УДК 004.822:514

АЛГОРИТМЫ СЕМАНТИЧЕСКОГО ПРОТОКОЛИРОВАНИЯ ПРОЦЕССОВ ОБРАБОТКИ ЗНАНИЙ

В.П. Ивашенко
Доцент кафедры интеллектуальных информационных технологий, кандидат технических наук

Белорусский государственный университет информатики и радиоэлектроники, Республика Беларусь
E-mail: ivashenko@bsuir.by

Аннотация. В работе рассматриваются алгоритмы, обеспечивающие семантическое протоколирование процессов обработки знаний в виде однородных семантических сетей модели унифицированного семантического представления знаний. Рассмотрены алгоритмы формирования семантического протокола и поиска для линейно и частично упорядоченных событий.

Ключевые слова: темпоральные отношения, базы знаний, процессы обработки знаний, однородные семантические сети, семантическое протоколирование.

Введение. В системах искусственного интеллекта одним из важных качеств является умение объяснить полученное решение, выразив его на языке, что является одним из признаков сознания [1]. Наличие соответствующего качества обеспечивает систему интроспективными возможностями с целью её совершенствования. В основе получения решений лежат процессы обработки знаний, поэтому важно рассмотреть сопутствующие механизмы, которые способны обеспечить возможность объяснения решений, например путём протоколирования соответствующих процессов. В работе предлагается использовать семантическое протоколирование процессов обработки знаний с целью выявления временных (темпоральных) отношений [2]. Семантическое протоколирование заключается в записи на языке представления знаний, например в виде семантической сети, знаний о порядке выполняемых действий и событий в процессах обработки знаний. При этом следует учитывать НЕ-факторы знаний [3], включая их неполноту, неопределённость и гипотетичность [4] и требования к работе в реальном масштабе времени [5].

В основу подхода положена онтологическая модель [6,7] событий и явлений [8]. Каждое явление рассматривается как модель, заданная на множестве (элементарных) событий, связанных бинарным отношением становления. Частными видами явлений являются: ситуация [9], движение, взаимодействие, действие, система. Между явлениями рассматриваются такие отношения как воздействие, процесс, состояние и другие [8]. Знания и их обработка могут рассматриваться как особый вид явлений, обладающих некоторой структурой. Все эти явления и события записываются на языке представления знаний в рамках модели унифицированного семантического представления знаний [4].

Семантическое протоколирование процессов обработки знаний

Тексты используемых языков соответствуют теоретико-множественной модели [10,11], которая задаётся (конечным) алфавитом, а сами языки L задаются как:

$$L \subseteq A^{**}, \tag{1}$$

где: A^{**} задано иерархией:

$$A^{*1} = A^{*};\ A^{*i+1} = \left(A^{*i} \cup A\right)^{*};\ A^{**} = \bigcup_{i=0}^{\infty}\left(A^{*i}\right). \tag{2}$$

Семантика языков модели унифицированного семантического представления знаний основана на модели ситуативных множеств [4]. Модель ситуативных (событийных) множеств может быть задана следующей шестёркой компонентов:

$$\langle U,[0;1],E,r,g,\mu\rangle;\ \mu \subseteq 2^{E} \times \left(2^{U\times\left(Z\times\left(Z^{E}\right)\right)}\right);\ Z = \bigcup_{q}^{q\in(\square/\{0\})}[0;1]^{\left([0;1]^{q-1}\right)}, \tag{3}$$

где: U – универсальное множество объектов предметной области, E – множество элементарных событий, r – отношение доступности (следования во времени) событий, g – функция, задающая множество событий существования каждого элемента универсального множества, μ – семейство пар множеств событий существования ситуативного множества и соответствий (нечёткой) ситуативной принадлежности элементов универсального множества ситуативному множеству, отображающих элементы ситуативных множеств, множества событий и соответствующие им наборы степеней нечёткой принадлежности высших порядков на множество степеней нечёткой принадлежности.

В системах обработки знаний происходят события, соответствующие установлению и удалению знаков информационных конструкций. Процессы в системах обработки знаний, построенных на основе модели унифицированного семантического представления знаний, способны реагировать на эти события с помощью механизмов пре- и пост- уведомлений. Первые позволяют предотвратить уничтожение или установление нежелательного события, вторые – обработать результат. Каждое событие или явление в соответствии с семантикой ситуативных множеств может быть представлено в виде соответствующего знака-элемента семантической сети.

Над событиями и явлениями задано (конечное) множество отношений в рамках некоторой онтологии. В онтологию входят понятия события и феномена и отношения между ними [8]. Семантический протокол [12] – это конструкция, которая представляется средствами модели унифицированного семантического представления знаний [4].

В случае линейно-упорядоченных отношением “раньше” событий для того, чтобы зафиксировать какое из двух событий произошло раньше, достаточно проиндексировать все события в порядке их следования и сравнить их индексы. Для того, чтобы найти события, которые произошли раньше указанного события, достаточно иметь соответствующий список событий (последовательный протокол).

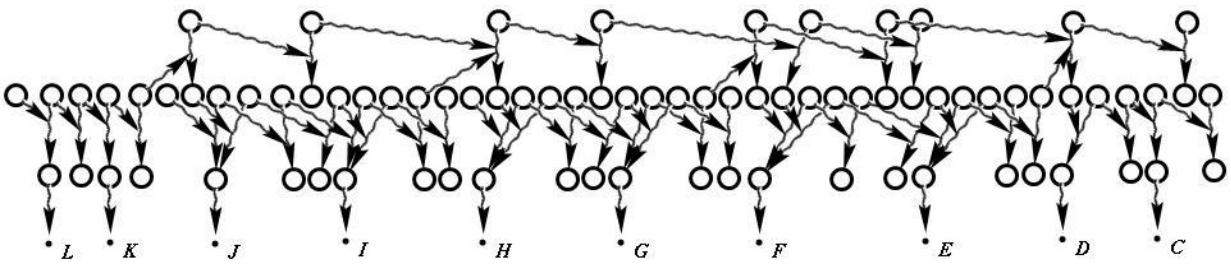

Рисунок 1. Онтологическая структура семантического протокола (C, D,… L – события)

Однако для быстрого поиска событий между указанными событиями (поиска по индексу), требуются более сложные структуры. Примером такой структуры является триплетно-свободный биноминальный стек (рис. 1), состоящий из звеньев (рис. 2), для которого известен алгоритм добавления [12].

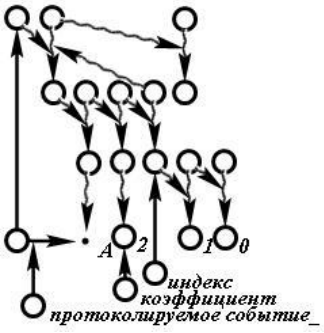

Рисунок 2. Протоколируемое событие A и представление звена семантического протокола, хранящего коэффициент, индекс, следующее звено

Для того чтобы проверить является ли одно событие раньше другого, сравниваются их индексы и выполняется поиск события с меньшим индексом в протоколе события с большим индексом (рис. 3).

$$
\begin{aligned}
&\langle c, l, r\rangle \leftarrow \\
&if\left(getIndex(l) > getIndex(r)\right) then \\
&\left|swap\left(\langle l, r\rangle\right)\right. \\
&\leftarrow getByIndex\left(\left\langle getProtocol(r), getIndex(l)\right\rangle\right)
\end{aligned}
$$

Рисунок 3. Алгоритм проверки раньше ли одно событие другого

Процедуры *getIndex* и *getProtocol* являются процедурами получения индекса и протокола события. Процедура *swap* выполняет обмен значений переменных.

Алгоритмы подсчёта количества элементов в стеке, поиска события по номеру в стеке и индексированного поиска события в стеке, которые позволяют реализовать получение ответов на основные запросы к семантическому протоколу. Алгоритм (Рис. 4) подсчёта количества элементов (*volume*) получает на вход стек (протокол) и возвращает количество элементов в нём.

$$
\begin{aligned}
&p \leftarrow \\
&a \leftarrow 1 \\
&s \leftarrow a \\
&l \leftarrow getFirstLink(p) \\
&while\left(\neg last\left(\langle p, l\rangle\right)\right) \\
&\left|\begin{aligned}
&if\left(coefficient\left(\langle p, l\rangle\right) > 1\right) then \\
&\left|a \leftarrow a + a\right. \\
&s \leftarrow s + a \\
&l \leftarrow getNextLink\left(\langle p, l\rangle\right)
\end{aligned}\right. \\
&\leftarrow s
\end{aligned}
$$

Рисунок 4. Алгоритм подсчёта количества элементов

Алгоритмы поиска получают стек и номер или индекс искомого элемента. Алгоритм поиска события по номеру возвращает номер ближайшего по номеру элемента и сам элемент (рис. 5). Алгоритм поиска события по индексу (*getByIndex*) возвращает номер найденного элемента и найденный элемент (рис. 6).

