \Psi$, եթե կամայական $\varphi \in \Psi$ համար ներդրված հաստատունների $\Psi \setminus \{\varphi\}$ բազմությունը լրիվ չէ $L' = (M, C', X, T')$ լեզվի համար, որտեղ $C' = M \cup \Psi \setminus \{\varphi\}$, $T' \subset T$:

Գլուխ 2-ում դիտարկվում են լրիվության և մինիմալության խնդիրները ծրագրավորման ֆունկցիոնալ լեզուների համար, որոնք օգտագործում են *Lisp* լեզվի ներդրված ֆունկցիաներ: Երկրորդ գլուխը բաղկացած է երեք բաժիններից:

2.1 բաժնում տրվում են անհրաժեշտ սահմանումներ և օգտագործված արդյունքներ:

Կդիտարկենք ծրագրավորման ֆունկցիոնալ լեզուներ, որոնք որոշվում են $L = (M, C, V, \Lambda(C, V))$ քառյակով, որտեղ $M = S\text{-}expressions \cup \{\bot\}$, $C = M \cup \{car, cdr, cons, atom, eq, if\_then\_else\}$, $car, cdr, atom \in [M \to M]$, $cons, eq \in [M^2 \to M]$, $if\_then\_else \in [M^3 \to M]$.

$$car(m) = \begin{cases} m_1, \text{ եթե } m = (m_1 \dots m_k), m_i \in S\text{-}expressions, i = 1, \dots, k, k \geq 1 \\ \bot, \text{ մնացած դեպքերում} \end{cases}$$

$$cdr(m)=\begin{cases} nil, & \text{եթե } m=(m_1), m_1 \in S\text{-}expressions \\ (m_2 \dots m_k), & \text{եթե } m=(m_1 \dots m_k), m_i \in S\text{-}expressions, i=1,\dots,k, k>1 \\ \perp, & \text{մնացած դեպքերում} \end{cases}$$

$$cons(m_0,m)=\begin{cases} (m_0), & \text{եթե } m=nil, m_0 \in S\text{-}expressions \\ (m_0\ m_1 \dots m_k), & \text{եթե } m=(m_1 \dots m_k), m_i \in S\text{-}expressions, \\ & \qquad i=0,\dots,k, k\geq 1 \\ \perp, & \text{մնացած դեպքերում} \end{cases}$$

$$atom(m)=\begin{cases} t, & \text{եթե } m \in Atoms \\ nil, & \text{եթե } m \notin Atoms, m \neq \perp \\ \perp, & \text{մնացած դեպքերում} \end{cases}$$

$$eq(m_1,m_2)=\begin{cases} t, & \text{եթե } m_1,m_2 \in Atoms,\ m_1=m_2 \\ nil, & \text{եթե } m_1,m_2 \in Atoms,\ m_1 \neq m_2 \\ \perp, & \text{մնացած դեպքերում} \end{cases}$$

$$if\_then\_else(m_1,m_2,m_3)=\begin{cases} m_2, & \text{եթե } m_1 \in S\text{-}expressions, m_1 \neq nil \\ m_3, & \text{եթե } m_1=nil \\ \perp, & \text{մնացած դեպքերում} \end{cases}$$

δ-ռեդուկցիայի գաղափարը սահմանենք հետևյալ ձևով՝

1. $< f(m_1), m > \in \delta$, որտեղ $f \in \{car, cdr, atom\},\ m_1, m \in M$ և $f(m_1)=m$
2. $< g(m_1,m_2), m > \in \delta$, որտեղ $g \in \{cons, eq\}, m_1, m_2, m \in M$ և $g(m_1,m_2)=m$
3. $< if\ nil\ then\ t_1\ else\ t_2, t_2 > \in \delta$, որտեղ $t_1, t_2 \in \Lambda_M$
4. $< if\ m\ then\ t_1\ else\ t_2, t_1 > \in \delta$, որտեղ $m \in M, m \neq nil, m \neq \perp\ t_1, t_2 \in \Lambda_M$
5. $< if\ \perp\ then\ t_1\ else\ t_2, \perp > \in \delta$, որտեղ $t_1, t_2 \in \Lambda_M$

[1]Աշխատանքից հետևում է, որ δ-ն հանդիսանում է իրական δ-ռեդուկցիայի գաղափար:

Նշանակենք $\Phi = \{car, cdr, cons, atom, eq, if\_then\_else\}$:

2.2 բաժնում ուսումնասիրվում է Φ բազմության լրիվությունը:

Թեորեմ 2.2.1. Ներդրված ֆունկցիաների Φ բազմությունը լրիվ է $L=(M,C,V,\Lambda(C,V))$ լեզվի համար, որտեղ $C=M \cup \Phi$:

Թեորեմ 2.2.1-ը ապացուցելու համար կօգտվենք հետևյալ լեմմայից:

Լեմմա 2.2.1: Դիցուք տրված է P ծրագիրը: $\bar{f} =< f_1, \dots, f_n >$ P ծրագրի փոքրագույն լուծումն է: Ենթադրենք P' ծրագիրը ստացվել է P ծրագրից հետևյալ երկու քայլերը կատարելով.

1. P ծրագրից հեռացնել i-րդ հավասարումը,
2. P ծրագրի մյուս հավասարումներում F_i փոփոխականի բոլոր ազատ մուտքերը փոխարինել f_i-ով,

որտեղ $i=2,\dots,n, n\geq 1$: Այդ դեպքում՝ $f_P = f_{P'}$:

Թեորեմ 2.2.1-ի ապացույցում դիտարկվում է կամայական $f: B \longrightarrow S\text{-}expressions, (B \subset S\text{-}expressions^k, k \geq 1)$ հաշվարկելի ֆունկցիա: Այդ ֆունկցիային

[1] *Budaghyan L. E.* A Necessary and sufficient condition of completeness of computation rule for strong typed functional programs. Proceedings of the Conference of Computer Science and Information Technologies (CSIT-2005), Publishing House of NAS of RA, 2005, p. 16-19.

համապատասխան P ծրագիր է սահմանվում: Այդ ծրագիրը պայմանականորեն կարելի է բաժանել երեք հիմնական տրամաբանական մասերի՝

- Կոդավորում: Տրված $s_1, \dots, s_k \in S\text{-}expressions$ $(k \geq 1)$ համար կառուցվում է ցուցակ, որը համապատասխանում է $s_1, \dots, s_k$-ի ներկայացմանը ժապավենի վրա:
- Կոդավորված ցուցակը փոփոխվում է նույն ձև, ինչ Թյուրինգի մեքենայի ժապավենը՝ աշխատանքի ընթացում:
- Ապակոդավորում: Կոդավորված ցուցակի ապակոդավորում:

P ծրագիրը կազմված է 21 հավասարումներից: Այդ ծրագրի համար կկառուցվի $P_0, P_1, \dots, P_{20}$ ծրագրերի հաջորդականություն, այնպիսին, որ P_0-ն P ծրագիրն է, P_{20} ծրագիրը կազմված է մեկ հավասարումից, կամայական $i = 0, \dots, 19$ համար P_{i+1} ծրագիրը ստացվում է P_i ծրագրից հետևյալ երկու քայլերը կատարելով.

1. P_i ծրագրից հեռացնել երկրորդ հավասարումը,
2. P_i ծրագրի մյուս հավասարումներում երկրորդ հավասարման ձախ մասի փոփոխականի բոլոր ազատ մուտքերը փոխարինել երկրորդ հավասարման փոքրագույն լուծումով:

Կիրառելով Լեմմա 2.2.1-ը կստացվի, որ կամայական $i = 0, \dots, 20$ համար $f_P = f_{P_i}$:

Կամայական $i = 0, \dots, 19$ համար P_i ծրագրի երկրորդ հավասարման աջ մասի ազատ փոփոխականների բազմությունը պարունակում է ամենաշատը մեկ փոփոխական, որը համընկնում է այդ հավասարման ձախ մասում գտնվող փոփոխականի հետ: Ուստի այդ հավասարումը կարող է լուծվել P_i ծրագրի մյուս հավասարումներից անկախ: Յուրաքանչյուր հեռացվող հավասարման փոքրագույն լուծումը որոշվում է ինդուկցիայի մեթոդով:

2.3 բաժնում ուսումնասիրվում է Φ բազմության մինիմալությունը:

Թեորեմ 2.3.1. Ներդրված ֆունկցիաների Φ բազմությունը մինիմալ է $L = (M, C, V, \Lambda(C, V))$, $C = M \cup \Phi$, լեզվի համար, որն օգտագործում է երկուսից ավելի ատոմ:

Թեորեմ 2.3.1-ը ապացուցվում է Լեմմա 2.3.1-2.3.6-ի միջոցով: Յուրաքանչյուր լեմմայում դիտարկվում է Φ բազմության մեկ ֆունկցիա՝ $f \in \Phi$, և ցույց է տրվում, որ $\Phi \setminus \{f\}$ բազմությունը լրիվ չէ $L = (M, C, V, \Lambda(C, V))$ լեզվի համար, որտեղ $C = M \cup (\Phi \setminus \{f\})$:

$L = (M, C, V, \Lambda(C, V))$ լեզվի համար, որտեղ $C = M \cup (\Phi \setminus \{f\}), f \in \Phi$, $\Phi \setminus \{f\}$ բազմության ոչ լրիվությունը ցույց է տրվում հետևյալ ձևով՝

- Դիտարկվում է g հաշվարկելի S-ֆունկցիան:
- Ենթադրվում է, որ g ֆունկցիան ծրագրավորելի է L լեզվում:
- Ցույց է տրվում, որ որոշ մուտքային տվյալների համար FS ինտերպրետացիայի ալգորիթմը լրիվ չէ, ուստի ստացվում է հակասություն, այսինքն՝ g ֆունկցիան ծրագրավորելի չէ L լեզվում:

Թեորեմ 2.3.2. Երկու ատոմ օգտագործող լեզուների համար ունենք՝

1. Ներդրված ֆունկցիաների $\Phi \setminus \{eq\}$ բազմությունը լրիվ է և մինիմալ $L = (M, C, V, \Lambda(C, V))$ լեզվի համար, որտեղ $C = M \cup (\Phi \setminus \{eq\})$:

2. Կամայական $f \in \Phi \setminus \{eq\}$ համար ներդրված ֆունկցիաների $\Phi \setminus \{f\}$ բազմությունը լրիվ չէ $L = (M, C, V, \Lambda(C, V))$ լեզվի համար, որտեղ $C = M \cup (\Phi \setminus \{f\})$:

Թեորեմ 2.3.2-ը ապացուցվում է հետևյալ ձևով: Ցույց է տրվում, որ eq ֆունկցիան ներկայացվում է $L = (M,\ C,\ V,\ \Lambda(C, V))$ լեզվում, որն օգտագործում է երկու ատոմ և $C = M \cup (\Phi \setminus \{eq\})$ ներդրված հաստատունների բազմությունը: Ապացույցի համար օգտագործվում են նաև Լեմմա 2.3.1-2.3.5-ները:

Գլուխ 3-ում դիտարկվում են լրիվության և մինիմալության խնդիրները FP լեզվի համար: Երրորդ գլուխը բաղկացած է չորս բաժիններից:

3.1 բաժնում տրվում են անհրաժեշտ սահմանումներ և օգտագործված արդյունքներ:

Նկարագրենք FP լեզվի $(M,\ C,\ X,\ T)$ քառյակը:

$M = S\text{-}expressions \cup \{\bot\}$, որտեղ $Atoms$ բազմությունը պարունակում է առնվազն երեք ատոմ՝ $true, false, nil \in Atoms$: $true$ և $false$ ատոմները համապատասխանում են տրամաբանական ճիշտ և սխալ արժեքներին, իսկ nil ատոմը համապատասխանում է դատարկ ցուցակին:

Նկարագրենք C բազմությունը: $C = M \cup C_1 \cup C_2$, որտեղ C_1-ն առաջին կարգի հաստատուններն են (ֆունկցիաներ), C_2-ը երկրորդ կարգի հաստատուններն են (ֆունկցիոնալներ): Կօգտագործենք [1]աշխատանքում սահմանված հաստատունները:

$C_1 = \{id, hd, tl, apndl, eq\}$: Նշենք, որ եթե $f \in C_1$, ապա $f \in [M \to M]$: Սահմանենք դրանք ցանկացած $m \in M$ համար.

$$id(m) = m$$

$$hd(m) = \begin{cases} m_1, & \text{եթե } m = (m_1 \dots m_k), m_i \in S\text{-}expressions, i = 1, \dots, k, k \geq 1 \\ \bot, & \text{մնացած դեպքերում} \end{cases}$$

$$tl(m) = \begin{cases} nil, & \text{եթե } m = (m_1), m_1 \in S\text{-}expressions \\ (m_2 \dots m_k), & \text{եթե } m = (m_1 \dots m_k), m_i \in S\text{-}expressions, i = 1, \dots, k, k > 1 \\ \bot, & \text{մնացած դեպքերում} \end{cases}$$

$$apndl(m) = \begin{cases} (m_0\ m_1 \dots m_k), & \text{եթե } m = (m_0\ (m_1 \dots m_k)), m_i \in S\text{-}expressions, \\ & \qquad i = 0, \dots, k, k \geq 1 \\ (m_0), & \text{եթե } m = (m_0\ nil), m_0 \in S\text{-}expressions \\ \bot, & \text{մնացած դեպքերում} \end{cases}$$

$$eq(m) = \begin{cases} true, & \text{եթե } m = (m_1\ m_2), m_1 = m_2,\ m_1, m_2 \in S\text{-}expressions \\ false, & \text{եթե } m = (m_1\ m_2), m_1 \neq m_2,\ m_1, m_2 \in S\text{-}expressions \\ \bot, & \text{մնացած դեպքերում} \end{cases}$$

C_2 բազմությունը բաղկացած է հետևյալ չորս ֆունկցիոնալներից:

$Composition$:

$comp \in [[M \to M]^2 \to [M \to M]]$ և ցանկացած $g, h \in [M \to M]$ ֆունկցիաների համար $comp(g, h) = f \in [M \to M]$, որը որոշվում է հետևյալ կերպ. ցանկացած $m \in M$

[1] *Նիգիյան Ս.Ա., Բուդաղյան Լ.Է.* Ծրագրավորման ֆունկցիոնալ համակարգեր: ԵՊՀ Հրատարակչություն, 2006, 56 էջ:

$$f(m) = g(h(m))$$

$Construction$:

$constr \in [[M \to M]^2 \to [M \to M]]$ և ցանկացած $g, h \in [M \to M]$ ֆունկցիաների համար $constr(g,h) = f \in [M \to M]$, որը որոշվում է հետևյալ կերպ. ցանկացած $m \in M$

$$f(m) = (g(m)\ h(m))$$

$Constant$:

$const \in [M \to [M \to M]]$ և ցանկացած $m_0 \in M$ հաստատունի համար $const(m_0) = f \in [M \to M]$, որը որոշվում է հետևյալ կերպ. ցանկացած $m \in M$

$$f(m) = \begin{cases} m_0, & \text{եթե } m \neq \bot \\ \bot, & \text{եթե } m = \bot \end{cases}$$

$Condition$:

$cond \in [[M \to M]^3 \to [M \to M]]$ և ցանկացած $p, g, h \in [M \to M]$ ֆունկցիաների համար $cond(p,g,h) = f \in [M \to M]$, որը որոշվում է հետևյալ կերպ. ցանկացած $m \in M$

$$f(m) = \begin{cases} g(m), \text{եթե } p(m) = true \\ h(m), \text{եթե } p(m) = false \\ \bot, \quad \text{մնացած դեպքերում} \end{cases}$$

Փոփոխականների բազմությունը հետևյալն է՝ $X = \{F_i \mid F_i \in V_{[M \to M]}, i \geq 1\}$:

T-ն այն թերմերի բազմությունն է, որոնք կառուցվում են օգտագործելով հաստատուններ և փոփոխականներ C և X բազմություններից, թերմի սահմանման 4-րդ կետը չի օգտագործվում, իսկ M բազմության էլեմենտները օգտագործվում են միայն որպես $const$ ֆունկցիոնալի արգումենտի արժեք:

3.2 բաժնում ուսումնասիրվում է $C_1 \cup C_2$ բազմության լրիվությունը:

Թեորեմ 3.2.1. Ներդրված հաստատունների $C_1 \cup C_2$ բազմությունը լրիվ է FP լեզվի համար, որտեղ $C = M \cup C_1 \cup C_2$:

Այս թեորեմի ապացույցը կատարվում է Թեորեմ 2.2.1-ի նման:

3.3 բաժնում ուսումնասիրվում է $C_1 \cup C_2$ բազմության մինիմալությունը:

Թեորեմ 3.3.1. Ներդրված հաստատունների $C_1 \cup C_2$ բազմությունը մինիմալ է FP լեզվի համար, որում օգտագործվում են երեքից ավելի ատոմներ:

Թեորեմ 3.3.1-ը ապացուցվում է Լեմմա 3.3.1-3.3.9-ի միջոցով: Յուրաքանչյուր լեմմայում դիտարկվում է Φ բազմության մեկ ներդրված հաստատուն՝ $\varphi \in \Phi$, և ցույց է տրվում, որ $\Phi \setminus \{\varphi\}$ բազմությունը լրիվ չէ FP լեզվի համար, որտեղ $C = M \cup (\Phi \setminus \{\varphi\})$:

FP լեզվի համար, որտեղ $C = M \cup (\Phi \setminus \{\varphi\}), \varphi \in \Phi$, $\Phi \setminus \{\varphi\}$ բազմության ոչ լրիվությունը ցույց է տրվում հետևյալ ձևով՝

- Ցույց է տրվում, որ դիտարկվող լեզվում ծրագրավորելի բոլոր S-ֆունկցիաները բավարարում են մեկ կամ մի քանի պայմանի:
- Ցույց է տրվում, որ գոյություն ունի հաշվարկելի S-ֆունկցիա, որը չի բավարարում դիտարկվող պայմաններից որևէ մեկին:

Թեորեմ 3.3.2. FP լեզվի համար, որում օգտագործվում են ճիշտ երեք ատոմներ, տեղի ունի հետևյալը.

1. Ներդրված հաստատունների $C_1 \cup C_2 \setminus \{const\}$ բազմությունը լրիվ է և մինիմալ:

2. Կամայական $\varphi \in C_1 \cup C_2 \setminus \{const\}$ համար ներդրված հաստատունների $C_1 \cup C_2 \setminus \{\varphi\}$ բազմությունը լրիվ չէ:

Թեորեմ 3.3.2-ը ապացուցվում է հետևյալ ձևով: Ցույց է տրվում, որ կամայական հաստատուն ֆունկցիան ներկայացվում է FP լեզվում, որն օգտագործում է երեք ատոմ և $M \cup (\Phi \setminus \{const\})$ ներդրված հաստատունների բազմությունը: Ապացույցի համար օգտագործվում են նաև Լեմմա 3.3.1-3.3.8-ները:

3.4 բաժնում դիտարկվում են FP լեզվի երեք մոդիֆիկացիաներ: Ցույց է տրվում այդ լեզուների ներդրված հաստատունների բազմությունների լրիվությունն ու մինիմալությունը:

$L^{(1)}$-ով նշանակենք FP լեզվի այն մոդիֆիկացիան, որտեղ eq ֆունկցիան որոշված է միայն ատոմների համար և ավելացված է $atom$ ֆունկցիան:

$$eq^{(1)}(m) = \begin{cases} true, & \text{եթե } m = (m_1\ m_2), m_1 = m_2,\ m_1, m_2 \in Atoms \\ false, & \text{եթե } m = (m_1\ m_2), m_1 \neq m_2,\ m_1, m_2 \in Atoms \\ \perp, & \text{մնացած դեպքերում} \end{cases}$$

$$atom(m) = \begin{cases} true, & \text{եթե } m \in Atoms \\ false, & \text{եթե } m \neq \perp, m \notin Atoms \\ \perp, & \text{մնացած դեպքերում} \end{cases}$$

Նշանակենք $\Phi^{(1)} = \{id, hd, tl, apndl, atom, eq^{(1)}\} \cup \{comp, constr, cond, const\}$:

$L^{(2)}$-ով նշանակենք FP լեզվի այն մոդիֆիկացիան, որտեղ տրամաբանական սխալին համապատասխանում է nil ատոմը: Համապատասխանաբար փոփոխված են նաև eq ֆունկցիան և $cond$ ֆունկցիոնալը:

$$eq^{(2)}(m) = \begin{cases} true, \text{եթե } m = (m_1\ m_2),\ m_1 = m_2,\ \ m_1, m_2 \in S \\ nil, \quad \text{եթե } m = (m_1\ m_2),\ m_1 \neq m_2,\ \ m_1, m_2 \in S \\ \perp, \quad \text{մնացած դեպքերում} \end{cases}$$

$cond^{(2)} \in [[M \to M]^3 \to [M \to M]]$ և ցանկացած $p, g, h \in [M \to M]$ ֆունկցիաների համար $cond^{(2)}(p,\ g,\ h) = f \in [M \to M]$, որը որոշվում է հետևյալ կերպ. ցանկացած $m \in M$

$$f(m) = \begin{cases} g(m), & \text{եթե } p(m) \neq nil, p(m) \neq \perp \\ h(m), & \text{եթե } p(m) = nil \\ \perp, & \text{մնացած դեպքերում} \end{cases}$$

Նշանակենք $\Phi^{(2)} = \{id,\ hd,\ tl,\ apndl,\ eq^{(2)}\} \cup \{comp,\ constr,\ cond^{(2)},\ const\}$:

$L^{(3)}$-ով նշանակենք FP լեզվի այն մոդիֆիկացիան, որտեղ eq ֆունկցիան որոշված է միայն ատոմների համար, ավելացված է $atom$ ֆունկցիան, տրամաբանական սխալին համապատասխանում է nil ատոմը: Համապատասխանաբար փոփոխված են նաև $eq, atom$ ֆունկցիաները և $cond$ ֆունկցիոնալը:

$$eq^{(3)}(m) = \begin{cases} true, \text{եթե } m = (m_1\ m_2),\ m_1 = m_2,\ \ m_1, m_2 \in Atoms \\ nil, \quad \text{եթե } m = (m_1\ m_2),\ m_1 \neq m_2,\ \ m_1, m_2 \in Atoms \\ \perp, \quad \text{մնացած դեպքերում} \end{cases}$$

$$atom^{(3)}(m) = \begin{cases} true, & \text{եթե } m \in Atoms \\ nil, & \text{եթե } m \neq \perp,\ m \notin Atoms \\ \perp, & \text{մնացած դեպքերում} \end{cases}$$

$cond^{(3)} \in [[M \to M]^3 \to [M \to M]]$ և ցանկացած $p, g, h \in [M \to M]$ ֆունկցիաների համար $cond^{(3)}(p,\ g,\ h) = f \in [M \to M]$, որը որոշվում է հետևյալ կերպ. ցանկացած $m \in M$

$$f(m) = \begin{cases} g(m), & \text{եթե } p(m) \neq nil, p(m) \neq \perp \\ h(m), & \text{եթե } p(m) = nil \\ \perp, & \text{մնացած դեպքերում} \end{cases}$$

Նշանակենք $\Phi^{(3)} = \{id,\ hd,\ tl,\ apndl,\ atom^{(3)},\ eq^{(3)}\} \cup \{comp,\ constr,\ cond^{(3)},\ const\}$:

Թեորեմ 3.4.1. Ներդրված հաստատունների $\Phi^{(i)}$ բազմությունը լրիվ է $L^{(i)}$ լեզվի համար $(i = 1,2,3)$:

Թեորեմ 3.4.2.

1. Ներդրված հաստատունների $\Phi^{(1)}$ բազմությունը մինիմալ է $L^{(1)}$ լեզվի համար, որում օգտագործվում են երեքից ավելի ատոմներ:
2. Ներդրված հաստատունների $\Phi^{(i)}$ բազմությունը մինիմալ է $L^{(i)}$ լեզվի համար, որում օգտագործվում են երկուսից ավելի ատոմներ $(i = 2,3)$:

Թեորեմ 3.4.3.

