А.И. Белоусов

СЕМАНТИКА ЯЗЫКА ПРЕДСТАВЛЕНИЯ ЗНАНИЙ В ИНСТРУМЕНТАЛЬНОЙ СРЕДЕ ФРЕЙМОВОГО ТИПА

(ФГБОУ ВПО «Самарский государственный аэрокосмический университет им. академика С.П. Королева (национальный исследовательский университет)»)

Семантика**,** в широком смысле слова – анализ отношения между языковыми выражениями и миром, реальным или воображаемым, а также само это отношение и совокупность таких отношений (так, можно говорить о семантике некоторого языка). В данной статье речь идет о семантике языка представления знаний на основе фреймовых структур.

Определим на множестве фреймов инструментальной среды систему типов. В данной системе при выполнении операций присваивания в вычислении выражения производится строгая проверка типов. Используя результаты Сошникова [1] для систем с динамическим приведением типов, полагаем, что в нашем случае более слабые ограничения системы без типизации обязательно будут выполнены для систем со строгой типизацией. А именно:

Пусть система типов – $\tau = <T, \Theta>$,
где T – множество типов, Θ – множество операций.

Условия:

1. T – решетка с отношениями порядка $\subseteq$
2. $\exists$ NIL $\in$ T: $\forall_x \in T$ $NIL \in X$
3. $\forall (*)_T$: $\forall_{x,y} \in T$ x y $\in (*)$
4. $\forall (*)_T$: $\forall_x \in T$ NIL x = x $(*)$ L = NIL $(*)$

Будем рассматривать следующие типы данных:

1. $\mathbb{B}$ = {true, false}– логический тип.
2. $\mathbb{N}$ – множество натуральных чисел, представимых в ЭВМ с разрядностью не менее 32.
3. $\mathbb{D}$ – множество действительных чисел, представимых в ЭВМ с разрядностью не менее 32.
4. $\mathbb{S}$ – множество строк в некотором алфавите A
5. $\mathbb{R}$ – множество ссылок, $\mathbb{R} = \mathbb{R}_1 + \mathbb{R}_2$, где $\mathbb{R}_1$ = I, $\mathbb{R}_2$= <f,s>, f$\in$ I, s $\in I_f$
6. $\mathbb{M}= \mathbb{M}_1 + \mathbb{M}_2$ – множество массивов одномерных и двумерных.
7. $\mathbb{L}$ – множество списков конечной длины из элементов одного типа

$T \in T$. T={NIL}+$\mathbb{B}$+$\mathbb{N}$+$\mathbb{D}$+$\mathbb{S}$+$\mathbb{R}$+ $\mathbb{M}$ + $\mathbb{L}$

В [1] показано, что при таких предположениях даже для нестрого типизированной системы при рассмотрении графа состояний логического вывода можно в анализе ограничиться анализом графа с конечным множеством состояний и конечным множеством переходов.

Так как на всех рассматриваемых типах может быть введено отношения порядка $\subseteq$, то при условии неизменности знаний о значениях по умолчанию ссылок по агрегациям и обобщениям, а так же процедурных знаний, отношение порядка может быть определено на множестве пространства состояний и в частности на множестве $\mathcal{W}_{CA}$. В дальнейшем, учитывая неизменность состояний информационной базы в процессе логического вывода, индекс «CA» для состояний будем опускать, считая что рассуждения о смене состояний ведутся лишь для фреймов участвующих в логическом выводе, для которых изменяются значения f.s.v и f.s.α и если α – признак активности фрейма в логическом выводе, то при f.s.α=false слот s фрейма f не может изменить свое значение в процессе вывода, что должно быть проконтролировано в процедурах вывода.

В системе все фреймы являются наследниками корневого системного фрейма. Отношение isa (определяет иерархию фреймов и используется для переноса слотов и их значений от родительского фрейма к потомкам). Для записи иерархии наследования воспользуемся обозначением стрелки обобщения, принятой в UML [2]. $f \to g \Leftrightarrow \| f.isa \| = g$. Тогда и только тогда – $\|\cdot\|$ операция вычисления значения слота (ссылочного типа) f.isa $\in \mathbb{R}_1 \subseteq \mathrm{I}$. Ограничиваясь случаем однозначного наследования, будем считать что эти отношения не меняются в процессе вывода, а для записи прямых родителей в системном фрейме – прототипа предусмотрен специальный слот f.isa со значениями ссылочного типа.

