ТЕОРИЯ ВЫЧИСЛЕНИЙ И ЯЗЫКИ СПЕЦИФИКАЦИЙ
(Вычислительные системы)
1995 год Выпуск 152

УДК 510.62:519.68

ОБЪЕКТНО-ОРИЕНТИРОВАННЫЙ ВАРИАНТ ЯЗЫКА Σ-СПЕЦИФИКАЦИЙ*)

В.Ш.Гумиров

Введение

Объектно-ориентированный стиль программирования, явившийся, на наш взгляд, прагматическим воплощением идеи абстрактных типов данных, становится в настоящее время идеологической основой в программной индустрии. Классические работы по объектной ориентации [5,6] выделяют следующие основные признаки использования ее методов в некотором языке программирования: 1) абстрагирование; 2) ограничение доступа; 3) иерархическое наследование; 4) полиморфизм; 5) устойчивость; 6) параллелизм; 7) типизация. Последние три пункта считаются не обязательными и реализуются в основном в языках объектно-ориентированных баз данных или на многозадачных системах.

Базовым понятием в объектном подходе являются понятия объекта и класса объектов. У каждого класса обычно определяется некоторый набор характеристик (данных) и методов (функций - членов класса), с помощью которых можно манипулировать этим классом. Оно в чем-то схоже с понятием абстрактного типа данных. Тогда объект - это представитель класса (элемент абстракт-

*) Работа частично поддержана РФФИ (грант № 093-01-01506).

ного типа данных). Создавать новые классы можно наследуя уже существующие (иерархическое наследование). Под наследованием понимается наследование свойств (методов и характеристик) базовых классов. При этом часть методов и характеристик класса можно спрятать внутри реализации класса, так что обратиться к этим характеристикам и методам можно будет только из методов данного или производных (наследниках) от него классов (ограничение доступа). Помимо этого можно переопределить методы базовых классов в производных от них (полиморфизм). Под абстрагированием понимается создание абстрактных классов, имеющих нереализованные методы. Абстрактные классы используют в качестве базовых классов для других, имеющих тот же набор методов, но уже переопределенных. Под устойчивостью понимают продолжительное время существования объектов в системе.Традиционные объектные языки программирования, такие как C++ или Smalltalk,позволяют объектам существовать только во время выполнения программы (runtime). Устойчивость в настоящее время реализована в языках объектных СУБД, в IBM System Object Model [8] (технология объектно-ориентированного программирования в OS/2 и новых проектах Workplase OS и Taligent),а также в Common Object Request Broker Architecture [7]. Естественность идей объектного подхода привлекает к нему внимание, что способствует расширению промышленного применения объектно-ориентированных языков программирования.

Наиболее используемые объектно-ориентированные языки программирования, такие как C++, Smalltalk, основываются на процедурном подходе. Именно это позволяет создавать с помощью этих языков эффективно работающие программные продукты, однако идет в ущерб выразительной мощи по сравнению, например, с той, которой обладают декларативные языки. С другой стороны, языки абстрактных типов данных, такие как OBJ, обладающие низкой производительностью, по этой причине практически не применяются в промышленном программировании.

Подход, начала которого излагаются в данной работе, основан на идее совмещения выразительной мощи языка семантического программирования (Σ-языка) [1,2] и идеологической простоты объектного подхода к пооектированию и построению программ. Ограничения, накладываемые на синтаксис языка Σ-схем [3] (допускаются только ограниченные кванторы по классам объектов) в свете последних результатов А.В.Манциводы [4], могут позволить описать по-настоящему *эффективную* операционную семантику для данного варианта языка Σ-спецификаций. С другой стороны, тот факт, что синтаксис последнего основан на синтаксисе вводимого здесь варианта программной логики (RDL), обладающей, на наш взгляд, весьма интересными свойствами, может позволить включить эти свойства в будущую операционную семантику.

§1. Ресурсная динамическая логика (RDL)

1. Введение в RDL. Будем рассматривать множество *базовых сортов* $Sort_0$, предполагая, что $\underline{bool} \in Sort_0$; σ_0 - базовая сигнатура с множеством сортов $Sort_0$.

Множество *базовых типов* Type задается следующими условиями:

1) $Sort_0 \subseteq Type$;

2) если $s_1, s_2 \in Type$, то $(s_1 x s_2) \in Type$;

3) считается, что операция "x" ассоциативна.

Кроме того, на типах будем рассматривать частичный порядок $\leq$, удовлетворяющий следующему условию: если $t_1, t_2 \in Type$ и $t = (t_1 x\ t_2)$, то $t_1 \leq t$ и $t_2 \leq t$.

Пусть σ - многосортная сигнатура с множеством сортов Type и множеством *переменных* V. Каждому функциональному (предикатному) символу F соответствует его тип type(F) (для преди -

катного символа P $type(P) = \underline{bool}$), а также тип его аргумента*) $arg(F)$. Каждая переменная $v \in V$ тоже имеет тип $type(v)$.

Через M будем обозначать многосортную модель сигнатуры σ. Пусть $a \in |M|$, тогда через $type(a)$ будем обозначать тип этого элемента модели M.

Пусть Res – множество ресурсных символов. Каждому ресурсному символу R соответствует его тип $type(R) \in Type$. Кроме того, на множестве Res задан частичный порядок $\sqsubseteq$, удовлетворяющий следующему условию: если $R_1, R_2 \in Res$ и $type(R_1) \leq type(R_2)$, то $R_1 \sqsubseteq R_2$. Пусть $R \in Res$, тогда обозначим $R_{\sqsubseteq} = \{R_o \in Res | R_o \sqsubseteq R\}$ и $Res_{/\sqsubseteq} = \{R_{\sqsubseteq} | R \in Res\}$. Будем также считать, что $type(R_{\sqsubseteq}) = \sup\{type(R_o) | R_o \in R_{\sqsubseteq}\}$.

Означиванием переменных из V будем называть такое отображение $s: V \to |M|$, что $(\forall v \in V)(type(s(v)) \leq type(v))$.

Через $S(M)$ будем обозначать множество всех таких означиваний.

Введем еще некоторые обозначения. Пусть A - некоторое множество, тогда $FL(A) = \{\tau: A \to N | dom(\tau)$ - конечное множество$\}$, где $dom(\tau) = \tau^{-1}(N \setminus \{0\})$. Определим также операции $\oplus$: $FL(A) \times A \to FL(A)$ и $\ominus$: $FL(A) \times A \to FL(A)$ следующим образом: для $y \in A \setminus \{x\}$ $(\tau \oplus x)(y) = \tau(y)$ и $(\tau \ominus x)(y) = \tau(y)$, для $y = x$ $(\tau \oplus x)(y) = \tau(y)+1$ и $(\tau \ominus x)(y) = \tau(y)-1$. Заметим, что любой $\tau \in FL(A)$ можно представить в виде $\tau = \Sigma_{x \in dom(\tau)} \tau(x) \times x$.

Положим $\mathcal{P}_{res} = \{ P: Res_{/\sqsubseteq} \times N \to \cup_{R_{\sqsubseteq} \in Res_{/\sqsubseteq}} FL(R_{\sqsubseteq})\}$. Множество *состояний* - это множество $ST = S(M) \times \mathcal{P}_{res} \times |M| \times N$. То есть каждое состояние $a \in ST$ мы будем представлять четверкой: $\langle s(a), pr(a), rv(a), ctxt(a) \rangle$. При этом через $a[s = s_1]$ будем кратко обозначать состояние $\langle s_1, pr(a), rv(a), ctxt(a) \rangle$.

*) Поскольку операция "x" на типах ассоциативна, то можно считать все предикаты и функции унарными.

Аналогично для остальных компонент.

2. Термы RDL. Здесь мы определим термы RDL - TR(RDL) или просто TR. Помимо этого для каждого $p \in TR$ определим type(p) (тип терма p), $par(p) \subseteq V$ (параметры терма p) и $sv(p) \subseteq V$ (присвоенные переменные терма p). Определяем по индукции.

1. Каждая переменная $v \in V$ есть терм RDL, тип которого совпадает с type(v), $par(v) = \{v\}$, $sv(v) = \emptyset$.

2. Пусть F - функциональный символ сигнатуры σ, $p \in TR$, $type(p) = arg(F)$. Тогда $F(p) \in TR$, $type(F(p)) = type(F)$, $par(F(p)) = par(p)$, $sv(F(p)) = sv(p)$.

3. Пусть $p,q \in TR$ и выполнено условие: $par(p) \cap sv(q) = \emptyset \wedge par(q) \cap sv(p) = \emptyset \wedge sv(p) \cap sv(q) = \emptyset$. Тогда $t = (p|q) \in TR$, при этом $type(t) = (type(p) \times type(q))$, $par(t) = par(p) \cup par(q)$, $sv(t) = sv(p) \cup sv(q)$.

4. Пусть $v \in V$, $t \in TR$, при этом $type(v) \leq type(t)$. Тогда $(v := t) \in TR$, $type(v := t) = type(v)$, $par(v := t) = par(t)$, $sv(v := t) = sv(t) \cup \{v\}$.

5. Пусть $p,q \in TR$, тогда $(p;q) \in TR$, $type(p;\ q) = type(q)$, $par(p;\ q) = par(p) \cup par(q) \setminus sv(p))$.

6. Пусть $t \in TR$, тогда $\{t\} \in TR$, $type(\{t\}) = type(t)$, $par(\{t\}) = par(t)$, $sv(\{t\}) = \emptyset$.

7. Пусть Φ - формула сигнатуры σ, тогда $[\Phi] \in TR$, $type([\Phi]) = $ bool, $par([\Phi]) = \{v \in V | v$ - свободная переменная формулы $\Phi\}$, $sv([\Phi]) = \emptyset$.

8. Пусть $\tilde{R} \in Res_{/\subseteq}$, B - формула сигнатуры $\sigma \cup \{rv\}$, тогда $t = [B \circ \tilde{R}] \in TR$, $type(t) = type(\tilde{R})$, $sv(t) = \emptyset$, $par(t) = par(\tilde{R})$.

Определением ресурсного символа $R \in Res$ будем называть конструкцию вида: Rdefbody(R), где $body(R) \in TR$ и $type(R) = type(body(R))$.

3. Формулы RDL. F-множество формул RDL определяется следующим образом:

1) true, false $\in F$;

2) формулы сигнатуры σ являются формулами RDL;

3) пусть $\Phi, \Psi \in F$, тогда $\Phi \wedge \Psi$, $\Phi \vee \Psi, \neg\Phi \in F$;

4) пусть $t \in TR$, $\Phi \in F$, тогда $\langle t \rangle \Phi \in F$.

Пусть α - набор определений ресурсных символов. Означивание термов RDL - это отображение $m_\alpha : TR \to (ST \to FL(ST))$. Означивание формул RDL - это отображение $\pi_\alpha F \to 2^{ST}$. Эти отображения определяются ниже.

4. Модель RDL. Моделью RDL будем называть тройку $N = \langle M, m_\alpha, \pi_\alpha \rangle$. Пусть $\Phi \in F$, тогда будем говорить, что формула Φ истинна на модели RDL N в состоянии $a \in ST$, и писать $N \models_\alpha \Phi$ тогда и только тогда, когда $a \in \pi_\alpha(\Phi)$.

Теперь определим отображение π_α.

1. Если Φ - формула сигнатуры σ, то $\pi_\alpha(\Phi) = \{a \in ST | M \triangleright \triangleright a(\Phi)\}$.

2. $\pi_\alpha(\underline{true}) = ST$, $\pi_\alpha(\underline{false}) = \emptyset$.

3. $\pi_\alpha(\Phi \forall \Psi) = \pi_\alpha(\Phi) \cup \pi_\alpha(\Psi)$.

4. $\pi_\alpha(\Phi \wedge \Psi) = \pi_\alpha(\Phi) \cap \pi_\alpha(\Psi)$.

5. $\pi_\alpha(\Phi) = ST \setminus \pi_\alpha(\Phi)$.

6. $\pi_\alpha(\langle t \rangle \Phi) = \{a \in ST | (\exists a')(a' \in \pi_\alpha(\Phi) \ \&\ a' \in m_\alpha(t,a))\}$.

Теперь определим отображение $m_\alpha : TR \to (ST \to FL(ST))$ (предполагается, что $a,b,\ldots \in ST$, $x,y \in V$, $t,p,q,\ldots \in TR$):

1) $m_\alpha(x,a) = a[rv = s(a)(x)]$,

2) $m_\alpha(F(t),a) = \Sigma_{b \in m(t,a)} m_\alpha(t,a)(b) \times b[rv = F^M(rv(b))]$;

3) $m_\alpha((p;q),a) = \Sigma_{b \in m(p,a)} m_\alpha(p,a)(b) \times$
$\times (\Sigma_{c \in m(q,b)} m_\alpha(q,b)(c) \times c)$;

4) $m_\alpha(p|q,a) = \sum\limits_{b \in m(p,a) \& c \in m(q,a)} (m_\alpha(p,a)(b) \times$
$\times b[rv = \langle rv(b), rv(c) \rangle] + m_\alpha(q,a)(c) \times$
$\times c[rv = \langle rv(b), rv(c) \rangle])$;

5) $m_\alpha((x:= t),a) = \Sigma_{b\in m(t,a)} m_\alpha(t,a)(b) \times$

$$\times\, b[s = s(b)[x = rv(b)]];$$

6) $m_\alpha(\{t\},a) = \Sigma_{b\in m(t,a)} m_\alpha(t,a)(b) \times a[rv = rv(b)]$;

7) если Φ - формула сигнатуры σ, то $m_\alpha(\Phi,a) = a$, если $M \rhd a(\Phi)$, и $m_\alpha(\Phi,a) = 0$, если $M \not\rhd a(\Phi)$;

8) $m_\alpha([B \circ \tilde{R}]) =$

$$= \Sigma_{R\in\tilde{R}}(\Sigma_{b\in m_\alpha(body(R),\tilde{a})\&N \models_b B}\, m_\alpha(body(R),\tilde{a})(b) \times$$

$$\times b[pr=pr(b) \oplus R] \oplus \Sigma_{b\in m_\alpha(body(R),\tilde{a})\&N \not\models_b B}\, m_\alpha(body(R),\tilde{a})(b) \times$$

$$\times b[pr=pr(b) \ominus R]) \times ctxt(a)),$$ где $\tilde{a} = a[ctxt = pr(a)(\tilde{R})(R)]$.

5. Исчисление RDL.

Аксиомы RDL.

(A1) все тавтологии классической логики предикатов,

(A2) $<p> (\Phi \vee \Psi) \equiv (<p>\Phi \vee <p>\Psi)$,

(A3) $<(p;q)>\Phi \equiv <p> <q>\Phi$,

(A4) $<p|q>\Phi \equiv <p>\Phi \vee <\check{q}>\Phi$,

(A5) если $\Phi \in F$, то $<\Phi>\Psi \equiv \Phi \wedge \Psi$,

(A6) $[p](\Phi \to \Psi) \to ([p]\Phi \to [p]\Psi)$,

(A7) если $R \in \tilde{R}$ и $r = body(R)$, то $[r]\Phi \to <[B \circ \tilde{R}]>\Phi$.

Правила вывода RDL:

$$(MP)\frac{\Phi,\Phi \to \Psi}{\Psi} \text{ modus ponens},$$

$$(G)\ \frac{\Phi}{[p]\Phi} \text{ generalization}.$$

ЛЕММА 1. *Если* $a \in \pi(\Phi_0)$, *то для всех* $d \in |N|$ *выполняется* $a[rv = d] \in (\Phi_0)$.

ДОКАЗАТЕЛЬСТВО. Обозначим $a' = a[rv = d]$.

1. Если Φ_0 - формула сигнатуры σ, то очевидно из определения.

2. Пусть $\Phi_o = \Phi \vee \Psi$. Тогда $a \in \pi(\Phi \vee \Psi) \Leftrightarrow a \in \pi(\Phi) \vee a \in \pi(\Psi)$. По индукции имеем $a' \in \pi(\Phi)$ или $a' \in \pi(\Psi)$, что эквивалентно утверждению $a' \in \pi(\Phi)$. Аналогично для формул вида $\Phi \wedge \Psi$, $\neg \Phi$.

3. Пусть $\Phi_o = \langle p\rangle\Phi . a \in \pi(\Phi_o) \Leftrightarrow (\exists b \in \pi(\Phi))(b \in m_\alpha(p,a))$. Нужно доказать, что в этом случае $(\exists b' \in \pi(\Phi))(b' \in m_\alpha(p,a'))$. Это верно, поскольку верно утверждение $b \in m_\alpha(p,a) \to b \in m_\alpha(p,a')$, что легко доказывается по индукции построения терма RDL.

Лемма доказана.

ТЕОРЕМА 1 (о кратности). *Если формула* Φ *выводима, то она общезначима. То есть для любой модели* RDL N *и для любого состояния* $s \in ST(N)$ *выполняется* $N \models_s \Phi$.

ДОКАЗАТЕЛЬСТВО.

Аксиома (A1). Для примера докажем, что если выводима формула $\neg\Phi$, то она истинна в RDL-семантике. Таким образом, нужно показать, что $N \models_s \neg\Phi$ тогда и только тогда, когда $N \not\models_s \Phi$.

По определению $N \models_s \neg\Phi$ означает, что

$$s \in \pi(\neg\Phi) \Leftrightarrow s \in ST(N) \setminus \pi(\Phi) \Leftrightarrow s \notin \pi(\Phi) \Leftrightarrow N \not\models_s \Phi ,$$

что и требовалось.

Аналогично: $N \models_s \Phi \vee \Psi \Leftrightarrow N \models_s \Phi$ или $N \models_s \Psi$.

Аксиома (A2). Докажем, что

$$N \models_s \langle p\rangle(\Phi \vee \Psi) \Leftrightarrow N \models_s \langle p\rangle(\Phi) \vee \langle p\rangle(\Psi).$$

Для этого достаточно показать $\pi(\langle p\rangle(\Phi \vee \Psi)) = \pi(\langle p\rangle(\Phi) \vee \langle p\rangle(\Psi))$. Обозначим правую часть через A. По определению π имеем $\pi(\langle p\rangle(\Phi) \vee \langle p\rangle(\Psi)) = \pi(\langle p\rangle\Phi) \cap \pi(\langle p\rangle(\Psi))$. Тогда

$$\begin{aligned} a \in A &\Leftrightarrow (\exists b \in \pi(\Phi) \cap \pi(\Psi))(b \in m_\alpha(p,a)) \Leftrightarrow \\ &\Leftrightarrow (\exists b \in \pi(\Phi))(b \in m_\alpha(p,a)) \vee (\exists b \in \pi(\Psi))(b \in m_\alpha(p,a)) \Leftrightarrow \\ &\Leftrightarrow a \in \pi(\langle p\rangle(\Phi) \text{ или } a \in \pi(\langle p\rangle)\Psi), \end{aligned}$$

что и требовалось.

Аксиома (A3). Нужно доказать, что $\pi(\langle p;q\rangle\Phi) = \pi(\langle p\rangle\langle q\rangle\ \Phi)$. Тогда $a \in \pi(\langle p;q\rangle\Phi) \Leftrightarrow (\exists c \in \pi(\Phi))(c \in m_\alpha((p;q),a)) \Leftrightarrow$ $\Leftrightarrow (\exists c \in \pi(\Phi))(\exists b \in m_\alpha(p,a))(c \in m_\alpha(q,b))$. С другой стороны, $a \in \pi(\langle p\rangle\langle q\rangle\ \Phi) \Leftrightarrow (\exists b \in \pi(\langle q\rangle\Phi)) \Leftrightarrow (\exists b)(b \in \pi(\langle q\rangle\Phi) \wedge$ $\wedge\ b \in m_\alpha(p,a)) \Leftrightarrow (\exists b)(\exists c)(c \in \pi(\Phi) \cap m_\alpha(q,b) \wedge b \in m_\alpha(p,a)$, что и требовалось.

Аксиома (A4). Нужно доказать, что $\pi(\langle p|q\rangle\ \Phi) = \pi(\langle p\rangle\Phi) \cup$ $\cup\ \pi(\langle q\rangle\Phi)$. Пусть $a \in \pi(\langle p|q\rangle\ \Phi)$. По определению это означает, что $(\exists b \in \pi(\Phi))(b \in m_\alpha(p|q,a)) \Leftrightarrow (\exists b)(b \in \pi(\Phi) \wedge$ $\wedge\ (\exists c)(\exists d)(c \in m_\alpha(p,a) \wedge b = c[rv = \langle rv(c),rv(d)\rangle] \wedge$ $\wedge\ (d \in m_\alpha(q,a) \vee b = d[rv = \langle rv(c),rv(d)\rangle]))$.

Из леммы 1 следует $(\exists c)(\exists d)(c \in m_\alpha(p,a) \wedge d \in m_\alpha(q,a) \wedge$ $\wedge\ (c \in \pi(\Phi) \vee d \in \pi(\Phi)) \rightarrow a \in \pi(\langle p\rangle\ \Phi) \cup \pi(\langle q\rangle\ \Phi)$.

Таким образом, получили включение $\pi(\langle p|q\rangle\ \Phi) \subseteq \pi(\langle p\rangle\Phi) \cup$ $\cup\ \pi(\langle q\rangle\ \Phi)$.

Теперь докажем обратное включение. Пусть $a \in \pi(\langle p\rangle\Phi) \cup$ $\cup\ \pi(\langle q\rangle\Phi)$. Тогда $(\exists c \in \pi(\Phi))(c \in m_\alpha(p,a))$ или $(\exists d \in \pi(\Phi))(d \in$ $\in m_\alpha(q,a))$. Предположим, выполнена первая альтернатива. Тогда пусть $b = c[rv = \langle rv(c),rv(d)\rangle]$. По лемме 1, $b \in \pi(\Phi)$ и,по определению, $b \in m_\alpha(p|q,a)$ влечет $a \in \pi(\langle p|q\rangle\Phi)$.

Аксиома (A5). Нужно доказать, что $\pi(\langle\Phi\rangle\Psi) = \pi(\Phi \wedge \Psi)$. Тогда $a \in \pi(\langle\Phi\rangle\Psi) \Leftrightarrow (\exists b \in \pi(\Phi))(b \in m_\alpha(\Phi,a)) \Leftrightarrow b \in \pi(\Phi) \wedge b = a \wedge$ $\wedge\ N \models_\alpha \Phi \Leftrightarrow a \in \pi(\Psi \wedge \Phi)$.

Аксиома (A6). Нужно доказать,что $\pi([p](\Phi \rightarrow \Psi)) \subseteq \pi([p]\Phi \rightarrow$ $\rightarrow [p]\Psi)$.

Пусть $a \in \pi([p]\Phi) = \pi(\neg\langle p\rangle\neg\Phi) = ST\setminus\pi(\langle p\rangle\neg\Phi) \Leftrightarrow$ $\Leftrightarrow (\forall b)(b \notin \pi(\neg\Phi) \vee b \notin m_\alpha(p,a)) \Leftrightarrow (\forall b)(b \in m_\alpha(p,a) \rightarrow b \in \pi(\Phi))$.

Теперь пусть $a \in \pi(\neg\langle p\rangle(\Phi \wedge \neg\Psi))$. Нужно показать, что в этом случае $a \in \pi(\langle p\rangle\neg\Phi \vee \neg\langle p\rangle\neg\Psi) = \pi(\langle p\rangle\neg\Phi\ \cup$ $\cup\ (ST\setminus\pi(\langle p\rangle\neg\Psi))$. Из предпосылки имеем $a \notin \pi([p]\Phi) \vee a \in$ $\in \pi([p]\Psi) \Rightarrow (\forall b)(b \in m_\alpha(p,a) \Rightarrow b \in (ST\setminus\pi(\Phi)) \cup \pi(\Psi)) \Rightarrow (\exists b)(b \notin$ $\notin \pi(\Phi) \wedge b \in m_\alpha(p,a)) \vee (\forall c)(c \in m_\alpha(p,a) \rightarrow c \in \pi(\Psi))$.

Аксиома (A7). Пусть $p \in R, type(b) = \underline{bool}$. Нужно доказать $\pi(<p>\Phi) \subseteq \pi(<[B \bigcirc \tilde{R}]>\Phi)$. Пусть $a \in \pi(<p>\Phi)$. Тогда $(\exists b \in \pi(\Phi))(b \in m_\alpha(p,a))$. Положим $c = b[pr = pr(b) \bigcirc p]$, где $\bigcirc \in \{+,-\}$ в зависимости от того $N \models_b B$ или $N \not\models_b B$. По лемме 1, $c \in \pi(\Phi)$. Таким образом, $(\exists c \in \pi(\Phi))(c \in m_\alpha([B \bigcirc \tilde{R}],a)) \Rightarrow$ $\Rightarrow a \in \pi(<[B \bigcirc \tilde{R}]>\Phi)$, что и требовалось.

Правило вывода (MP). Пусть $N \models_a \Phi$ и $N \models_a \Phi \rightarrow \Psi$. Нужно доказать, что $N \models_a \Psi$. Для этого докажем $\pi(\Phi) \cap \pi(\Phi \rightarrow \Psi) \subseteq \pi(\Psi)$. Это так, поскольку $\pi(\Phi) \cap \pi(\Phi \rightarrow \Psi) = \pi(\Phi) \cap (\pi(\neg\Phi) \cup \pi(\Psi)) =$ $= (\pi(\Phi) \cap \pi(\neg\Phi)) \cup (\pi(\Phi) \cap \pi(\Psi)) = (\pi(\Phi) \cap \pi(\Psi)) \subseteq \pi(\Phi)$.

Правило вывода (G). Нужно доказать, что если $(\forall a \in ST) N \models_a \Phi$, то $(\forall a \in ST) N \models_a [p]\Phi$. Поскольку $(\forall a \in ST) N \models_a \Phi$, то $\pi(\Phi) = ST$. Тогда очевидно $(\forall a \in ST)(\forall b \in m_\alpha(p,a))(b \in \pi(\Phi) = ST)$, т.е. $(\forall a \in ST)(N \models_a [p]\Phi)$. Теорема 1 доказана.

§ 2. Синтаксис описания классов

1. Базовые и определяемые классы. Мы будем рассматривать следующее множество базовых классов BC = {int, real, char, string, bool, void}. Множество определяемых классов мы будем обозначать DC, причем $BC \cap DC = \emptyset$. Множество всех классов Class= $= BC \cup DC$. Введем несколько базовых понятий.

1. Множество полей класса A будем обозначать Field(A).

2. Каждому $A \in DC$ соответствует некоторый класс $B \in Class$, являющийся для A родительским классом. Этот факт мы будем обозначать следующим образом: B = ancestor(A). Это отношение между классами индуцирует частичный порядок на множестве всех классов, задаваемый следующим образом: $B \subseteq A$ iff $B = ancestor(A) \vee$ $\vee B \subseteq ancestor(A)$.

3. Каждому $A \in Class$ соответствует его метакласс metaclass(A). Мы будем предполагать, что каждый класс C является объектом класса metaclass(A).

Для каждого класса $A \in DC$ существуют следующие множества имен методов и отношений:

- $Meth(A)$ - все методы класса A;

- $ProtMeth(A)$ - методы класса A, видимые для наследников класса A (т.е. всех классов C таких, что $A \sqsubseteq C$);

- $PubMeth(A)$ - методы класса A, видимые для всех классов;

- $Rel(A)$ - все отношения класса A;

- $ProtRel(A)$ - отношения класса A, видимые для наследников класса A;

- $PubRel(A)$ - отношения класса A, видимые для всех классов;

- $PermMeth(A) = (\bigcup_{C \in Class} PubMeth(C)) \cup Meth(A) \cup \cup PermMeth(ancestor(A))$ - множество видимых в классе A методов. Аналогично для отношений: $PermRel(A) = (\bigcup_{C \in Class} PubRel(C)) \cup \cup Rel(A) \cup PermRel(ancestor(A))$.

Помимо этого для любого $C \in BC$ имеем $PermRel = \emptyset$, $PermMeth = \emptyset$, $Meth = PubMeth = ProtMeth$, $Rel = ProtRel = PubRel$ и для $C \in Class$ имеем $PubMeth(C) \subset ProtMeth(C) \subset Meth(C)$ и $PubRel(C) \subset \subset ProtRel(C) \subset Rel(C)$.

Заметим, что $PubMeth(metaclass(A)) \subset PermMeth(A)$ и $PubRel(metaclass(A)) \subset PermRel(A)$.

5. Каждому методу каждого класса поставим в соответствие n-ку имен классов, являющихся типами соответствующих параметров метода:

$$arg: \bigcup_{C \in Class} Meth(C) \to \bigcup_{n=1}^{\infty} (Class)^n.$$

6. Каждому классу $C \in Class$ соответствует множество имен реализаций этого класса $Impl(C)$.

Для удобства будем считать, что $Rel \subset Meth \wedge ProtRel \subset \subset ProtMeth \wedge PubRel \subset PubMeth$. Это позволит нам обращаться с отношениями, как с булевыми методами классов.

Пусть VN - некоторое множество имен объектов (идентификаторов), для которого определена функция $type: VN \to Class$.Мы будем предполагать, что всегда $Class \subseteq VN$ и $(\forall C \in Class)(type(C) = metaclass(C))$.В дальнейшем это предположение позволит нам обращаться к методам метаклассов. Каждому классу $C \in DC$ поставим в соответствие множество "допустимых" термов $T_C(VN)$, тип термов $type: T_C(VN) \to Class$ и множество свободных переменных, соответствующее каждому терму $fv: T_C(VN) \to VN$, определяемые следующим образом.

1. Пусть $a \in VN$, тогда $t \rightleftarrows a \in T_C(VN), fv(t) = \{a\}, type(t) = type(a)$.

2. Пусть $t \rightleftarrows this$, тогда $t \in T_C(VN), fv(t) = \emptyset, type(T) = C$.

3. Пусть $t \rightleftarrows a \in Field(C)$, тогда $t \in T_C(VN)$, $f(v) = \emptyset$, $type(t) = type(a)$.

4. Пусть $F \in Meth(C)$, $type(F) = A$, $arg(F) = (C_1,\dots,C_n)$ и $t_1,\dots,t_n \in T_C(VN)$, $type(t_i) = C_i$.

Тогда

$$t \rightleftarrows F(t_1,\dots,t_n) \in T_C(VN),$$
$$type(t) = type(F),$$
$$fv(t) = \bigcup_{i=1}^{n} fv(t_i).$$

5. Пусть $A \sqsubseteq C$, $I \in Impl(A)$ и $F \in ProtMeth(A)$, $arg(F) = (C_1,\dots,C_n)$, $t_1,\dots,t_n \in T_C(VN)$, $type(t_i) = C_i$. Тогда

$$t \rightleftarrows A :: I : F(t_1,\dots,t_n) \in T_C(VN),$$
$$type(t) = type(F),$$
$$fv(t) = \bigcup_{i=1}^{n} fv(t_i).$$

6. Пусть $a \in Field(C) \cap VN$, $I \in Impl(type(A))$, $F \in PermMeth(C) \cap Meth(type(a))$, $arg(F) = (C_1,\dots,C_n)$, $t_1,\dots,t_n \in T_C(VN)$, $type(t_i) = C_i$. Тогда

$$t \rightleftharpoons a.I : F(t_1,\dots,t_n) \in T_C(VN),$$
$$type(t) = type(F),$$
$$fv(t) = \bigcup_{i=1}^{n} fv(t_i).$$

7. Пусть $p,q \in T_C(VN)$, $t_1 \rightleftharpoons (p|q)$ е $t_2 \rightleftharpoons (p;q)$, тогда $t_1 \in T_C(VN)$ и $t_2 \in T_C(VN)$; $type(t_1) = void$, $type(t_2) = type(q)$, $fv(t_1) = fv(t_2) = fv(p) \cup fv(q)$.

8. Пусть $A \subseteq C$, $B \in T_C(VN)$, $type(B)=bool$ и $F \in ProtMeth(A)$, $arg(F) = (C_1,\dots,C_n)$, $t_1,\dots,t_n \in T_C(VN)$, $type(t_i) = C_i$. Тогда

$$t \rightleftharpoons [A::F(t_1,\dots,t_n) \bigcirc B] \in T_C(VN),$$
$$type(t) = type(F),$$
$$fv(t) = \bigcup_{i=1}^{n} fv(t_i).$$

9. Пусть $a \in VN \cup Field(C)$, $B \in T_C(VN)$, $type(B) = bool$ и $F \in PermMeth(A) \cap Meth(type(a))$, $arg(F) = (C_1,\dots,C_n)$, $t_1,\dots,t_n \in T_C(VN)$, $type(t_i) = C_i$. Тогда

$$t \rightleftharpoons [a.F(t_1,\dots,t_n) \bigcirc B] \in T_C(VN),$$
$$type(t) = type(F),$$
$$fv(t) = \bigcup_{i=1}^{n} fv(t_i).$$

10. Пусть t_1, t_2, $b \in T_C(VN)$ и $type(t_1) = type(t_2) = A$, $type(b) = bool$. Тогда $t \rightleftharpoons$ if b then t_1 else $t_2 \in T_C(VN)$, $type(t) = A$, $fv(t) = fv(b) \cap fb(t_1) \cap fv(t_2)$.

11. Пусть t_1, $b \in T_C(VN)$ и $type(t_1) = A$ $type(b) = bool$. Тогда $t \rightleftharpoons$ while b do $t_1 \in T_C(VN)$, $type(t) = A$, $fv(t) = fb(b) \cap fv(t_1)$.

Множество замкнутых термов класса C $CT_C(VN) = \{t \mid t \in T_C(VN) \wedge fv(t) = \emptyset\}$.

2. <u>Общий вид описания определяемых классов.</u>

```
Class: <название_класса>,
PARENT CLASS: <название_родительского_класса>;
METACLASS: <название_мета_класса>;
DATA:
     <поле>: <название_класса>
      [,<поле>: <название класса>];
METHODS:
      <название_метода>:(<список_классов>)-><название_класса>
      [ ,<название_метода>:(<список_классов>)-><название_класса>];
RELATIONS:
      < название_отношения>:(<список_классов>)
      [,< название_отношения>:(<список_классов>)];
```

§3. Синтаксис записи реализации класса

Ниже приводится общий вид описания реализации класса

```
<реализация_методов_класса>=
CLASS_IMPLEMENTATION<название_реализации>:<название_класса>;
       METHODS:
        <реализация_метода>
        [<реализация_метода>]
       RELATIONS:
        <реализация_отношений>
```

1. <u>Реализация методов классов.</u>

Ниже приводится общий вид описания реализации метода класса:

```
<реализация_метода>=
    <название_класса> <название_метода>(<список_параметров>)
    BEGIN
```

<оператор>

END

<список_параметров>=

<параметр>

[,<параметр>]

<параметр>=

<название_класса> <идентификатор_объекта>

где <оператор> есть терм из $T_C(ObjId \cap Class \cap Field(C))$, причем множество ObjId содержит только <идентификаторы_объектов > из <списка_параметров> данного метода. Очевидным образом задается тип (функция type) идентификатора объекта.

2. <u>Реализация отношений классов</u>.Пусть VN - множество имен объектов, для которого определена функция type : VN $\to$ Class и известно, что Class $\subset$ VN. Определим множество допустимых формул класса $C \in DC$ $L_C(VN)$, а также для каждой допустимой формулы $\Phi \in L_C(VN)$ множество ее свободных переменных $fv(\Phi)$.

1. Если $t \in T_C(VN)$ и $type(t) = bool$,то $\Phi = (t) \in L_C(VN)$ и $fv(\Phi) = fv(t)$.

2. Пусть $\Phi, \Psi \in L_C(VN)$. Тогда $\Phi \vee \Psi$, $\Phi \wedge \Psi$, $\neg\Phi \in L_C(VN)$ $fv(\Phi \vee \Psi) = fv(\Phi \wedge \Psi) = fv(\Phi) \cap fv(\Psi)$ и $fv(\neg\Phi) = fv(\Phi)$.

3. Пусть $\Phi \in L_C(VN)$, $a \in fv(\Phi)$ и $type(a) = A$. Тогда $\Psi = (\forall a : A)(\Phi) \in L_C(VN)$ и $\Psi = (\exists a : A)(\Phi) \in L_C(VN)$. При этом $fv(\Psi) = fv(\Phi) \setminus \{a\}$.

Ниже приводится общий вид описания реализации отношений класса:

<реализация_отношений>=

<название_отношения>:<тип>(<список_параметров>)

BEGIN

<формула>

END

```
<тип>={
        { IFF | RESTRICTION}   /* ограничение' */
      | {IF       | DATABASED  }  /*  табличное отношение */
      }
<список_параметров>=
        <параметр>
        [,<параметр>]
<параметр>=
        <название_класса> <идентификатор_объекта>
```

где <формула> $\in L_C(VN)$ и VN = Field(C) $\cap$ Class$\cap$<список_параметров>.

Заключение

Автор признателен за помощь в подготовке настоящей работы академику С.С.Гончарову, своему научному руководителю академику Д.И.Свириденко, а также В.В.Ващенко,С.В.Котову, С.А.Луговому, К.В.Селютину и О.Г.Юрченко, общение с которыми весьма способствовало работе.

Литература

1. ГОНЧАРОВ С.С., СВИРИДЕНКО Д.И. Математические основы семантического программирования //Докл. АН СССР.-1986. -Т.289, № 6.

2. СВИРИДЕНКО Д.И. Проект СИГМА. Цели и задачи //Логические методы в программировании. - Новосибирск,1990.-Вып. 133: Вычислительные системы. - С. 68-94.

3. СВИРИДЕНКО Д.И., КОТОВ С.В. СИГМА-язык //Логические методы в программировании.-Новосибирск,1990. - Вып. 133: Вычислительные системы. - С. 95-134.

4. МАНЦИВОДА А.В. Σ-программирование и задачи дискретной оптимизации. - Иркутск. - 1994.

5. СТРАУСТРОП Б. Программирование на С++: Пер.с англ.-М.: 1992.

6. БУЧ Гради. Объектно-ориентированное программирование: Пер. с англ. - М.: 1992. - 502 с.

7. OpenDoc, Shaping Tomorrow's Software. (White Paper,Copyright 1993 Apple Computer, Inc.)

8. System Object Model (SOM) Reference.(IBM Technical Library 2.0, Copyright 1993 IBM Corp.)

Поступила в редакцию
20 апреля 1995 года