1. $L^{(1)}$ լեզվի համար, որում օգտագործվում են երեք ատոմներ, տեղի ունի հետևյալը.
   - Ներդրված հաստատունների $\Phi^{(1)} \setminus \{const\}$ բազմությունը լրիվ է և մինիմալ:
   - Կամայական $\varphi \in \Phi^{(1)} \setminus \{const\}$ համար ներդրված հաստատունների $\Phi^{(1)} \setminus \{\varphi\}$ բազմությունը լրիվ չէ:
2. $L^{(2)}$ լեզվի համար, որում օգտագործվում են երկու ատոմներ, տեղի ունի հետևյալը.
   - Ներդրված հաստատունների $\Phi^{(2)} \setminus \{const\}$ բազմությունը լրիվ է և մինիմալ:
   - Կամայական $\varphi \in \Phi^{(2)} \setminus \{const\}$ համար ներդրված հաստատունների $\Phi^{(2)} \setminus \{\varphi\}$ բազմությունը լրիվ չէ:
3. $L^{(3)}$ լեզվի համար, որում օգտագործվում են երկու ատոմներ, տեղի ունի հետևյալը.
   - Ներդրված հաստատունների $\Phi^{(3)} \setminus \{eq^{(3)},\ const\}$ բազմությունը լրիվ է և մինիմալ:
   - Կամայական $\varphi \in \Phi^{(3)} \setminus \{eq^{(3)},\ const\}$ համար ներդրված հաստատունների $\Phi^{(3)} \setminus \{\varphi\}$ բազմությունը լրիվ չէ:

Այս թեորեմների ապացույցները հետևում են Թեորեմ 3.2.1, 3.3.1 և 3.3.2-ից:

ԱՇԽԱՏԱՆՔԻ ՀԻՄՆԱԿԱՆ ԱՐԴՅՈՒՆՔՆԵՐԸ

Ատենախոսության շրջանակներում կատարված հետազոտությունները բերել են հետևյալ արդյունքների.

1. Ներդրված ֆունկցիաների $\Phi = \{car, cdr, cons, atom, eq, if\_then\_else\}$ բազմության համար ստացվել է հետևյալը.
   - Ցույց է տրվել, որ կամայական $f : B \longrightarrow S\text{-}expressions$ $(B \subset S\text{-}expressions^k, k \geq 1)$ հաշվարկելի ֆունկցիա ծրագրավորելի է ծրագրավորման ֆունկցիոնալ լեզվում, որն օգտագործում է Φ բազմության ֆունկցիաները [1]: Ցույց է տրվել, որ երկուսից ավելի ատոմների դեպքում Φ բազմությունը մինիմալ է [2]:
   - Ցույց է տրվել, որ երկու ատոմների դեպքում կամայական $f : B \longrightarrow S\text{-}expressions$ $(B \subset S\text{-}expressions^k, k \geq 1)$ հաշվարկելի ֆունկցիա ծրագրավորելի է ծրագրավորման ֆունկցիոնալ լեզվում, որն օգտագործում է $\Phi \setminus \{eq\}$ բազմության ֆունկցիաները: Ցույց է տրվել, որ $\Phi \setminus \{eq\}$ բազմությունը մինիմալ է: Ապացուցվել է նաև, որ կամայական $\varphi \in \Phi \setminus \{eq\}$ համար ոչ բոլոր $f : B \longrightarrow S\text{-}expressions$ $(B \subset S\text{-}expressions^k, k \geq 1)$ հաշվարկելի ֆունկցիաներն են ծրագրավորելի լեզվում, որն օգտագործում է $\Phi \setminus \{\varphi\}$ բազմության ֆունկցիաները [2]:
2. Ներդրված հաստատունների $\Phi = \{id, hd, tl, apndl, eq\} \cup \{comp, constr, const, cond\}$ բազմության համար ստացվել է հետևյալը.
   - Ցույց է տրվել, որ կամայական $f : B \longrightarrow S\text{-}expressions$ $(B \subset S\text{-}expressions)$ հաշվարկելի ֆունկցիա ծրագրավորելի է FP լեզվում, որն օգտագործում է Φ բազմության հաստատունները: Ցույց է տրվել, որ երեքից ավելի ատոմների դեպքում Φ բազմությունը մինիմալ է [3]:
   - Ցույց է տրվել, որ երեք ատոմների դեպքում կամայական $f : B \longrightarrow S\text{-}expressions$ $(B \subset S\text{-}expressions)$ հաշվարկելի ֆունկցիա ծրագրավորելի է FP լեզվում, որն օգտագործում է $\Phi \setminus \{const\}$ բազմության հաստատունները: Ցույց է տրվել, որ $\Phi \setminus \{const\}$ բազմությունը մինիմալ է: Ապացուցվել է նաև, որ կամայական $\varphi \in \Phi \setminus \{const\}$ համար ոչ բոլոր $f : B \longrightarrow S\text{-}expressions$ $(B \subset S\text{-}expressions)$ հաշվարկելի ֆունկցիաներն են ծրագրավորելի FP լեզվում, որն օգտագործում է $\Phi \setminus \{\varphi\}$ բազմության հաստատունները [3]:
3. Դիտարկվել են FP լեզվի մոդիֆիկացիաներ: Ցույց է տրվել, որ կամայական $f : B \longrightarrow S\text{-}expressions$ $(B \subset S\text{-}expressions)$ հաշվարկելի ֆունկցիա ծրագրավորելի է այդ լեզուներում: Ցույց է տրվել, որ օգտագործված ներդրված հաստատունների բազմությունները մինիմալ են [3]:

ԱՏԵՆԱԽՈՍՈՒԹՅԱՆ ՇՐՋԱՆԱԿՆԵՐՈՒՄ ՀՐԱՏԱՐԱԿՎԱԾ ԱՇԽԱՏԱՆՔՆԵՐԻ ՑԱՆԿԸ

1. *Martirosyan G. A.* Turing Completeness of Functional Programming Languages with one Set of Built-in Functions. Proceeding of the Conference of Computer Science and Information Technologies (CSIT-2011), Gitutyun Press, Yerevan, 2011, p. 370-373.

2. *Martirosyan G. A.* On Minimality of One Set of Built-in Functions for Functional Programming Languages. Proceedings of Yerevan State University, Physical and Mathematical Sciences, YSU Press, 2012, N2, p. 42-49.

3. *Martirosyan G. A.* On One Complete and Minimal Set of Built-in Constants for Backus FP System", Proceedings of Yerevan State University, Physical and Mathematical Sciences, YSU Press, 2012, N3, p. 44-51.

РЕЗЮМЕ

Мартиросян Геворг Артурович

“О встроенных константах функциональных языков программирования”

В данной работе исследуются свойства функциональных языков программирования. Предметом исследования являются подмножества встроенных констант функциональных языков программирования, а рассматриваемыми проблемами - полнота и минимальность этих подмножеств.

В работе Маккарти определяется язык $Lisp$, который является первым языком функционального программирования. Предметным множеством языка $Lisp$ являестся множество S-выражений ($S\text{-}expressions$) - атомы и составленные из них списки. В работе Маккарти определены элементарные встроенные функции $car, cdr, cons, atom, eq, if\_then\_else$, определенные на множестве $S\text{-}expressions$. На основе фундаментальных понятий языка $Lisp$ были созданы языки $Scheme, Scala, ML, SASL$ и так далее. Множества встроенных функций этих языков содержат вышеуказанные элементарные функции.

В работе Бекуса определен язык функционального программирования FP, предметным множеством которого также является множество $S\text{-}expressions$. В языке FP не используются переменные нулевого порядка (предметные переменные), не используется λ-абстракция и используются переменные первого порядка (функциональные переменные). Во множество встроенных констант языка FP входят одноместные функции $id\ (Identity)$, $hd\ (Head)$, $tl\ (Tail)$, $apndl\ (Append\ left)$, $eq\ (Equals)$ и фунционалы $comp\ (Composition)$, $constr\ (Construction)$, $cond\ (Condition)$, $const\ (Constant)$, а так же другие функции и функционалы.

Представляет интерес выбрать подмножество из встроенных констант данного функционального языка, которое будет полным (в этом языке возможно запрограммировать каждую $f : B \longrightarrow S\text{-}expressions\ (B \subset S\text{-}expressions^k, k \geq 1)$ вычислимую функцию, используя только эти константы) и минимальным (удаление любой встроенной константы из этого множества нарушает полноту).

Целью диссертационной работы является:

1. Из множества встроенных функций языка программирования $Lisp$ выбрать полное и минимальное подмножество.

2. Из множества встроенных функций и функционалов языка программирования FP выбрать полное и минимальное подмножество.

3. Рассмотреть модификации языка программирования FP. Из множеств встроенных функций и функционалов этих языков выбрать полные и минимальные подмножества.

В результате исследований, проведенных в данной работе, были получены следующие основные результаты:

1. Для множества встроенных функций $\Phi = \{car, cdr, cons, atom, eq, if\_then\_else\}$ получено следующее:
   - Доказано, что каждая $f : B \longrightarrow S\text{-}expressions$ $(B \subset S{-}expressions^k, k \geq 1)$ вычисляемая функция программируема в языке функционального программирования, который использует функции множества Φ. Доказано так же, что если число атомов > 2, то множество Φ минимально.
   - Доказано, что если число атомов равно 2, то каждая $f : B \longrightarrow S\text{-}expressions$ $(B \subset S{-}expressions^k, k \geq 1)$ вычисляемая функция программируема в языке функционального программирования, который использует функции множества $\Phi \setminus \{eq\}$. Доказано, что множество $\Phi \setminus \{eq\}$ минимально. Доказано так же, что для каждого $\varphi \in \Phi \setminus \{eq\}$ не все $f : B \longrightarrow S\text{-}expressions$ $(B \subset S{-}expressions^k, k \geq 1)$ вычисляемые функции программируемы в языке, который использует функции множества $\Phi \setminus \{\varphi\}$.

2. Для множества встроенных констант $\Phi = \{id, hd, tl, apndl, eq\} \cup \{comp, constr, const, cond\}$ получено следующее:
   - Доказано, что каждая $f : B \longrightarrow S\text{-}expressions$ $(B \subset S\text{-}expressions)$ вычисляемая функция программируема в языке FP, который использует константы множества Φ. Доказано так же, что если число атомов > 3, то множество Φ минимально.
   - Доказано, что если число атомов равно 3, то каждая $f : B \longrightarrow S\text{-}expressions$ $(B \subset S\text{-}expressions)$ вычисляемая функция программируема в языке FP, который использует константы множества $\Phi \setminus \{const\}$. Доказано, что множество $\Phi \setminus \{const\}$ минимально. Доказано так же, что для каждого $\varphi \in \Phi \setminus \{const\}$ не все $f : B \longrightarrow S\text{-}expressions$ $(B \subset S - expressions)$ вычисляемые функции программируемы в языке FP, который использует функции множества $\Phi \setminus \{\varphi\}$.