Фреймы–экземпляры являются конечными в иерархии и наследников не имеют [3] Отсутствовать значение может только у системного фрейма самого верхнего уровня и он только один. У остальных фреймов значение прямого родителя имеется обязательно и заполняется при создании нового частного фрейма-прототипа или фрейма–экземпляра. Поскольку наследование в системе является обязательным для всех фреймов то при обращении к слоту за значением («чтение») и его отсутствием проверяется нельзя ли определить это значение по цепочке «снизу–вверх» обобщений (это может быть зафиксировано одним из предикатов $C_i \in C_k$, прикрепленных к данному слоту). Такая ситуация возможна в процедуре IF_NEEDED_i которая так же наследуется от родителя [4].

Будем считать, что каждая процедура демон и процедурный фрейм представляет собой типизированные выражения $E \in \mathbb{E}$ или композицию таких выражений $C \in \mathbb{C}$ и являются вычислимой. Вычисление такого выражения в контексте может иметь следующие результаты:

1. Прерывание если требуется значение другого слота данного фрейма или другого а оно не задано, передача управления соответствующей процедуре IF_NEEDED_i другого слота.

2. Возврат значения v или множества значений $\{v_1, v_1, \ldots, v_n\}$ в процессе вычисления.
3. Возврат значения NIL если одна из компонент выражения NIL и определить ее не удается (в том числе из–за ошибок в вычислениях).

Будем считать, что прерванные состояния системы сохраняются, так что любое продолжение вычисления выражения приводит к результатам 2 и 3. Это соответствует: $\| E \|^c = v; v \neq NIL$; $\| E \|^c = \{v_1, v_2, .., v_n\}; v_i \neq NIL;$ $\| E \|^c = NIL$, где $\| E \|^c$ – результат вычисления выражения E в контексте $C \in \mathbb{C}$.

Определим синтаксис множества выражений. Выражения «продвигающие» шаги вывода делятся как и фреймы на активные (содержащие операции присваивания, записи нового значения слота) и пассивные (информационные) вспомогательные не содержащие операций изменения значений слотов и лишь подготавливающие промежуточные данные для вычисления выражений на последующих шагах вывода, однако демоны IF_NEEDED всегда активны, если не завершаются ошибкой, а демоны IF_CHANGED могут быть как активными, так и пассивными.

Для выражения E=f.s определим следующую систему вычисления значений слота:

$$\| f.s \|^c = \begin{cases} f.s.v, если\ v \neq NIL \\ f.s.u, если\ v = NIL\ и\ f.s.u \neq NIL \\ \| f.s.IF\_NEEDED_i \|, если\ v = NIL, \\ f.s.IF\_NEEDED_i \neq NIL,\ f.s.u = NIL \\ NIL, если\ v = NIL,\ f.s.u = NIL,\ f.s.IF\_NEEDED_i = NIL \end{cases}$$

Предполагается, что контекст $C \in C_k$ разрешает активизировать только одну процедуру IF_NEEDED из множества $\{Q_i\}, \overline{i = 1, n}$ (одновременный вызов двух процедур–оракулов считается конфликтом и должен исключаться предикатом или правилом с контекста). Значение f.s.v может быть установлено равным значению g.s.v соответствующего родительского фрейма, если это разрешено контекстом, при выполнении операции создания нового фрейма, Все процедуры-демоны и присоединенные процедуры и правила наследуются потомками, что учитывается при создании фрейма. Если контекст C отсутствует (C =NIL), то $\| f.s \|^{NIL} = \| f.s \|$, но при этом используется первая по порядку (младшая) процедура $f.s.IF\_NEEDED_1 = Q_1$. Процесс вычисления выражения $f.s.IF\_NEEDED_i$ может включать в себя вызов функции оракула (запроса к внешней среде с использованием некоторой текстовой команды – дескриптора dsc)