Процедуры *getFirstLink, getNextLink, getDownLink, coefficient, value, index* являются соответственно процедурами получения первого, следующего, нижнего звеньев в стеке, коэффициента, значения и индекса в звене. Процедура *last* проверяет является ли звено последним в стеке.

Чтобы найти все события, которые находятся позже указанного события достаточно запомнить вершины всех стеков, найти стеки, в которых находится указанное событие, его номер в каждом стеке и осуществить поиск соответствующих событий по номерам.

```
⟨p,i⟩ ←
a ← 1
s ← a
l ← getFirstLink(p)
while(s ≤ volume(p))
|if (coefficient(⟨p,l⟩) > 1) then
||a ← a + a
|if (s < i) then
||s ← s + a
||l ← getNextLink(⟨p,l⟩)
|else
||if (a > 1) then
|||a ← 1
|||l ← getDownLink(⟨p,l⟩)
||else
|||break
← ⟨s, value(⟨p,l⟩)⟩
```

Рисунок 5. Алгоритм поиска события в последовательном протоколе по номеру

```
⟨p,i⟩ ←
a ← 1
s ← a
l ← getFirstLink(p)
while(i ≤ index(⟨p,l⟩))
|if (coefficient(⟨p,l⟩) > 1) then
||a ← a + a
|if (i < index(⟨p,l⟩)) then
||if (last(⟨p,l⟩)) then
|||break
||else
|||s ← s + a
|||l ← getNextLink(⟨p,l⟩)
|else
||if (a > 1) then
|||a ← 1
|||l ← getDownLink(⟨p,l⟩)
||else
|||break
← ⟨s, value(⟨p,l⟩)⟩
```

Рисунок 6. Алгоритм поиска события в последовательном протоколе по индексу

Для событий, частично-упорядоченных отношением «раньше» (рис. 7), используется распределённый семантический протокол, содержащий комбинированные идентификаторы, каждый из которых соответствует набору бинарных идентификаторов базовых событий (первопричин).

$$\langle c,l,r\rangle \leftarrow$$
$$\leftarrow newCID\Big(\big\langle c, combine\big(\big\langle getCID\big(\langle c,l\rangle\big), getCID\big(\langle c,r\rangle\big)\big\rangle\big)\big\rangle\Big)$$

Рисунок 7. Алгоритм формирования события на основе двух исходных

Процедуры *newCID* и *getCID* формирования события на основе (нового) и получения комбинированного идентификатора в рамках некоторого контекста. Процедура формирования события на основе (нового) комбинированного идентификатора проверяет: есть ли уже такой идентификатор, и, если он есть, возвращает ассоциированное с ним и дополнительными структурами событие. К дополнительным структурам могут относиться: стек для последовательного протокола, индекс для этого стека и т.п.

$\langle c,l,r\rangle \leftarrow$
$s \leftarrow \varnothing$
$if\ \big(getSize(l) > getSize(r)\big) then$
$\quad swap\big(\langle l,r\rangle\big)$
$ei \leftarrow getMinor(l)$
$ej \leftarrow getMinor(r)$
$while\big(in\big(\langle r,ei\rangle\big)\big)$
$\quad \langle s,r,ej,c\rangle \leftarrow find\big(\langle s,r,ej,l,ei,c\rangle\big)$
$\quad if\ \big(match\big(\langle c,r,l,ei\rangle\big) \wedge (ei = ej)\big) then$
$\quad\quad if\ \big(last\big(\langle l,ei\rangle\big) \wedge last\big(\langle r,ej\rangle\big)\big) then$
$\quad\quad break$
$\quad\quad if\ \big(inside\big(\langle l,ei\rangle\big) \wedge inside\big(\langle r,ej\rangle\big)\big)$
$\quad\quad then$
$\quad\quad ej \leftarrow sequent\big(\langle ej,r,l\rangle\big)$
$\quad\quad ei \leftarrow sequent\big(\langle ei,l,r\rangle\big)$
$\quad\quad if\ \big(\neg in\big(\langle r,ei\rangle\big)\big) then$
$\quad\quad\quad if\ \big(empty(s)\big) then$
$\quad\quad\quad break$
$\quad\quad\quad else \langle s,\langle r,ej,l,ei\rangle\rangle \leftarrow pop(s)$
$\leftarrow match\big(\langle c,r,l,ei\rangle\big)$

Рисунок 8. Алгоритм проверки раньше ли одно событие другого

$\langle s,r,ej,l,ei,c\rangle \leftarrow$
$while\big(ej < ei\big)$
$\quad if\ \big(getSize(r) \le 2\big) then$
$\quad\quad break$
$\quad t \leftarrow getRight(r)$
$\quad r \leftarrow getLeft(r)$
$\quad if\ \big(match\big(\langle c,r,l,ei\rangle\big)\big) then$
$\quad\quad if\ \big(match\big(\langle c,t,l,ei\rangle\big)\big) then$
$\quad\quad\quad s \leftarrow push\big(\langle s,\langle t,ej,l,ei\rangle\rangle\big)$
$\quad else$
$\quad\quad r \leftarrow t$
$\quad\quad if\ \big(\neg in\big(\langle r,ej\rangle\big)\big) then$
$\quad\quad\quad if\ \big(match\big(\langle c,r,l,ei\rangle\big) \wedge inside\big(\langle r,ej\rangle\big)\big) then$
$\quad\quad\quad\quad ej \leftarrow sequent\big(\langle ej,r,l\rangle\big)$
$\quad\quad\quad else$
$\quad\quad\quad\quad break$
$\leftarrow \langle s,r,ej,c\rangle$

Рисунок 9. Алгоритм поиска события по идентификатору базового события

Вариант алгоритма построения комбинированного идентификатора (*combine*) на основе

идентификаторов исходных событий получает на вход два идентификатора и возвращает их комбинацию, являющуюся парой, содержащей упорядоченное множество и количество элементов в нём («длину» комбинированного идентификатора). Алгоритм проверки раньше ли одно событие другого для частично-упорядоченных отношением «раньше» событий (рис. 8 и рис. 9) получает на вход контекст и два события и возвращает результат их сравнения процедурой *match* до идентификатора базового события. Процедура *getMinor* возвращает (наименьший) идентификатор базового события из комбинированного идентификатора, процедуры *last* и *inside* проверяют равенство или превышение идентификатора базового события наибольшим (последним) идентификатором базового события в комбинированном идентификаторе.

Процедуры *getSize*, *sequent,* соответственно получают длину комбинированного идентификатора и следующий больший идентификатор базового события, а процедуры *in, match* соответственно проверяют: входит ли идентификатор базового события в комбинированный идентификатор, совпадают ли комбинированные идентификаторы до указанного идентификатора базового события в указанном контексте.

Процедуры *empty*, *push*, *pop* соответственно: проверяет является ли стек пустым, добавляет элемент в стек и извлекает элемент из стека. Алгоритм поиска события по идентификатору базового события (*find*) получает на вход стек, по два комбинированных идентификатора и идентификатора соответствующих базовых событий, а также – контекст, и возвращает стек, комбинированный идентификатор, идентификатор базового события и контекст. Процедуры *getLeft, getRight* получают левое и правое непосредственно предшествующие события.

Для сокращения и амортизации времени работы алгоритма хорошим приёмом является использование запоминания (memoization) [13].

Заключение. Разработанные алгоритмы обеспечивают протоколирование и поиск связанных темпоральными отношениями событий и явлений и могут быть могут быть применены для протоколирования процессов обработки и событий в больших базах знаний. При условии не более чем логарифмически зависящего от размера базы знаний n времени доступа к элементам семантической сети для отношения «раньше» поиск событий осуществляется за полилогарифмическое время для последовательных протоколов, а оценка временной сложности итерации однократного обхода распределённых протоколов, структурно соответствующих направленному ациклическому графу, предполагается полилогарифмически-линейной $O\left(n * \log^{c}\left(n\right)\right)$.

