一种基于ADT的函数语言及其操作语义模型*

梅宏 孙永强

(上海交通大学计算机系,上海 200030) (上海交通大学计算机系,上海 200030)
(北京大学计算机科学与技术系,北京 100871)

摘要 本文介绍一以ADT为主要构件的函数语言,它是作者设计的函数式及面向对象式合成语言的函数部分.进而描述了其操作语义模型:多态λ演算+代数重写系统=多态λ重写系统,并讨论该模型的Church-Rosser性质和强范式性质.

关键词 函数式程序设计,λ演算,重写系统,Church-Rosser性质,强范式性质.

面向对象程序设计对复杂系统的模块化设计、代码复用及可扩展性的支持,函数语言的数学优美性、简单性及引用透明性等优点,使得对这两种风格的语言的合成的研究成为一个热点.在我们的合成语言FOPL的研究中,我们试图在类型理论下将二者的特点平滑地结合起来.显然,抽象数据类型(ADT)和对象有许多相似性,而事实上面向对象的思想正是源于模块封装和ADT.函数语言提供了含ADT的丰富的类型系统,因此在类型理论下的合成是可取的.我们的语言设计主要遵循OOP≈ADTs+Inheritance[1]这一框架,ADT是程序的主要构件.

我们的设计目标之一是使语言仍能用于纯的多态函数程序设计,纯函数程序的构件仍是ADT,也支持继承性,即可在函数级上完成类似OO风格的程序设计.本文旨在讨论FOPL函数部分的操作语义模型及其Church-Rosser性质(CR性)和强范式性质(SN性).因此,为形式描述的简便考虑,我们忽略继承,因为继承仅是代码复用的重要手段,并不能作为OO风格的精髓[2],事实上,凡是继承出现的地方,我们均可在编译时将其去掉而产生没有继承的代码,对继承的忽略无损于我们所讨论的主题.

1 FOPL概述

FOPL函数部分的语法结构类似Miranda[3],但要求显式类型信息.下面给出其语法概述.

* 本文1992-01-14收到,1992-12-30定稿

本项研究受国家自然科学基金及863高科技资助.作者梅宏,31岁,讲师,主要研究领域为新型语言及其支撑环境,软件工程.孙永强,63岁,教授,主要研究领域为新型语言,计算理论.

本文通讯联系人:梅宏,北京 100871,北京大学计算机科学与技术系

Script::=EQ_1…
　　EQ_n
EQ::=Var{P_1…P_n}=exp　　;函数定义
　　|Var{P_1…P_n}=exp_1,$Bexp_1$
　　exp_m,$Bexp_m$　　;函数条件定义
　　|EQ_1…
　　EQ_n
　　where
　　Script−type definition　　;方程组嵌套,嵌套中不允许类型定义
　　|type definition　　;类型定义
　　|type declaration　　;类型说明
type declaration::=exp::texp|texp::Type|texp::Type→Type
type definition ::=ADT name{t}
　　with constructor signature　　;构造子类型说明
　　and operator signature　　;操作子类型说明
　　Script−type definition　　;不允许 ADT 定义
　　TDA
exp::=CONST|Var|exp_1 exp_2|exp texp|(exp)|exp::texp
texp::=tCONST|tVar|$texp_1$→$texp_2$
signature::=CONST::tCONST|Var::$texp_1$→$texp_2$|$signature_1$;$signature_2$
P::=Var|tVar|c P_1…P_n(n≥0)　　;类型模式,c 为构造子
CONST::=ADT 定义中出现于 signature 中的标识符
tCONST::=所有定义的 ADT 名

2　操作语义模型

在本节中我们首先给出模型的定义,然后简要描述从 FOPL 到该模型的解释.

定义 1. 多态 λ 重写系统 λπR=多态 λ 演算 λπ+代数重写系统⟨S,$\sum$,R⟩. 其中 S 为类型集合,$\sum$为标记(signature)集合,$\sum$中元素形为 c::s_1→…→s_n→s,n≥0,s_i∈S,s∈S. 如 n=0,c 为类型 s 的常量,否则 c 为代数操作符.

·常量类型仅为 S 中类型,称为基类型

σ::=s|t|σ→σ|πt.σ,s∈S,t 是类型变量

·$\sum$中符号作为常量,操作符为高阶常量

M::=A|x|M_1 M_2|λx::σ.M|Mσ|Λt.M

A::=c|fA, c,f∈$\sum$;c 为常量,f 为操作符

·重写规则

(1)β 规 则:(λx::σ.X)Y→$^{\beta}$X[Y/x]

(2)η 规 则:λx::σ.Xx→$^{\eta}$X,x∉FV(X)

(3)Tβ规则:(Λt.X)σ→$^{T\beta}$X[σ/t]

(4)Tη规则:Λt.Xt→$^{T\eta}$X,t∉FV(X)

(5)R 规则:r≡A→B::s,要求 A,B 有相同类型 s∈S,且 FV(B)≤FV(A),A 不是变量. 在 λπR 项上的 R 重写可将规则中出现的变量(模式)用相同基类型的项例化.

• λπR 项的归约

从重写规则 r,M→'N iff N 从 M 得到,通过将 M 中的某子项 X 重写成同类型项 Y,即 X→'Y.

• 类型检查

△表示类型赋值,这是一个有限域上的部分函数,将变量映射到类型表达式上. 类型判断形为△⊢M::σ. 进行类型检查的形式系统如下:

$\triangle \vdash c :: \sum'(c), c \in dom(\sum')$;$\sum$视为映射

$\triangle \vdash x :: \triangle(x), x \in dom(\triangle)$

$\dfrac{\triangle \vdash M :: \sigma \to \tau \quad \triangle \vdash N :: \sigma}{\triangle \vdash MN :: \tau}$;作用项的类型

$\dfrac{\triangle; x :: \sigma \vdash M :: \tau}{\triangle \vdash \lambda x :: \sigma. M :: \sigma \to \tau}$;λ 抽象项的类型

$\dfrac{\triangle \vdash M :: \pi t. \sigma}{\triangle \vdash M\tau :: \sigma[\tau/t]}$;类型作用项的类型

$\dfrac{\triangle \vdash M :: \sigma}{\triangle \vdash \Lambda t. M :: \pi t. \sigma}$, t∉FV(ran(△));多态项的类型

$\dfrac{\triangle \vdash f :: s_1 \to \cdots \to s_n \to s, A_1 :: s_1, \cdots, A_n :: s_n}{\triangle \vdash fA_1 \cdots A_n :: s}$, $f \in \sum', s_i \in S, s \in S$;代数项的类型

下面我们简要描述从 FOPL 到模型的翻译,一些具体的细节忽略. 这里采用的翻译技术类似文献[4].

1. 表达式的翻译

TE【CONST】==CONST, TE【Var】== Var

TE【exp_1 exp_2】==TE【exp_1】TE【exp_2】

TE【exp texp】==TE【exp】TE【texp】

TE【exp::texp】==TE【exp】::TE【texp】

TE【tCONST】==tCONST, TE【tVar】==tVar

TE【$texp_1 \to texp_2$】==TE【$texp_1$】→TE【$texp_2$】, TE【Type】==space

2. 一般方程组的翻译.

(1)全称类型定义的翻译

TD【tVar::Type→⋯→Type

tVar $t_1 \cdots t_n$=texp】 ==tVar=$\pi t_1. \cdots. \pi t_n$. TE【exp】

(2)函数定义的翻译

TD【Var::$t_1 \to \cdots \to t_n \to t$ Var=$\overline{\lambda v}$. (((λ*P'$_{11}$. ⋯λ*P'$_{1n}$. TE【exp_1】)$\bar{v}$)

Var $P_{11} \cdots P_{1n}$=exp_1⋯ ==‖⋯⋯

Var $P_{m1} \cdots P_{mn}$=exp_m】 ‖((λ*P'$_{m1}$. ⋯. λ*P'$_{mn}$. TE【exp_m】)$\bar{v}$)

‖ERROR)

其中 $\bar{v}$ 表示 $v_1, \cdots, v_n$, P'$_{ij}$=TP【P_{ij}】::TE【t_j】, 1≤i≤m, 1≤j≤n, λ* 或者为 Λ, 如 t_i 为 Type, 或者为 λ, 如 t_i::Type

TD【Var::$t_1 \to \cdots \to t_n \to t$ Var=$\overline{\lambda v}$. (((λ*P'$_1$⋯λ*P'$_n$.

$$\begin{aligned}&\text{Var } P_1\cdots P_n=exp_1,Bexp_1\cdots && == && \text{if TE}\llbracket Bexp_1\rrbracket\text{then TE}\llbracket exp_1\rrbracket\text{else}\\ &\qquad exp_m,Bexp_m && && \text{if TE}\llbracket Bexp_m\rrbracket\text{then TE}\llbracket exp_m\rrbracket\text{else}\\ &&&& \text{Fail})\bar{v})\\ &&&& \|\text{ERROR})\end{aligned}$$

$$\begin{aligned}&\text{TD}\llbracket EQ_1\cdots && \text{letrec}\\ &\quad EQ_n && \quad \text{TD}\llbracket\text{Script}-\text{type definition}\rrbracket\\ &\quad \text{where} && == \quad \text{in}\\ &\quad \text{Script}-\text{type definition}\rrbracket && \quad \text{TD}\llbracket EQ_1\cdots EQ_n\rrbracket\end{aligned}$$

在上面翻译中,我们引入了‖,letrec…in,模式抽象,Fail,ERROR 等中间语法结构,将它们进一步翻译到 λπR 是需要的且容易的[4].

3. ADT 定义的翻译

$$\begin{aligned}&\text{TD}\llbracket\text{ADT name}\{t\}\\ &\quad \text{with signature}_1\\ &\quad \text{and signature}_2 && == && \text{name}\{t\}=\langle S,\sum,R\rangle\\ &\quad \text{Script}-\text{type definition}\\ &\quad \text{TDA}\rrbracket\end{aligned}$$

其中 S 为标记中涉及的基类型之集,含有 name{t},其类型可用 λπ 表示;$\sum=\text{signature}_1+\text{signature}_2$. R 由 TDR 过程对方程组翻译而得.

$$\begin{aligned}&\text{TDR}\llbracket\text{Var } P_{11}\cdots P_{1n}=exp_1\cdots && && R=\{\text{Var TP}\llbracket P_{11}\rrbracket\cdots\text{TP}\llbracket P_{1n}\rrbracket\to\text{TE}\llbracket exp_1\rrbracket,\\ &\quad \text{Var } P_{m1}\cdots P_{mn}=exp_m\rrbracket && == && \quad \text{Var TP}\llbracket P_{m1}\rrbracket\cdots\text{TP}\llbracket P_{mn}\rrbracket\to\text{TE}\llbracket exp_m\rrbracket\}\\ &\text{TDR}\llbracket\text{Var } P_1\cdots P_n=exp_1,Bexp_1\cdots && && R=\{\text{Var TP}\llbracket P_1\rrbracket\cdots\text{TP}\llbracket P_n\rrbracket\to\\ &\qquad exp_m,Bexp_m\rrbracket && == && \quad \text{if TE}\llbracket Bexp_1\rrbracket\text{then TE}\llbracket exp_1\rrbracket\text{else}\\ &&&& && \quad \text{if TE}\llbracket Bexp_m\rrbracket\text{then TE}\llbracket exp_m\rrbracket\text{else}\\ &&&& && \quad \text{Fail}\}\end{aligned}$$

$$\begin{aligned}&\text{TDR}\llbracket EQ_1\cdots && R=\text{TDR}\llbracket\text{Script}-\text{type defintion}\rrbracket\\ &\quad EQ_n && == \quad +\\ &\quad \text{where} && \quad \text{TDR}\llbracket EQ_1\cdots EQ_n\rrbracket\\ &\quad \text{Script}-\text{type defintion}\rrbracket\end{aligned}$$

4. 模式的翻译

因方程定义中涉及模式的使用,故必须保持模式的翻译.

$$\text{TP}\llbracket\text{Var}\rrbracket==\text{Var},\ \text{TP}\llbracket t\text{Var}\rrbracket==t\text{Var}$$

$$\text{TP}\llbracket c\ P_1\cdots P_n\rrbracket==c\ \text{TP}\llbracket P_1\rrbracket\cdots\text{TP}\llbracket P_n\rrbracket$$

3　关于 CR 性及 SN 性

本节我们分析 λπR 的 CR 性及 SN 性,首先给出 λπRCR 性的描述,为简便计,我们将 β 和 Tβ 均称为 β,η 和 Tη 均称为 η,事实上,FOPL 将类型作为一阶参数传送,我们可将其视为值. 下面定理1的证明源于文献[5].

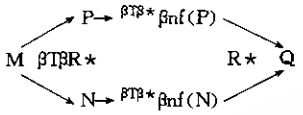
引理1. 设 X,Y 是 λπR 项,(1)设 $r\in R$,如 $X\to^{r}Y$,则 $\beta nf(X)\to^{r*}\beta nf(Y)$. (2)如 $X\to^{\beta R*}Y$,则 $\beta nf(X)\to^{R*}\beta nf(Y)$,这里 * 表示多步归约,βnf(X)表示 X 的不可再进行 β,Tβ 归约的

范式.(证明略)

引理2. (1)如M是β范式,$M\rightarrow^{R*}N$,则N也是β范式.(2)如R归约在代数项上是CR的,则它在λπR的项上也是CR的.(证明略)

定理1. 如R对代数项具有CR性,则β,Tβ,R对λπR项也具有CR性.

证明:设有$M\rightarrow^{\beta T\beta R*}N$,$M\rightarrow^{\beta T\beta R*}P$,由引理1有,$\beta nf(N)\leftarrow^{R*}\beta nf(M)\rightarrow^{R*}\beta nf(P)$,由引理2,存在Q,使得$\beta nf(N)\rightarrow^{R*}Q\leftarrow^{R*}\beta nf(P)$,从而有:



得证

需要指出的是,η+R并不具有CR性,如有$fx\rightarrow^{R}a$,则有$f\leftarrow^{\eta}\lambda x.\,fx\rightarrow^{R}\lambda x.\,a$,但是,规则η并不出现于FOPL的计算解释中.因此,η归约对CR性的破坏并不会影响语言的并行计值,只是在程序推理中,作为证明系统中的方程公理而起作用.事实上,我们已知:

定理. 一个λπR项有βη范式iff它有β范式.[6]

下面我们讨论λπR的SN性,用类似Girard的可归约候选者方法(candidates of reducibility).[7]

定义2. SN——候选者集合

SN候选者集合族是一个族$F=(F_\sigma)_{\sigma\in Type}$,σ是类型,其中每个$F_\sigma$由所有具有SN性的项的集合C组成.令SNσ是类型σ的满足SN性的项的集合,则$C\leqslant SN\sigma\in F_\sigma$.

引理3. 设$C\leqslant SN\sigma$是σ类型的满足SN性的项集,则有下列性质成立,这里要求R对代数项是SN的.

C1. 如x是变量,$T_1,\cdots,T_k(k\geqslant 0)$或者是符合类型的SN项,或者是类型,且$xT_1\cdots T_k::\sigma$,则$xT_1\cdots T_k\in C$.

C2. 如$f\in\sum$是常量符号,$T_1,\cdots,T_k(k\geqslant 0)$同上,$fT_1\cdots T_k::\sigma$,则$fT_1\cdots T_k\in C$.

C3. M,N是符合类型的SN项,$T_1,\cdots,T_k(k\geqslant 0)$如上,在M中$x::\sigma$且$M[N/x]T_1\cdots T_k\in C$,则$(\lambda x::\sigma.\,M)NT_1\cdots T_k\in C$.

C4. 如M是SN项,$T_1,\cdots,T_k(k\geqslant 0)$如上,τ是类型,且$M[\tau/t]T_1\cdots T_k\in C$,则$(\Lambda t.\,M)\tau T_1\cdots T_k\in C$.

引理4. 对SN性,有下列性质成立:

B1. 如M在△下符合类型,Mx在△,$x::\sigma$下符合类型且是SN的.则M满足SN性.

B2. t为类型变量,如Mt是SN的,则M有SN性.

B3. σ为任意类型,SNσ本身具有SN性,故$SN\sigma\in F_\sigma$.

定义3. 定义环境$\rho:\nu\rightarrow Type\times F_\sigma$,对每个类型变量$t\in\nu$附上一个对$(\tau,C)$,$\tau\in Type$,$C\in F_\tau$.

$【】_\rho$对类型σ产生一个σ类型项集.

$【s】_\rho==SNs,s\in S,【t】_\rho==snd(\rho(t))$

$【\sigma\rightarrow\tau】_\rho==\{M\mid M::(\sigma\rightarrow\tau)_\rho,\forall\ N\in【\sigma】_\rho\Rightarrow MN\in【\tau】_\rho\}$

$\llbracket \pi t.\sigma \rrbracket_\rho = = \{M \mid M :: (\pi t.\sigma)_\rho, \forall\ \tau \in Type, \forall\ C \in F_\tau, M\tau \in \llbracket \sigma \rrbracket_{\rho[\langle \tau, C\rangle/t]}\}$

其中,$\rho[\langle \tau, C\rangle/t](t') = \langle \tau, C\rangle$,if $t' = t$,或为 $\rho(t')$,if $t' \neq t$. 定义中$(\sigma)_\rho$表示在 ρ 下对 σ 求类型,$(s)_\rho = s$,$(t)_\rho = fst(\rho(t))$.

引理5. 对类型 σ 和环境 ρ,$\llbracket \sigma \rrbracket_\rho \in F_{(\sigma)\rho}$,则$\llbracket \sigma \rrbracket_\rho$是类型$(\sigma)_\rho$的 SN 候选集.

证明:对 σ 进行结构归纳.

(1)如 σ 为变量或常量,因 $\rho(t)$是一个 SN 候选者集,由 B3,SNσ 本身也是 SN 候选者集.

(2)设 M 在△下类型检查,且 $M \in \llbracket \sigma \to \tau \rrbracket_\rho$,如 $x \notin dom(\triangle)$,因 x 在△,$x :: \sigma$ 下附类型为 $x :: \sigma$,故由 C1,有 $x :: \sigma \in \llbracket \sigma \rrbracket_\rho$,再根据$\llbracket \sigma \to \tau \rrbracket_\rho$的定义,我们有 $Mx \in \llbracket \tau \rrbracket\rho$,由归纳假设,Mx 是 SN 的,又由 B1,M 满足 SN 性.

(3)设 $M \in \llbracket \pi t.\sigma \rrbracket_\rho$,由$\llbracket \pi t.\sigma \rrbracket_\rho$的定义,我们有 $Mt \in \llbracket \sigma \rrbracket\rho[\langle t, SNt\rangle/t]$,由归纳假设,Mt 有 SN 性,又由 B2,M 有 SN 性.

由此可知,$\llbracket \sigma \rrbracket_\rho$中每个项具有 SN 性.

引理6. 给定任意环境 ρ 和项替代 φ,如 M 是类型检查为 σ 的项,则$(M)_{\rho\cup\varphi} \in \llbracket \sigma \rrbracket_\rho$,这里 ρ 进行类型变量替代,φ 进行项变量替代.

证明:对$\triangle \vdash M :: \sigma$的推导深度进行归纳.这里仅给出两种抽象情形的证明.

(1)λ 抽象:$\dfrac{\triangle, x :: \sigma \vdash M :: \tau}{\triangle \vdash (\lambda x :: \sigma.\ M) :: \sigma \to \tau}$

$\forall\ y \in dom(\triangle)$,有 $y :: (\triangle(y))_\rho$,设 N 是任意类型为 σ 的项且满足 $N \in \llbracket \sigma \rrbracket_\rho$,由归纳假设,对△,$x :: \sigma \vdash M :: \tau$,有$(M)_{\rho\cup\varphi[N/x]} \in \llbracket \tau \rrbracket_\rho$. 如令 $N \equiv x$,则$(M)_{\rho\cup\varphi} \in \llbracket \tau \rrbracket_\rho$. 由引理5,N 和$(M)_{\rho\cup\varphi}$均具有 SN 性. 又因$(M)_{\rho\cup\varphi[N/x]} = (M)_{\rho\cup\varphi}[N/x] \in \llbracket \sigma \rrbracket_\rho$,由 C3,我们有:$(\lambda x :: (\sigma)_\rho.\ (M)_{\rho\cup\varphi})N \in \llbracket \tau \rrbracket_\rho$,从而$((\lambda x :: \sigma.\ M)_{\rho\cup\varphi})N \in \llbracket \tau \rrbracket_\rho$,由定义3,我们有$(\lambda x :: \sigma.\ M)_{\rho\cup\varphi} \in \llbracket \sigma \to \tau \rrbracket_\rho$.

(2)类型抽象:$\dfrac{\triangle \vdash M :: \sigma}{\triangle \vdash \wedge t.\ M :: \pi t.\sigma}$;　$t \notin FV(ran(\triangle))$

由归纳假设有$(M)_{\rho[\langle \tau, SN\tau\rangle/t]\cup\varphi} \in \llbracket \sigma \rrbracket_{\rho[\langle \tau, SN\tau\rangle/t]}$,$\forall\ \tau \in Type$,令 $\tau \equiv t$,则有$(M)_{\rho\cup\varphi} \in \llbracket \sigma \rrbracket_\rho$,由引理5,$(M)_{\rho\cup\varphi}$具有 SN 性. 因$(M)_{\rho[\langle \tau, SN\tau\rangle/t]\cup\varphi} = (M)_{\rho\cup\varphi}[\tau/t] \in \llbracket \sigma \rrbracket_{\rho[\langle \tau, SN\tau\rangle/t]} = \llbracket \sigma \rrbracket_\rho[SNt/t]$,由 C4,有$(\wedge t.\ (M)_{\rho\cup\varphi})\tau \in \llbracket \sigma \rrbracket_\rho[SN\tau/t]$,从而$((\wedge t.\ M)_{\rho\cup\varphi})\tau \in \llbracket \sigma \rrbracket_\rho[SN\tau/t]$,由定义5有:$(\wedge t.\ M)_{\rho\cup\varphi} \in \llbracket \pi t.\sigma \rrbracket_\rho$.

定理2. 如 R 对代数项有 SN 性,则 $\lambda\pi R$ 中每个类型检查项 M 均满足 SN 性.

证明:从引理6,令 $\rho(t) = \langle t, SNt\rangle$,$\varphi(x) = x$,即有 $M \in \llbracket \sigma \rrbracket_\rho$,由引理5,M 具有 SN 性.

定理3. 如 R 对代数项是 SN 的,则 $\lambda\pi R$ 对所有项都是 SN 的.

4　结束语

本文描述了一种基于 ADT 的函数语言及其操作语义模型,并讨论了该模型的 CR 性及 SN 性. 本文讨论的模型限于 $\lambda\pi$ 加上由代数类型规范产生的代数重写系统所得到的多态重写系统,目的是为了解决语言中没有原始类型所带来的证明问题以及扩展语言的表达能力. 关于两个一般重写系统之和及和一般代数重写系统之和的研究已有许多,如文献[8,9],本文的写作很多得益于这些研究成果.

我们已在Sun386i工作站上实现了FOPL的原型系统.并达到了预期设计目标.为了使FOPL实用化,进一步的工作是必需的.

参考文献

1 Danforth S, Tomlinson C. Type theories and object—oriented programming. ACM Computing Surveys, 1988,**20**(1).

2 America P *et al*. Denotational semantics of a parallel object—oriented language. Information and Computation 83, 1989.

3 Turner D A. Miranda: a non—strict functional language with polymorphic type. LNCS, 1985,**201**.

4 Jones S L P. The implementation of functional programming language. Prentice Hall. 1987.

5 Breazu—Tannen V. Combining algebra and higher—order types. Symposium on Logic in Computer Science. Edinburgh, 1988,**7**.

6 Hindley J R, Seldin J R. Introduction to combinators and λ—calculus. Chapter 7. Cambridge University Press. 1986.

7 Girard J Y *et al*. Proofs and types. Cambridge University Press, 1989.

8 Toyama Y. On the church—rosser property for the direct sum of term rewriting systems. Journal of the ACM, 1987,**34**(1).

9 Breazu—Tannen V, Gallier J. Polymorphic rewriting conserves algebraic strong normalization. Theoretical Computer Science, 1991,**83**(1).

A ADT—BASED FUNCTIONAL LANGUAGE AND ITS OPERATIONAL SEMANTICS

Mei Hong

(*Department of Computer Science, Shanghai Jiaotong University, Shanghai* 200030)

(*Department of Computer Sciences and Technology, Peking University, Beijing* 100871)

Sun Yongqiang

(*Department of Computer Science, Shanghai Jiaotong University, Shanghai* 200030)

Abstract This paper presents a ADT—based functional language which is the functional component of a hybrid language that the authors designed supporting both the functional programming and object—oriented programming, and describes its operational semantics model: polymorphic λ—calculus+algebraic rewriting system=polymorphic λ—rewriting system. Moreover, the Church—Rosser property and strong normalization property of this model are discussed.

Key words Functional programming, λ—calculus, rewriting system, church—rosser property, strong normalization property.