УДК 004.825

РЕАЛИЗАЦИЯ СОБЫТИЙНОГО УПРАВЛЕНИЯ ЗНАНИЯМИ В МОДЕЛЯХ, ПРЕДСТАВЛЕННЫХ ОБЪЕКТНО-ОРИЕНТИРОВАННЫМИ СЕМАНТИЧЕСКИМИ ГИПЕРГРАФАМИ

Починский И.А., Зинкин С.А.

ФГБОУ ВПО «Пензенский государственный университет», Пенза, e-mail: i.pochinskiy@yandex.ru

В статье приводится пример формализации последовательной ситуации, заключающейся в совершении клиентом заказа новой услуги у провайдера. Для декомпозиции ситуации используется концепция событийного подхода, описанного ранее. Каждое событие связывается с одним или несколькими продукционными правилами, которые добавляются к фронту готовых продукций при возникновении события. В качестве условий применимости продукций предлагается использовать расширение активаторов логико-трансформационных правил, предоставляющее возможность запуска продукционного правила по завершении определенного события или продукции. Кроме того, введены понятия семантически-дизъюнктивного и семантически-конъюнктивного списков, которые могут использоваться в качестве значений свойств объектов любого класса в пределах семантического гиперграфа. Семантически-дизъюнктивным списком является множество значений, один или несколько элементов которого могут быть использованы как значение свойства. Семантически-конъюнктивный список – множество, каждый из элементов которого является значением свойства.

Ключевые слова: процедурные знания, продукционные правила, семантические гиперграфы, события, ситуация, интеллектуальные системы управления знаниями (ИСУЗ), онтологии, событийный подход к управлению знаниями

IMPLEMENTATION OF EVENT-BASED KNOWLEDGE CONTROL IN MODELS REPRESENTED BY OBJECT-ORIENTED SEMANTIC HYPERGRAPH

Pochinskiy I.A., Zinkin S.A.

Penza State University, Penza, e-mail: i.pochinskiy@yandex.ru

In the article there is provided formalization example of the serial situation that consists of ordering new service by the client from his ISP. To decomposition that situation event-based conception described earlier is used. Every event connects with one or more production rules that are added to the ready productions forefront when the event occurs. Applying of logical transformation rules activators extension is proposed as the production rule application condition. It provides the possibility of running a production rule right after ending of some event or another production rule. Moreover new concepts of semantically conjunctive and semantically disjunctive lists are introduced. This lists could be used as values of any semantic hypergraph class object property. Semantically disjunctive list is a set, in which one or more elements could be interpreted as a property value. Semantically conjunctive list is a set, in which every element is value of the property.

Keywords: procedural knowledge, semantic hypergraph, production rules, events, situations, intellectual knowledge management systems (IKMS), ontology, event-based knowledge control

В ряде предыдущих работ [2, 3, 6] был исследован семантический гиперграф как формализм представления декларативных знаний, а также аргументирована необходимость представления процедурных знаний в интеллектуальных системах управления знаниями. В работе [5] был предложен событийный подход к представлению процедурных знаний и управлению онтологиями. В данной работе приводится пример реализации событийного подхода к управлению системой продукционных правил для формализации процедурных знаний моделируемой системы.

В качестве значений свойств события могут быть использованы ссылки на один или несколько классов или их экземпляров. В таком случае семантическая связь этого свойства будет инцидента каждой вершине (или одной из вершин), указанной в его (свойства) значении. Графически обозначать такую семантической дугу предлагается с помощью контура вокруг всех вершин, указанных в значении свойства, к которому направлена стрелка от вершины-события (свойство которого и рассматривается), причем в месте соприкосновения контура и стрелки изображать окружность с вписанным в нее символом конъюнкции (дизъюнкции). В качестве символьной записи предлагается использовать следующую конструкцию: $\{\&: v_1, v_2, ..., v_k\}$, где $v_1, v_2, ..., v_k$ – множество вершин, записанных в значении свойства. Такие семантические дуги предлагается называть *событийно-конъюнктивными* (*событийно-дизъюнктивными*), а множества – *событийно-конъюнктивными* (*событийно-дизъюнктивными*) *списками*.