Список литературы

[1]. McDermott, D. Mind and Mechanism. Cambridge (Mass), MIT Press. – 2001. – xv + 262 p.

[2]. Allen, J.F. Time and time again: the many ways to represent time / J.F. Allen // Intern. J. of Intelligent Systems. – 1991. – Vol. 6. – pp. 341–355.

[3]. Нариньяни А.С. НЕ-факторы: неточность и недоопределенность — различие и взаимосвязь // Изв. РАН. Сер. Теория и системы управления. – 2000. – №5. С. 44–56.

[4]. Ивашенко, В.П. Модели и алгоритмы интеграции знаний на основе однородных семантических сетей. Материалы Международной научн.-техн. Конференции OSTIS, 2015:Минск, Республика Беларусь, БГУИР 19-21 февраля 2015, 111–132 с.

[5]. Ивашенко, В.П. Принципы платформенной независимости и платформенной реализации OSTIS / В.П. Ивашенко, М.М. Татур // Открытые семантические технологии проектирования интеллектуальных систем (OSTIS 2016) : материалы Междунар. науч.-техн. конф., Минск, 18–20 февр. 2016 г. / Белорус. гос. ун-т информатики и радиоэлектроники ; редкол. : В.В. Голенков (отв. ред.) [и др.]. – Минск, 2016. – С. 145–150.

[6]. Голенков, В.В. Проектирование предприятий рецептурного производства на основе онтологий / В.В. Голенков [и др.] // Онтология проектирования. – 2017. – Т. 7, №2(24). – С. 123 – 144.

[7]. Ивашенко, В.П. Модели обработки информации в интеллектуальных системах, основанных на семантических технологиях / В.П. Ивашенко, А.С. Бельчиков, А.П. Еремеев // конф. «Информационные технологии и системы». – Минск: БГУИР, 2016. С. 106-107.

[8]. Ивашенко, В.П. Онтологическое моделирование причинно-следственных связей на основе событий / В. П. Ивашенко // Информационные технологии и системы 2017 (ИТС 2017) = Information Technologies and Systems 2017 (ITS 2017) : материалы междунар. науч. конф. (Республика Беларусь, Минск, 25 октября 2017 года) / редкол. : Л. Ю. Шилин [и др.]. – Минск : БГУИР, 2017. – С. 138 – 139.

[9]. Поспелов,Д.А. Ситуационное управление: теория и практика. : М.: Наука, 1986. — 288 с.

[10]. Маркус, С. Теоретико множественные модели языков/ / Перевод с англ. М. В. Арапова ; Под ред. Ю. А. Шрейдера. – Москва : Наука, 1970. – 332 с.

[11]. Ивашенко, В.П. От теоретико-множественных моделей к симплициальным моделям языков / В.П. Ивашенко // Карповские научные чтения : сб. науч. ст. / Белорус. гос. ун-т; редкол. : А.И. Головня (отв. ред.) [и др.]. – Минск, 2016. – Вып. 10, ч. 1. – С. 248–253.

[12]. Ивашенко, В.П. Семантическое протоколирование процессов обработки знаний / В. П. Ивашенко // Ин-формационные технологии и системы 2017 (ИТС 2017) = Information Technologies and Systems 2017 (ITS 2017) : материалы междунар. науч. конф. (Республика Беларусь, Минск, 25 октября 2017 года) / редкол. : Л. Ю. Шилин [и др.]. – Минск : БГУИР, 2017. – С. 110 – 111.

[13]. Norvig, P., "Techniques for Automatic Memoization with Applications to Context-Free Parsing," Computational Linguistics, March 1991. – Vol. 17 No. 1. – pp. 91–98.

ALGORITHMS FOR SEMANTIC LOGGING OF KNOWLEDGE PROCESSING

V.P. IVASHENKO, PhD
Associate Professor of the Department of
Intelligent Information Technologies

Belarusian State University of Informatics and Radioelectronics, Republic of Belarus
E-mail: ivashenko@bsuir.by

Abstract. The paper considers algorithms that provide semantic logging of knowledge processing based on homogeneous semantic network of the unified semantic knowledge representation model. These algorithms provide semantic protocol construction and information retrieval from partially and linearly ordered sets of events.

Key words: temporal relations, knowledge bases, knowledge processing, homogeneous semantic networks, semantic logging