3. Рассмотрены модификации языка FP. Доказано, что каждая $f : B \longrightarrow S\text{-}expressions$ $(B \subset S\text{-}expressions)$ вычисляемая функция программируема в этих языках. Доказано так же, что множества используемых встроенных функций минимальны.

ABSTRACT

Gevorg A. Martirosyan

"On Built-in Constants of Functional Programming Languages"

The thesis is devoted to properties of functional programming languages. Objects of the study are subsets of built-in constants of functional programming languages and considered main problems are completeness and minimality of these subsets.

First functional programming language is *Lisp*, which is introduced by McCarthy. The object set of *Lisp* is *S-expressions* (atoms and the lists constructed with those atoms). McCarthy introduced some elementary functions defined on *S-expressions* i.e. *car, cdr, cons, atom, eq, if_then_else*. Using the main ideas of *Lisp* some functional programming languages were created e.g. *Scheme*, *Scala*, *ML*, *SASL* and so on. The set of built-in constants of these languages contain the mentioned functions.

The object set of Backus FP language is also *S-expressions.* FP language doesn't use zero order variables (object variables), doesn't use λ-abstraction, and uses first order variables (functional variables). The set of built-in constants of FP language contains $id\ (Identity)$, $hd\ (Head)$, $tl\ (Tail)$, $apndl\ (Append\ left)$, $eq\ (Equals)$ one place functions and $comp\ (Composition)$, $constr\ (Construction)$, $cond\ (Condition)$, $const\ (Constant)$ functionals, and also contains other functions and functionals.

It shows interest in selecting a complete (any $f : B \longrightarrow S\text{-}expressions, B \subset S\text{-}expressions^k, k \geq 1$ computable function is programmable in such functional programming language which uses only the selected constants) and minimal (if any constant is removed from the set of the selected constants, the set will not be complete) subset from the set of built-in constants of a functional programming language.

The objectives of the thesis are to investigate completeness and minimality of subsets of built-in constants of functional programming languages.

The main topics of the thesis are:

1. Select a complete and minimal subset of built-in functions of functional programming language *Lisp*.
2. Select a complete and minimal subset of built-in functions and functionals of functional programming language FP.
3. Consider modifications of FP language and select complete and minimal subsets of built-in functions and functionals of those languages.

The research carried out in the thesis has produced the following main results:

1. For the set of built-in functions $\Phi = \{car, cdr, cons, atom, eq, if\_then\_else\}$ the following is obtained:
   - It is proved that any $f : B \longrightarrow S\text{-}expressions$ $(B \subset S\text{-}expressions^k, k \geq 1)$ computable function is programmable in such a functional programming

language which uses the functions of the set Φ. Also it is proved that if the number of atoms > 2, then the set Φ is minimal.

- It is proved that if the number of atoms is equal to 2, then any $f : B \longrightarrow S\text{-}expressions$ $(B \subset S\text{-}expressions^k, k \geq 1)$ computable function is programmable in a functional programming language which uses the functions of the set $\Phi \setminus \{eq\}$. It is proved that the set $\Phi \setminus \{eq\}$ is minimal. Also it is proved that for any $\varphi \in \Phi \setminus \{eq\}$ not all $f : B \longrightarrow S\text{-}expressions$ $(B \subset S\text{-}expressions^k, k \geq 1)$ computable functions are programmable in a language which uses the functions of the set $\Phi \setminus \{\varphi\}$.

2. For the set of built-in constants $\Phi = \{id, hd, tl, apndl, eq\} \cup \{comp, constr, const, cond\}$ the following is obtained:

- It is proved that any $f : B \longrightarrow S\text{-}expressions$ $(B \subset S\text{-}expressions)$ computable function is programmable in FP language which uses the constants of the set Φ. Also it is proved that if the number of atoms > 3, then the set Φ is minimal.
- It is proved that if the number of atoms is equal to 3, then any $f : B \longrightarrow S\text{-}expressions$ $(B \subset S\text{-}expressions)$ computable function is programmable in FP language which uses the constants of the set $\Phi \setminus \{const\}$. It is proved that the set $\Phi \setminus \{const\}$ is minimal. Also it is proved that for any $\varphi \in \Phi \setminus \{const\}$ not all $f : B \longrightarrow S\text{-}expressions$ $(B \subset S\text{-}expressions)$ computable functions are programmable in FP language which uses the constants of the set $\Phi \setminus \{\varphi\}$.

3. Are considered modifications of FP languages. It is proved that any $f : B \longrightarrow S\text{-}expressions$ $(B \subset S\text{-}expressions)$ computable function is programmable in those languages. Also it is proved that sets of used built-in constants are minimal.