Событийно-конъюнктивные и событийно-дизъюнктивные дуги, являясь средствами представления недоопределенной, неполной, неточной информации, повышают выразительные возможности ИСУЗ [7].

Рассмотрим пример, основывающийся на следующей последовательной фактической ситуации с каузально связанными событиями.

Клиент заказывает у провайдера услугу предоставления доступа к SIP-телефонии, но

на его лицевом счету недостаточно средств для оплаты этой услуги, поэтому доступ к ней блокируется. После оплаты клиентом стоимости услуги блокировка снимается, и клиент совершает исходящий звонок.

Для описания формализации последовательной событийной ситуации будет использоваться модель декларативных знаний, полученная в [4].

При допущении, что локальная сеть клиента подключена только к одному провайдеру, ее аппаратное обеспечение поддерживает услугу SIP-телефонии, а биллинговая система провайдера предоставляет возможность только авансового метода расчета, обобщенный набор событий, возникающих в системе, представлен ниже.

1. Заказ клиентом у провайдера услуги SIP-телефонии:

Объект: экземпляр класса *service:SIP*, семантически связанный с экземпляром класса *isp*, который в свою очередь семантически связан с экземпляром класса *client*, ассоциированным с клиентом Ивановым.

Субъект: экземпляр класса *client*, ассоциированный с клиентом Ивановым.

Правила:

a) семантическое связывание объекта и субъекта семантической дугой has_service.

2. Изменение прав доступа для клиента на оборудовании провайдера:

Объект: экземпляр класса *interface*, ассоциированный с портом доступа на оборудовании провайдера, к которому подключен клиент Иванов.

Субъект: объект класса *isp*, ассоциированный с провайдером, услугами которого пользуется клиент Иванов.

Правила:

a) изменение свойства destination объекта на trunk;

b) добавление к значению свойства service объекта ссылки на экземпляр класса service:SIP (объект события 1).

3. Конфигурирование абонентского оборудования:

Объект: 2 экземпляра класса *interface*, ассоциированных с интерфейсами пограничного устройства локальной сети клиента Иванова, подключенных к порту доступа оборудования провайдера и к SIP-телефону клиента Иванова.

Субъект: экземпляр класса *client*, ассоциированный с клиентом Ивановым.

Правила:

a) изменение свойства destination объекта 1 (ассоциированного с интерфейсом пограничного устройства локальной сети клиента Иванова, подключенным к оборудованию провайдера) на trunk;

b) изменение свойства state объекта 2 (ассоциированного с интерфейсом пограничного устройства локальной сети клиента Иванова, подключенным к SIP-телефону) на active;

c) изменение свойства *destination* объекта 2 на *clientSIP*.

4. Списание провайдером оплаты за услугу с лицевого счета клиента:

Объект: свойство *personal_account* экземпляра класса *client*, ассоциированного с клиентом Ивановым.

Субъект: объект класса *isp*, ассоциированный с провайдером, услугами которого пользуется клиент Иванов.

Правила:

a) изменение объекта в соответствии со значением свойства cost экземпляра класса service:SIP, ассоциированного с услугой, заказанной клиентом Ивановым.

4. Блокирование доступа клиента к сети передачи данных:

Объект: экземпляр класса *interface*, ассоциированный с портом доступа на оборудовании провайдера, к которому подключен клиент Иванов.

Субъект: объект класса *isp*, ассоциированный с провайдером, услугами которого пользуется клиент Иванов.

Правила:

a) если значение свойства personal_account экземпляра класса client, ассоциированного с клиентом Ивановым, меньше 0, то изменение свойства state объекта на blocked.

6. Зачисление денег на лицевой счет клиента:

Объект: свойство *personal_account* экземпляра класса *client*, ассоциированного с клиентом Ивановым.

Субъект: экземпляр класса *client*, ассоциированный с клиентом Ивановым.

Правила:

a) изменение объекта в соответствии с текущим и целевым его значениями.

7. Предоставление доступа клиента к сети передачи данных:

Объект: экземпляр класса *interface*, ассоциированный с портом доступа на оборудовании провайдера, к которому подключен клиент Иванов.

Субъект: объект класса *isp*, ассоциированный с провайдером, услугами которого пользуется клиент Иванов.

Правила:

a) изменение свойства state объекта на active.

8. Исходящий SIP-звонок клиента:

Объект: экземпляр класса *service:SIP*, семантически связанный с экземпляром класса *isp*, который в свою очередь семантически связан с экземпляром класса *client*, ассоциированным с клиентом Ивановым.

Субъект: экземпляр класса *client*, ассоциированный с клиентом Ивановым.

Правила:

а) изменение свойства personal_account субъекта в соответствии со значением свойства cost_call объекта.

Для события 1 нет необходимости указывать в качестве одного из дополнительных параметров провайдера, у которого заказывается услуга, т.к. заранее было оговорено, что клиент пользуется услугами только одного провайдера, в противном случае необходимо было бы предусмотреть процедуру определения провайдера, услугу у которого заказал клиент.

Для события 3 объектом являются 2 вершины-экземпляра класса *interface*. Формально объект события 3 представляется как *семантически-конъюнктивный список*.

Формально создание событий в терминах семантических гиперграфов выглядит следующим образом. Сначала необходимо создать сам класс *event*, экземпляры которого и будут являться событиями, описанными выше.

$$K \xleftarrow{+} event = \begin{Bmatrix} identified\_by : id \\ evType : type; \\ evName : name; \\ evObject : object; \\ evSubject : subject \end{Bmatrix};$$

$$S(event :: id) = 46.$$

Ниже приведена формальная запись создания экземпляра класса *event* для первого события рассматриваемой ситуации. В качестве значения свойства *evType* используется название ситуации, в которую входит событие (в случае если событие входит в несколько ситуаций, значением данного свойства будет являться событийно-конъюнктивный список, элементами которого будут являться названия каждой из ситуаций).

$$V \overset{instance}{+\!\leftarrow} event : ev_1;$$

$$S(event : ev_1 :: id) = 46;$$

$$S(event : ev_1 :: evType) = "client\ Order\ Service\ Sip\ With\ outBalance";$$

$$S(event : ev_1 :: evName) = "client\ Order\ Service\ Sip";$$

$$S(event : ev_1 :: evObject) = object_1;$$

$$S(event : ev_1 :: evSubject) = subject_1.$$

По аналогии с вышеприведенным вариантом создаются остальные события.

После создания всех экземпляров класса *event* для задания порядка возникновения событий в модели необходимо произвести каузальное связывание созданных экземпляров.

Первое событие является начальным в рассматриваемой ситуации, причиной его возникновения могут являться события других ситуаций. Поскольку в данной работе они не рассматриваются, будем использовать это событие как стартовое. Для возникновения второго и третьего событий необходимо возникновение первого. Для возникновения четвертого события необходимо возникновение второго события. Для возникновения пятого события необходимо отрицательное значение объекта четвертого события. Представление такой связи невозможно с помощью каузальных дуг (реализовать это можно в ядре правил продукций или с помощью алгоритмических решений при проектировании ИСУЗ), поэтому будем считать, что для возникновения пятого события необходимо возникновение четвертого, а обработку логического условия возложим на правило продукции, связанное с пятым событием. Для возникновения шестого события необходимо возникновение пятого. По аналогии с причиной возникновения пятого события для возникновения седьмого необходимо возникновение шестого события. Для возникновения восьмого события необходимо возникновение второго, третьего и седьмого событий. Формально это записывается следующим образом:

$$E \xleftarrow{+} causeArc;$$

$$r(causeArc) = \left\{ev_2, \dot{ev}_4\right\};$$

$$r(causeArc) = \left\{ev_5, \dot{ev}_6\right\};$$

$$r(causeArc) = \left\{ev_2, ev_3, ev_7, \dot{ev}_8\right\};$$

$$r(causeArc) = \left\{ev_1, \dot{ev}_3\right\};$$

$$r(causeArc) = \left\{ev_4, \dot{ev}_5\right\};$$

$$r(causeArc) = \left\{ev_7, \dot{ev}_6\right\}.$$

Ниже формально записан спектр продукций с правилами модификации исходной модели. В качестве сферы применения продукций выступает название события, при возникновении которого данная продукция будет отнесена к фронту готовых продукций. В качестве антецедент продукций используются условия существования обрабатываемых продукцией вершин. Кроме того, в продукциях использованы названия переменных $object_i$ и $subject_i$ с индексами, которые обозначают объект и субъект i-го события соответственно.

В качестве условий применимости продукций использовано расширение активаторов логико-трансформационных правил [1] – *ON_FINISHED()*, предоставляющее возможность запуска продукционного правила по завершении события или продукции, указанных в скобках этого активатора.

В составе антецедент ядер продукций используется операция *exist()*, принимающая истинное значение в случае, если вершина, указанная в скобках, существует в модели, и ложное в противном случае.

$$
\begin{aligned}
&S: evType = "client\ Order\ Service\ Sip"\\
&L: True\\
&A: exist(object_1)\,\&\,exist(subject_1)\,\&\,(\exists x \in E)(x = has\_service)\\
&B_1: r(has\_service) \xleftarrow{+} \{object_1, \dot{subject}_1\}\\
&B_2: null\\
&Q: null\\
&\Pr: 0\\
&Ex: null
\end{aligned}
\qquad (i_1)
$$

$$
\begin{aligned}
&S: evType = "aclChangeByIsp"\\
&L: \_ON\ FINISHED\ (ev_1)\\
&A: exist(object_2)\,\&\,(\exists x \in S(object_2))(x = destination)\\
&B_1: S(object_2 :: destination) = "Trunk\ client"\\
&B_2: null\\
&Q: (i_3)\\
&\Pr: 0\\
&Ex: null
\end{aligned}
\qquad (i_2)
$$

$$
\begin{aligned}
&S: evType = "aclChangeByIsp"\\
&L: exist(object_2)\,\&\,((\exists x)x \in S(object_2), x = service)\,\&\\
&\&\,(\exists x \in service: SIP)(\exists y \in isp)\left(r(use\_provider) = \{client: \#1456, \dot{y}\}\right)\left(r(provide\_service) = \{\dot{y}, x\}\right)\\
&A: ON\_FINISHED(ev_1)\\
&B_1: S(object_2 :: service) \xleftarrow{+} object_2: service\\
&B_2: null\\
&Q: null\\
&\Pr: 0\\
&Ex: null
\end{aligned}
\qquad (i_3)
$$

$$
\begin{aligned}
&S: evType = "configDeviceByClient"\\
&L: exist\left(object_3[1]\right)\&\left(\exists x \in S\left(object_3[1]\right)\right)\left(x = destination\right)\\
&A: ON\_FINISHED\ \left(ev_2\right)\\
&B_1: S\left(object_3[1]:: destination\right) = "Trunk"\\
&B_2:\ null\\
&Q: \left(i_5\right)\\
&\text{Pr: } 0\\
&Ex: null
\end{aligned}
\qquad (i_4)
$$

$$
\begin{aligned}
&S: evType = "configDeviceByClient"\\
&L: exist\left(object_3[2]\right)\&\left(\exists x \in S\left(object_3[2]\right)\right)\left(x = state\right)\\
&A: ON\_FINISHED\ \left(ev_2\right)\\
&B_1: S\left(object_3[2]:: state\right) = "Active"\\
&B_2:\ null\\
&Q: \left(i_6\right)\\
&\text{Pr: } 1\\
&Ex: null
\end{aligned}
\qquad (i_5)
$$

$$
\begin{aligned}
&S: evType = "configDeviceByClient"\\
&L: exist\left(object_3[2]\right)\&\left(\exists x \in S\left(object_3[2]\right)\right)\left(x = destination\right)\\
&A: ON\_FINISHED\ \left(ev_2\right)\\
&B_1: S\left(object_3[2]:: destination\right) = "Client\ SIP"\\
&B_2:\ null\\
&Q:\ null\\
&\text{Pr: } 2\\
&Ex: null
\end{aligned}
\qquad (i_6)
$$

$$
\begin{aligned}
&S: evType = "saldoDecreaseByIsp"\\
&L: exist\left(object_4\right)\&\left(\exists x \in service: SIP\right)\left(r\left(has\_service\right) = \left\{client: \#1456, \dot{x}\right\}\right)\&\\
&\&\left(\exists y \in S\left(service: SIP\right)\right)\left(y = \cos t\right),\\
&A: ON\_FINISHED\ \left(ev_3\right)\\
&B_1: object_4 = object_4 - S\left(x :: \cos t\right)\\
&B_2:\ null\\
&Q:\ null\\
&\text{Pr: } 0\\
&Ex: null
\end{aligned}
\qquad (i_7)
$$

$S: evType = "accessToDtnBlockingByIsp"$

$L: exist(object_5) \& (\exists y \in S(client: \#1456))\left(r(accounts\_with) = \left\{client: \#1456, \dot{y}\right\}\right)$

$A: ON\_FINISHED(ev_4) \& client: \#1456 :: personal\_account < 0$

$B_1: S(object_5 :: state) = "Blocked"$ (i_8)

$B_2: null$

$Q: null$

Pr: 0

$Ex: null$

$S: evType = "saldoIncreaseByClient"$

$L: exist(object_6) \& (\exists y \in S(client: \#1456))\left(r(accounts\_with) = \left\{client: \#1456, \dot{y}\right\}\right)$

$A: ON\_FINISHED(i_8) \& client: \#1456 :: personal\_account < 0$

$B_1: object_6 = object_6 + C$ (i_9)

$B_2: null$

$Q: null$

Pr: 0

$Ex: null$

$S: evType = "accessToDtnUnblockingByIsp"$

$L: exist(object_7) \& (\exists x \in S(object_7))(x = state) \&$

$\& (\exists y \in S(client: \#1456))\left(r(accounts\_with) = \left\{client: \#1456, \dot{y}\right\}\right)$

$A: ON\_FINISHED(i_9) \& client: \#1456 :: personal\_account > 0$ (i_{10})

$B_1: S(object_7 :: state) = "Active"$

$B_2: null$

$Q: null$

Pr: 0

$Ex: null$

$S: evType = "sipCallOutput"$

$L: exist(object_8) \& exist(subject_8) \&$

$\& (\exists x \in S(object_7))(x = state) \& (\exists y \in S(subject_7))(y = personal\_account)$

$A: S(subject_8 :: personal\_account) > S(object_8 :: \cos t)$

$B_1: S(subject_8 :: personal\_account) = S(subject_8 :: personal\_account) - S(object_8 :: \cos t)$ (i_{11})

$B_2: null$

$Q: null$

Pr:0

$Ex: null$

Вывод

В данной работе приведен пример проектирования последовательной фактической системы, описанной онтологией сети провайдера, знания в которой представлены с помощью объектно-ориентированного семантического гиперграфа. Практически доказана относительная несложность применения правил продукционного вывода для управления динамикой семантики модели системы.