$\mathcal{A}$: $\mathbb{D}sc \rightarrow \mathbb{T}\ \mathbb{D}sc$ – некоторое множество дескрипторов зависящие от предметной области

$\| \mathcal{A}(dsc) \|^c = \mathcal{A}(dsc)$, где $dsc \in \mathbb{D}sc$, $C \in C_k$

Вызов функции–оракула определяет диалоговый режим консультации (режим задачи). В предположении, что множество возможных ответов пользователя контролируется и не выводит процесс за пределы типа значений соответствующего слота в операции присваивания или не меняет состояния системы вывода, лишь подготавливая промежуточные данные для продолжения вычисления выражения $f.s.IF\_NEEDED_i$, считаем, что семантика языка представления знания вызовом функции-оракула не изменяется в части логического вывода. Основным свойством данной семантики, являющейся частным случаем семантики, рассмотренной в [1,6] является монотонность операции вывода. Для любого контекста $C \in \mathbb{C}$ и выражения $E \in \mathbb{E}$ функция $\|\in\|^C$: $\mathcal{W}_{CA} \to \mathcal{W}_{CA}$ определена и монотонна по изменении состояния в соответствии с упорядоченностью используемых процедур-демонов и контекстных ограничений. Это приводит к конечности числа переходов в пространств состояний при использовании корректных выражений $E \in \mathbb{E}$.

Контекст $C_k \subseteq \mathbb{C}$ определяет в том числе и семантику наследования при вычислениях. В принципе часть этого контекста выделенная в виде набора правил IF_USED [1] прикрепленных к фрейму f в целом может определять условия использования значений слотов, демонов и присоединенных процедур, родительских фреймов.

Будем рассматривать следующие операции для композиции выражений в множестве $\mathbb{E}$:

1. Подстановка константы из множества типов $\mathbb{T}$.
2. Ссылка на слот f.s, $f \in I$, $s \in I_f$
3. Арифметическая операция $E_1 \ldots \in \ldots, /$
4. Логическая операция $E_1 \ldots$ или NOT E_1
5. Условная операция $E_0 \to E_1, E_2$ или $E_0 \to E_1$
6. Вызов функции-оракула (запросы значений путем обращения к внешней среде) $\mathcal{A}$

Арифметические и логические операции должны соответствовать множеству типов $\mathbb{T}$ и условиям строгой типизации выражений из IE (общие ограничения на множестве типов уже были рассмотрены)

Константа соответственного типа может подставляться вместо переменной или выражения в целом независимо от состояния системы вывода.

Условное выражение вычисляется по схеме:

$$\| E \to E_1, E_2 \|^c = \begin{cases} \| E_1 \|^C, \| E \|^C = true \\ \| E_2 \|^C, \| E \|^C = false \end{cases}$$

Литература

1. Сошников Д.В. Инструментарий JULIA для построения распределённых интеллектуальных систем на основе продукционно-фреймового представления знаний // Электронный журнал «Труды МАИ». – М.: МАИ, 2002. – №7

2. Дерябкин В.П., Либерзон О.К. Методические указания к лабораторному практикуму по UML: учебное пособие для вузов. – Самара: 2008. – 40 с.

3. Белоусов А.И., Дерябкин В.П. Фреймовая база знаний информационной компьютерной среды // Перспективные информационные технологии для авиации и космоса (ПИТ-2010). Избранные труды Международной конференции с элементами научной школы для молодёжи. – Самара, СГАУ, 2010. – с. 61 – 64

4. Белоусов А.И., Дерябкин В.П. Унифицированное представление знаний продукционно-фреймового типа // Стандартизация информационных технологий и интераперабельность СИТОП 2011, Пятая всероссийская конференция. – М.: МИРЭА, 2011 – с. 12 – 16

5. Minsky M. A Framework for representing knowledge. – Cambridge: MIT Press, 1974

6. Сошников Д.В. Логический вывод на основе удалённого вызова и включения в системах с распределённой фреймовой иерархией / под ред. В.Е. Зайцева. – М.: «Вузовская книга», 2002. – 48 с.