

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

УДК 004.[822+823]

СЕМАНТИЧЕСКИЕ ФРЕЙМЫ: КЛАССИФИКАТОРЫ И КВАЛИФИКАТОРЫ

А.И. ДУНЧЕНКО

(Представлено: Д.В. ПЯТКИН)

Рассматривается концепция представления знаний в интеллектуальных компьютерных системах. Раскрывается сущность новой модели, основанной на семантических фреймах и ее связь с уже существующими решениями. Описываются отличительные особенности – классификаторы и квалификаторы, которые являются дополнениями к базовой фреймовой модели.

Введение. При создании систем искусственного интеллекта ключевым вопросом является выбор способа представления и обработки информации об объектах и явлениях реального мира применительно к возможностям современных ЭВМ. В зависимости от типа решаемой задачи, существует два подхода к этой проблеме. *Первый* – это использование искусственных нейронных сетей (ИНС) [1]. ИНС – это аппаратно или программно реализованная сеть, работа которой основана на тех же принципах, что и мозг высших представителей фауны. Каждая такая ИНС получается путем комбинирования множества различных нейронов, реализующих ту или иную математическую функцию. Недостатком нейронных сетей является то, что они требуют длительного обучения и являются ресурсоемким решением. Вторым подходом заключается в применении специфической нотации. Традиционными представителями данной группы являются: *правила* (выражения IF-THEN-ELSE), *предикаты* [2], *семантические сети* [3], *фреймы* [4].

Ни одно из перечисленных решений не является универсальным: все они обладают как определенными преимуществами друг перед другом, так и некоторыми недостатками. Поэтому выбор той или иной нотации сильно зависит от конкретно поставленной перед разработчиками задачи.

Семантические фреймы (СФ) – это попытка создания новой модели представления знаний, в основе которой лежит все та же концепция фреймов, только дополненная элементами из семантических сетей, предикатов и нечеткой логики.

В модели СФ отдельно взятый слот или даже целый фрейм может выражать *семантическое отношение* между связанными элементами. Для этой цели служат *классификаторы*.

Классификаторы

Классификатор – это символическая константа (в текущей спецификации это так; в будущих версиях вполне возможно, что константы будут заменены на классы, методы которых будут определять правила управления отношениями), выражающее характеристику своего операнда в наиболее обобщенной форме. То есть при определении перечня стандартных классификаторов делался упор на минимализм: из ряда близких по смыслу терминов выбирался тот, который имеет более абстрактное значение, включающее в себя множество смежных понятий. Каждый слот или фрейм может иметь несколько не противоречащих друг другу классификаторов, правила, назначения которых определяются спецификацией и приведены в таблицах 1–7.

Таблица 1. – Классификаторы фреймов

Классификатор	Описание
<i>set</i>	Коллекция связанных сущностей: сфера, группа, пространство имен, пакет и т.п.
<i>class</i>	То же, что и в ООП.
<i>prototype</i>	То же, что и в ООП.
<i>enumeration</i>	Перечисление символических констант (аналогично <i>enum</i> в С). Например – <i>сезон: зима, весна, лето, осень</i> .
<i>relation</i>	Множество тесно связанных сущностей. Слоты содержат объекты, связанные данным отношением.
<i>activity</i>	Используется для описания процедурных знаний. Слоты содержат участников (затрагиваемые объекты) и характеристики деятельности (события, явления и т.п.).
<i>object</i>	Экземпляр некоторого класса/прототипа или анонимный объект без определенного типа.

Таблица 2. – Модификаторы

Классификатор	Описание
<i>fixed</i>	Не позволяет добавлять/удалять слот из фрейма или изменять значение слота.
<i>not-inherited</i>	Фреймы, помеченные данным классификатором, не могут наследоваться (<i>class, enumeration</i>). Слоты с таким классификатором не наследуются.
<i>shared</i>	Экземпляры класса вместо копии получают ссылку на слот.
<i>attribute</i>	Анонимный слот. Значение слота содержит некоторую обобщенную характеристику (прилагательное, наречие, числительное).

Таблица 3. – Классификаторы слотов *set*-фреймов

Классификатор	Описание
<i>includes</i>	Безымянный слот, содержащий ссылку на вложенное множество.
	Слоты без перечисленных выше классификаторов содержат обычные элементы множества.

Таблица 4. – Классификаторы слотов *class*-фреймов

Классификатор	Описание
<i>extends</i>	Безымянная ссылка на базовый (расширяемый) класс.
<i>includes</i>	Безымянная ссылка на вложенный класс.
<i>activity</i>	Слот описывает деятельность, производимую экземпляром класса, или событие, на которое он может реагировать.
	Слоты без перечисленных выше классификаторов содержат поля, копии которых будут назначены всем экземплярам класса.

