ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ

УДК 004.82
DOI: 10.18101/2304-5728-2018-3-40-59

МОДЕЛИРОВАНИЕ СИСТЕМЫ УПРАВЛЕНИЯ АКТИВАЦИЕЙ ПРОДУКЦИОННЫХ ПРАВИЛ С ИСПОЛЬЗОВАНИЕМ АВТОМАТНОГО ПРОГРАММИРОВАНИЯ

©Дармахеева Татьяна Анатольевна
ассистент,
Восточно-Сибирский государственный университет технологий и управления
Россия, 670013, г. Улан-Удэ, ул. Ключевская, 40В
E-mail: tania-shatyeva@mail.ru

© Хаптахаева Наталья Баясхалановна
кандидат технических наук, доцент,
начальник отдела подготовки кадров высшей квалификации,
Восточно-Сибирский государственный университет технологий и управления
Россия, 670013, г. Улан-Удэ, ул. Ключевская, 40В
E-mail: khapnb@gmail.com

В статье приведены результаты моделирования системы управления активацией систем продукций. Управление активацией систем продукций осуществляется за счет использования знаний о системе продукций в виде онтологии с активной семантикой. Поведение системы, реализующей управление активацией систем продукций на основе онтологии с активной семантикой, является сложным. Для моделирования такой системы применяется теория автоматов и технологии автоматного программирования. В результате чего получена автоматная модель системы управления активацией систем продукций и выполнена её программная реализация. Посредством полученного программного обеспечения проведено тестирование методов естественно-языковой обработки текста и извлечения знаний о терминах и отношениях между ними из словарных статей терминологического словаря. Методы представлены в виде систем продукций.
Ключевые слова: теория автоматов; онтология с активной семантикой; моделирование систем со сложным поведением; управление активацией систем продукций; система, основанная на знаниях; база знаний; система продукций.

Введение

Продукционные системы, основанные на правилах, являются важной технологией при создании информационных систем, называемых интеллектуальными системами с базами правил (или с базами знаний). Представление решения задачи в виде множества правил является более естественным способом по сравнению с алгоритмом. Управление вычислительными процессами в таких системах реализует процессор (модуль активации систем продукций), который может быть основан на метаправилах, специальном управляющем языке, либо на онтологии с активной семантикой как это представлено в данной статье [1, 2].

В статье рассматривается база знаний, содержащая продукционные правила, сгруппированные в системы продукций. Каждая отдельная система продукций реализует либо метод естественно-языковой обработки текста, либо метод извлечения знаний о терминах и отношениях между ними. Для управления активацией этих систем продукций в [1] разработан способ управления, суть которого заключается в обходе сети знаков-фреймов, образующих онтологию системы продукций. В результате движения по сети знаков-фреймов происходит вызов внутренних и внешних присоединенных процедур знаков-фреймов, что обеспечивает автоматическое решение следующих задач: выбор системы продукций для активации, построение рабочей памяти, соответствующей активной системе продукций, запуск машины логического вывода на активной системе продукций [1]. Знак-фрейм – это концептуальный объект одной из категорий: «Понятие», «Свойство», «Действие», «Величина», «Событие» и «Состояние» [2]. Внутренние присоединенные процедуры предназначены для чтения XML-документов, содержащих описание терминов онтологии, и отвечают за обход сети знаков-фреймов. К ним относятся процедуры осуществляющие разбор концептуальных объектов категорий «Понятие», «Действие», «Свойство», «Величина», «Состояние» и «Событие»: ReadConcept(), ReadAction(), ReadProperty(), ReadQuantity(), ReadState() и ReadEvent() соответственно. В процессе выполнения внутренних процедур запускаются внешние присоединенные процедуры, реализующие основные действия механизма управления системами продукций. Например, процедура Build_relations(), которая строит таблицы рабочей памяти в соответствии со структурой предикатов, где имя таблицы соответствует имени предиката, а поля — аргументам. Заполнение таблиц фактами, необходимыми для решения задачи, осуществляется посредством выполнения процедур лексического анализа текста Lex_analysis(), загрузки словарей Load_BF(), извлечения множество пар <основа+окончание> из лексем Extract_morphemes() и другими. Запуск машины логического вывода на полученных фактах рабочей памяти и заданной системе продукций выполняет внешняя присоединённая процедура Play_InferenceEngine(). Библиотека присоединенных процедуры модифицируется и пополняется в зависимости от решаемой задачи. Вызов внутренних и внешних присоединенных процедур выполняется том порядке, в котором данные процедуры встречаются в знаке-фрейме. Содержимое слотов знаков-фреймов зависит от реализуемого системой продукций метода. Последовательность чтения знаков-фреймов определяется онтологией системы продукций, которая для каждой системы продукций является уникальной.

В рассматриваемой статье исследуется вопрос разработки автоматной модели системы управления активацией систем продукций, реализующей описанный выше способ управления на основе онтологии с активной семантикой.

1. Моделирование систем со сложным поведением на основе автоматного программирования

Система управления активацией систем продукций (далее – СУ АСП) состоит из следующих компонент: интерфейс программы, подсистема чтения онтологии системы продукций, подсистема обработки знаков-фреймов, библиотека присоединенных процедур знаков-фреймов. Поведение данной системы описывается следующим образом:

1. Через интерфейс программы пользователем вводится словарная статья и выбирается метод естественно-языковой обработки текста или метод извлечения знаний.

2. На вход подсистемы обработки онтологии подается онтология системы продукций, соответствующая выбранному методу. В процессе чтения онтологии происходит обход сети знаков-фреймов. Обход всегда начинается со знака-фрейма категории «Понятия», являющимся центральным в сети. