Системный анализ, управление и автоматизация

УДК 681.518.3

СЕМАНТИКА ЯЗЫКА ПРЕДСТАВЛЕНИЯ ЗНАНИЙ В ИНТЕЛЛЕКТУАЛЬНОЙ ФРЕЙМОВОЙ СРЕДЕ

А.И. Белоусов, В.П. Дерябкин

Самарский государственный аэрокосмический университет им. ак. С.П. Королёва
443086, г. Самара, Московское ш., 34

E-mails: timone65@yandex.ru; deriabkin_v@mail.ru

Продукционно-фреймовая среда представления знаний предоставляет возможность построения интеллектуальных систем. Ядром такой системы является база знаний с использованием фреймовых структур как унифицированных элементов представления знаний. В статье рассмотрена и уточнена в соответствии с унифицированной моделью фрейма модель базы знаний инструментальной системы и семантика языка представления знаний.

Ключевые слова: *фрейм, слот, фрейм-прототип, фрейм-экземпляр, база знаний, унифицированное хранение знаний, экспертная система, продукции.*

Введение

Качество и своевременность принятия решений в сложных ситуациях во многом определяются информационным окружением лица, принимающего решение. В [1 – 3] рассмотрена концепция построения интеллектуальной информационной компьютерной среды с фреймовой структурой знаний, которая позволяла бы создавать, развертывать и исполнять интеллектуальные приложения в различных предметных областях. Ядром такой среды является база знаний с использованием фреймовых структур как унифицированных элементов представления знаний. Несмотря на то, что фреймовая модель представления знаний предложена давно [4], до сих пор формализация процесса решения задач вызывает значительные трудности [5, 6].

В статье рассмотрена и уточнена в соответствии с унифицированной моделью фрейма модель базы знаний инструментальной системы и семантика языка представления знаний.

Структура базы знаний системы

Фреймы базы знаний могут быть подразделены на группы:

– активные в логическом выводе результата, имеющие слоты с исполняемыми присоединенными процедурами (демонами и/или значениями слотов) F_a – поведение этих фреймов определяет логический вывод;

– пассивные, не имеющие слотов с присоединенными процедурами F_p (они пред-

Артем Игоревич Белоусов, ассистент кафедры «Информационные системы и технологии».

Валентин Петрович Дерябкин (к.т.н.), доцент кафедры «Информационные системы и технологии».

ставляют фреймы-структуры, аналогичные таблицам, из которых только можно брать информацию в процессе вывода, но изменение значений слотов на время вывода запрещено).

К активным фреймам относятся системные фреймы, например фрейм приложения и др., имеющие хотя бы одну присоединенную исполняемую процедуру.

Общая схема решения задачи зависит от того, в какой форме ищется решение:

1) традиционно, в форме алгоритма по данным пассивных фреймов;

2) по образцу решения, наиболее близкого к образцу среди пассивных фреймов с использованием той или иной меры близости;

3) методом распространения возбуждения по сети фреймов, содержащих все возможные решения проблемы в группе конечных фреймов-результатов (экземпляров) по указанным экземплярам входных фреймов-параметров информационного запроса.

В системе должен иметься исходный активный фрейм запроса стандартной структуры, содержащий слоты параметров запроса, а также слот результата.

Следует различать базу знаний среды разработки – общую базу знаний, применимую для всех возможных приложений (справочная библиотека), и частную базу знаний (приложение), полностью определяющую состав и структуру частных фреймов-прототипов и экземпляров, специфичных для конкретного приложения.

Фрейм конкретного приложения является в общем случае агрегацией фреймов, входящих в информационную базу (пассивных во время решения задачи и переходящих в активные при вводе (сборе) информации) и активных фреймов логического вывода (в частном случае – одного).

Чаще всего приходится разделять фрейм-ситуацию «запрос» на частные фреймы достижения промежуточных результатов (агрегация фреймов-запросов).

Полная база знаний

$$K = K_{системы} + K_{приложений}, \quad K_{приложений} = \bigcup_{i=1}^{N} K_{\text{п}i} ,$$

где K_{ni} – частная база знаний i-го приложения.

В состав базы знаний входят знания статические K_s (структуры фреймов, значения слотов на множестве типов Т) и динамические K_d (системные методы и процедуры инструментальной системы, присоединенные процедуры и процедуры-демоны фреймов-приложений)

$$K = K_s + K_d = K_{s\_sys} + K_{d\_sys} + \bigcup_{i=1}^{N} K_{s\_app_i} + \bigcup_{i=1}^{N} K_{d\_app_i} ,$$

где индекс *sys* относится к инструментальной системе, индекс *app*i – к i-му приложению.

Фреймы

Фрейм как единица представления знаний отображает как статические, так и динамические знания, причем все множество фреймов *F*, эквивалентное знаниям *K*, представляется в виде

$$F = F_{sys} + F_{app1} + F_{app2} + ... + F_{appN},$$

где F_{appi} – множество фреймов i-го приложения (множества могут пересекаться).

Фрейм рассматривается как некоторый контейнер (пакет) знаний, имеющий имя (идентификатор $id \in I$, $I \subset II$ (*II*– полное множество идентификаторов фреймов и слотов) и содержащий как агрегацию набор слотов. Каждый слот $s \in S$ является также носителем статических и динамических знаний в виде возможных значений,

определенных на системе типов T, и набора присоединенных процедур из некоторого множества выражений E, λ-выражений $\Lambda(T)$; $E \subseteq \Lambda(T)$ [6]. В частном случае λ-выражение может включать константы, в том числе имена присоединенных процедур, а также ссылки на другие слоты и значения NIL (неизвестно).

Для формализации фреймовых структур будем (аналогично [6]) рассматривать функцию состояний w: $I \to F$, отображающую множество идентификаторов в множество фреймов (эта функция фиксирует состав всей базы знаний на текущий момент); $w \in W$, где W – множество возможных состояний.

Обозначим через $W_{IA} \subseteq W$ состояние информационной базы приложения, которое определяет часть фреймов приложения (системных и приложения), пассивных по отношению к логическому выводу, а через $W_{CA} \subseteq W$ – состояние системы управления вывода (решателя) – набора активных фреймов.

Каждый фрейм $f \in F$ является функцией $f : I_f \to S$, отображающей множество идентификаторов слотов фрейма в множестве слотов S. Динамические знания представляются множествами демонов и присоединенных процедур и правил. Состояние W_{IA} информационной базы приложения на период вывода считается неизменным.

Считаем, что фреймы включают слоты, задаваемые в общем случае кортежами

$$S = \{< v, u, \{Q_i\}, \{D_j\}, \{C_k\}, \alpha >\},$$

$v \in T$ – значение слота; $u \in T$ – значение слота по умолчанию; $\{Q_i\}$ – упорядоченное множество присоединенных к слотам процедур-демонов поиска значений слота типа IF_NEEDED; $\{D_j\}$ – упорядоченное множество присоединенных к слоту процедур-демонов типа IF_CHANGED, обрабатывающих событие изменения значения слота; $\{C_k\}$ – упорядоченное множество ограничений на значения слота (набор правил или предикатов $C_k \in E$); α – флаг, используется в контексте вывода для управления выводом, булевского типа.

Для доступа к значениям слота используем операцию разыменования:

$$s.v,\ s.u;\ s.\text{IF\_NEEDED}_i\ ;\ s.\ \alpha \text{ и т. п.}$$

Считая идентификаторы фреймов уникальными во всей системе, а идентификаторы слотов – уникальными в пределах фрейма, в операции разыменования объекты f и s заменяем их соответствующими идентификаторами и в этом контексте можем считать, что запись $f.s$ одновременно является идентификатором (адресом) слота s во фрейме f.

В первоначальном состоянии (на нулевом шаге вывода) $f.s.v$ = NIL для всех слотов всех фреймов подсистемы вывода, а также $f.s.\alpha$ = NIL.

Операции присваивания значения слоту будем рассматривать как функцию, в общем случае осуществляющую изменение состояния:

a) системы в целом;

b) информационной базы в режиме только пополнения длительно хранимых знаний;

c) только подсистемы вывода в процессе получения результата решения задачи при неизменном состоянии информационной базы.

Система типов фреймовой среды

Определим на множестве фреймов инструментальной среды систему типов. В данной системе при выполнении операций присваивания в вычислении выражения производится строгая проверка 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. $\exists \text{NIL} \in T : \forall x \in T \quad \text{NIL} \in X, x \in X$.
3. $\forall (*) \in E : \forall x, y \in T \quad x (*) y \in T$.
4. $\forall (*) \in E : \forall x \in T \, \text{NIL} (*) x = x (*) \text{NIL} = \text{NIL}$.

где $(*)$ - множество арифметический операций.