Таблица 5 – Классификаторы слотов *enumeration*-фреймов

Классификатор	Описание
<i>extends</i>	Безымянная ссылка на базовое (расширяемое) перечисление.
<i>activity</i>	Слот описывает деятельность, производимую элементом перечисления, или событие, на которое он может реагировать.
	Слоты без перечисленных выше классификаторов содержат элементы перечисления, а их имена являются идентификаторами этих значений.

Таблица 6. – Классификаторы слотов *relation*-фреймов

Классификатор	Описание
<i>extends</i>	Безымянная ссылка на базовое (расширяемое) отношение.
<i>activity</i>	Слот описывает деятельность, подразумеваемую данным отношением, или событие, которое может наступить в определенный момент.
	Слоты без перечисленных выше классификаторов содержат элементы, связанные отношением (являющиеся активными или пассивными участниками).

Таблица 7. – Классификаторы слотов *activity*-фреймов

Классификатор	Описание
<i>in</i>	Входной параметр (аргумент).
<i>out</i>	Выходной параметр (результат).
<i>in activity</i>	Деятельность, которая может инициировать данную активность (причина).
<i>out activity</i>	Деятельность, которая может быть инициирована данной активностью (последствие).
<i>includes</i>	Деятельность, на которой основана текущая активность.
	Слоты без перечисленных выше классификаторов содержат объекты, которые подвергаются воздействию или являются участниками деятельности.

Необходимо отметить, что слоты, помеченные классификатором *extends*, всегда располагаются в первой секции фрейма (т.е. в самом начале списка); за ними могут следовать *includes*-слоты, а уже затем все остальные. Исключением являются *activity*-фреймы – для них порядок следующий: входные параметры, выходные параметры, *includes*-слоты, прочие слоты. У экземпляров классов и прототипов первый слот должен иметь классификатор *class* или *prototype* и ссылаться на соответствующий объект.

Квалификаторы

Квалификатор – это предикат, записанный в форме *F-выражения*. В модели СФ квалификаторы служат для внесения ограничений, проверки соответствия условиям и т.п. Следует сразу указать, что система представления знаний с помощью СФ все еще находится на стадии разработки и апробации, поэтому в текущей спецификации потенциал *F-выражений* реализуется не полностью. На данный момент квалификаторы могут использоваться в качестве триггеров для проверки значений, присваиваемых слотам.

F-выражение (от англ. *F-expression* – functional expression) представляет собой запись, идентичную математической нотации определения *функции* [5], которая в форме *EBNF* [6] выглядит так:

```
f-expression = identifier, '(', arguments, ')';
arguments    = [ argument, { ',', argument } ];
argument     = value;
array        = '[', [ value, { ',', value } ], ']';
value        = f-expression | identifier | constant | array.
```

Причины, по которым был выбран такой синтаксис:

- 1) *F-выражения* по своей природе подобны функциям и не должны иметь побочных эффектов;
- 2) данный синтаксис, в отличие от *M-выражений* [7], используется для объявления/вызова функции во многих языках программирования (ЯП), производных от Algol, и, поэтому, является более привычным для восприятия;
- 3) данный синтаксис, в отличие от *символических выражений* [8], не привязан к конкретному ЯП (Lisp) и, используя *регулярные выражения* [9], может быть легко сконвертирован в *AST* [10], построенное на базе простых списков.

В отличие от обычных функций, *F-выражения* получают свои параметры в исходной форме и могут вычислять их по мере необходимости. Это позволяет унифицировать понятия *макрос* и *функция*.

Так, в зависимости от значения первого аргумента, форма *if(..., ..., ...)* вычисляет и возвращает либо второй, либо третий параметр.

Каждый квалификатор, будучи предикатом, по сути, должен возвращать строгое *логическое значение* [11]: $P() \in \{ \perp, T \}$.

Следует заметить, что в концепции *F-выражений* используется подобие *троичной логики* [12], где значения $(\neg 1, 0, 1)$ представлены соответственно константами $\$f$, $\$?$ и $\$t$.

Результаты логических операций над ними представлены в таблице 8.