Список литературы

1. Искусственный интеллект. Справочник в трех томах / под ред. В.Н. Захарова, Э.В. По-пова, Д.А. Поспелова, В.Ф. Хорошевского. – М.: Радио и связь, 1990.

2. Починский И.А. Использование гиперграфов для представления онтологии сетевого оборудования // Проблемы информатики в образовании, управлении, экономике и технике: Сб. статей XI Междунар. научно-техн. конф. – Пенза: ПДЗ, 2011. – С. 74–78.

3. Починский И.А. Продукционные правила в качестве средства формализации семантического гиперграфа // Университетское образование: Сб. статей XVI Междунар. научно-метод. конф. – Пенза: Изд-во ПГУ, 2012. – С. 433–445.

4. Починский И.А. Проектирование онтологии сети передачи данных на основе формализма семантических гиперграфов // Труды Всероссийского конкурса научно-исследовательских работ студентов и аспирантов в области технических наук: материалы работ победителей и лауреатов конкурса. – СПб.: Изд-во Политехн. ун-та, 2012. – С. 139–141.

5. Починский И.А. Событийный подход к управлению семантикой системы / И.А. Починский, С.А. Зинкин // Фундаментальные исследования. – 2013. – № 6 (часть 4). – С. 863–866.

6. Починский И.А. Формальное представление семантических гиперграфов и операций над ними. // Молодежь. Наука. Инновации: Труды V междунар. научно-практ. интернет-конференции / под ред. Г.К. Сафаралиева, А.Н. Андреева, В.А. Казакова – Пенза: Изд-во Пензенского филиала ФГБОУ ВПО «РГУИТП», 2012. – С. 373–377.

7. Zadeh L. Commonsense knowledge representation based on fuzzy logic // Computer. – 1983. – Vol. 16. – C. 256–281.

References

1. Zaharov V.N., Popov E.V., Pospelov D.A., Horoshevkiy V.F. Iskusstvenny intellect [Artificial Intelligence], Moscow, Radio and connection, 1990.

2. Pochinskiy I.A. Sbornik statej 11 mezhdunarodnoj nauchno-tehnicheskoj konferencii «Problemy informatiki v obrazovanii, upravlenii, ekonomiki i tehnike», Penza, 2011, pp. 74–78.

3. Pochinskiy I.A. Sbornik statej 16 mezhdunarodnoj nauchno-metodicheskoy konferencii (Works of 16th Int. Scientific Technical Conference «Informatics problems in education, management, economic and technique»), Penza, 2012, pp. 443–445.

4. Pochinskiy I.A. Trudy Vserossijskogo konkursa nauchno-issledovatel'skih rabot studentov i aspirantov v oblasti tehnicheskih nauk: materialy rabot pobeditelej i laureatov konkursa (Proc. All-Russian competition of the science-research works of students and graduates in the technical science area: winners' and laureates' works). St-Petersburg, 2012, pp. 139–141.

5. Pochinskiy I.A., Zinkin S.A. Fundamental'nye issledovaniya – Fundamental researches, 2013, no.6 (part 4), pp. 863–866.

6. Pochinskiy I.A. Trudy 5 mezhdunarodnoj nauchno-prakticheskoy konferencii «Molodezh'. Nauka. Innovacii». (Proc. of the 5th Int. Scince and Practic Conference «Youth. Science. Innovations»), Penza, 2012, pp. 373–377.

7. Zadeh, L. Commonsense knowledge representation based on fuzzy logic // Computer. 1983. Vol. 16. pp. 256–281.

Рецензенты:

Егоров С.И., д.т.н., доцент, профессор кафедры вычислительной техники, ФГБОУ «Юго-Западный государственный университет», г. Курск;

Ромм Я.Е., д.т.н., профессор, заведующий кафедрой информатики, ФГБОУ ВПО «ТГПИ имени А.П. Чехова», г. Таганрог.

Работа поступила в редакцию 19.07.2013.