Будем рассматривать следующие типы данных:

1. B = {true, false} – логический тип.
2. N – множество натуральных чисел, представимых в ЭВМ с разрядностью не менее 32.
3. D – множество действительных чисел, представимых в ЭВМ с разрядностью не менее 32.
4. P – множество строк в некотором алфавите A.
5. R – множество ссылок, R = R1 + R2, где R1 = I, R2 = <*f, s*>, $f \in I, s \in I_f$.
6. M = M1 + M2 – множество массивов одномерных и двумерных.
7. L – множество списков конечной длины из элементов одного типа T = {NIL} + B + N + D + S + R + M + L.

Так как на всех рассматриваемых типах может быть введено отношение порядка $\subseteq$, то при условии неизменности знаний о значениях по умолчанию ссылок по агрегациям и обобщениям, а также процедурных знаний отношение порядка может быть определено на множестве пространства состояний, и в частности на множестве 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.

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 также наследуется от родителя [7].

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:

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

Предполагается, что контекст $C \in C_k$ разрешает активизировать только одну процедуру IF_NEEDED из множества $\{Q_i\}$, $i = \overline{1,n}$ (одновременный вызов двух процедур-оракулов считается конфликтом и должен исключаться предикатом или правилом *c* 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 $\| f.s \|^{\text{NIL}} = \| f.s \|$, но при этом используется первая по порядку (младшая) процедура $f.s.\text{IF\_NEEDED}_1 = Q_1$. Процесс вычисления выражения $f.s.\text{IF\_NEEDED}_i$ может включать в себя вызов функции оракула (запроса к внешней среде с использованием некоторой текстовой команды – дескриптора *dsc*).

Контекст $C_k \subseteq C$ определяет в том числе и семантику наследования при вычислениях. В принципе, часть этого контекста, выделенная в виде набора правил IF_USED, 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 во множестве $E(E_1 \in E, E_2 \in E)$.

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

где $\boxed{.}$ - множество логических операций.

Арифметические и логические операции должны соответствовать множеству типов 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

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

Заключение

Изложенный подход и нотация могут быть использованы в процессе разработки языка представления знаний и обоснования стратегий решения задач в интеллектуальной информационной среде с унифицированной фреймовой базой знаний.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Дерябкин В.П.* База знаний системы синтеза и параметрической настройки проблемно-ориентированной информационной компьютерной среды // Перспективные информационные технологии в научных исследованиях, проектировании и обучении «ПИТ-2006», Т. 1. – Самара: СГАУ, 2006. – С. 65-69.
2. *Дерябкин В.П.* Среда визуальной разработки интеллектуальных систем продукционно-фреймового типа // Математическое моделирование информационных процессов и систем в науке, технике и образовании. – Самара: СГАСУ, 2010. – С. 48-51.
3. *Дерябкин В.П., Белоусов А.И.* Фреймовая база знаний информационной компьютерной среды // Перспективные информационные технологии для авиации и космоса (ПИТ-2010). Избранные труды Международной конференции с элементами научной школы для молодежи. – Самара: СГАУ, 2010. – С. 61-64.
4. *Minsky M.* A Framework for Representing Knowledge / M.A. Minsky. – Cambridge: MIT Press, 1974.
5. *Джарратано Д.* Экспертные системы: принципы разработки и программирования, 4-е изд. / Д. Джарратано, Г. Райли. – М.: ИД «Вильямс», 2007. – 1152 с.
6. *Сошников Д.В.* Логический вывод на основе удаленного вызова и включения в системах с распределенной фреймовой иерархией / Д.В. Сошников // Под ред. В.Е. Зайцева. – М.: Вузовская книга, 2002. – 48 с.
7. *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.

Статья поступила в редакцию 6 февраля 2013 г.

LANGUAGE SEMANTICS OF KHOWLEDGE REPRESENTATION IN THE INTELLECTUAL FRAME-BASED ENVIRONMENT

A.I. Belousov, V.P. Deriabkin

S.P. Korolyov Samara State Aerospace University
34, Moskovskoye sh., Samara, 443086

Production-frame-based environment of knowledge representation allows to construct intelligent system. The core of such system is a knowledge base using the frame structures as standard elements of knowledge representation. In the article a model of knowledge base of development system and semantics of knowledge representation are discussed and refined in accordance with the unified model of frame.

Keywords: *frame, slot, frame-prototype, frame-instance, knowledge base, unified storage knowledge, expert system, production.*

Artem I. Belousov, Assistant.
Valentin P. Deriabkin (Ph.D. (Techn.)), Associate Professor.