Таблица 8. – Логические операции в троичной логике

A	B	$\neg A$	$A \wedge B$	$A \vee B$	$A \rightarrow B$	$A \oplus B$	$A = B$
\$f	\$f	\$t	\$f	\$f	\$t	\$f	\$t
\$f	\$?	\$t	\$f	\$?	\$t	\$?	\$?
\$f	\$t	\$t	\$f	\$t	\$t	\$t	\$f
\$?	\$f	\$?	\$f	\$?	\$?	\$?	\$?
\$?	\$?	\$?	\$?	\$?	\$?	\$?	\$?
\$?	\$t	\$?	\$?	\$t	\$t	\$?	\$?
\$t	\$f	\$f	\$f	\$t	\$f	\$t	\$f
\$t	\$?	\$f	\$?	\$t	\$?	\$?	\$?
\$t	\$t	\$f	\$t	\$t	\$t	\$f	\$t

Важным дополнением является то, что константа \$? (undefined) представляет собой *суперзначение*, символизирующее отсутствие конкретных знаний как таковых, поэтому она может фигурировать в выражениях, оперирующих различными типами, например:

$$\text{sum}(1,2,3, \$?) = \$?; \text{union}([1, 2, 3], [4, 5, 6], \$?) = \$?.$$

По-умолчанию, значения слотов инициализируются данной константой.

Спецификация СФ в новой редакции включает следующие F-выражения:

- 1) математические операции: *neg, add, sub, mul, div, mod, rem, pow, exp, log, root2, root3*;
- 2) операторы сравнения: *below, below_or_equals, above, above_or_equals, equals, is_positive, is_negative*;
- 3) логические конструкции: *and, or, xor, not*;
- 4) предикаты для проверки типов: *is_number, is_string, is_array, is_boolean, is_strict_boolean, is_undefined*;
- 5) операции над контейнерами: *array, size, nth, is_empty, has*;
- 6) операции над множествами: *union, intersection, difference, symmetric_difference, contains*;
- 7) условные конструкции: *if*.

Заключение

Модель СФ претендует на статус универсальной системы представления знаний, включающей в себя свойства и возможности других наиболее распространенных нотаций. Насколько данное решение окажется приемлемым для разработчиков программного обеспечения, получив необходимую поддержку и стимул к дальнейшему развитию, зависит от множества факторов, в частности:

- 1) всесторонней продуманности модели, гибкости ее архитектуры и способности к расширению;
- 2) наличия исчерпывающей документации;
- 3) возможности и заинтересованности в стандартизации;
- 4) наличия достаточного количества примеров успешного применения и внедрения.

Если проанализировать текущую ситуацию на рынке коммерческого и в секторе свободного программного обеспечения, то можно заметить, что преобладающим является стремление к in-house разработке. Каждый год появляются новые бесполезные ЯП, форматы и протоколы, дистрибутивы операционных систем, библиотеки и фрейворки, дублирующие функциональность уже существующих решений. Это происходит потому, что особо консервативно настроенные компании, подверженные синдрому НИ (Not Invented Here [13]) предпочитают разрабатывать самостоятельно не только бизнес-логику, но и все сопутствующие технологии, тем самым пытаясь снизить риски, повысить конкурентно-способность и удержать финансовый приток от дочерних и партнерских фирм, использующих их решения. Что касается open-source сообщества, то оно страдает от так называемого «holy wars», неспособности к принятию единогласных решений и организованной разработке.

Таким образом, в условиях текущей ситуации на рынке коммерческого и в секторе свободного программного обеспечения маловероятно, что система представления знаний, основанная на СФ, получит должное внимание и тем более широкое распространение.

ЛИТЕРАТУРА

1. Artificial neural network [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Artificial_neural_network. – Date of access: 20.09.2017.
2. First-order logic [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/First-order_logic. – Date of access: 20.09.2017.
3. Semantic network [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Semantic_network. – Date of access: 20.09.2017.
4. Frame (artificial intelligence) [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: [https://en.wikipedia.org/wiki/Frame_\(artificial_intelligence\)](https://en.wikipedia.org/wiki/Frame_(artificial_intelligence)). – Date of access: 20.09.2017.
5. Function (mathematics) [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: [https://en.wikipedia.org/wiki/Function_\(mathematics\)](https://en.wikipedia.org/wiki/Function_(mathematics)). – Date of access: 24.09.2017.
6. Extended Backus–Naur form [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Extended_Backus–Naur_form. – Date of access: 24.09.2017.
7. M-expression [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: <https://en.wikipedia.org/wiki/M-expression>. – Date of access: 24.09.2017.
8. S-expression [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: <https://en.wikipedia.org/wiki/S-expression>. – Date of access: 24.09.2017.
9. Regular expression [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Regular_expression. – Date of access: 24.09.2017.
10. Abstract syntax tree [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Abstract_syntax_tree. – Date of access: 24.09.2017.
11. Boolean algebra [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Boolean_algebra. – Date of access: 25.09.2017.
12. Three-valued logic [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Three-valued_logic. – Date of access: 25.09.2017.
13. Not invented here [Electronic resource] // Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Not_invented_here. – Date of access: 25.09.2017.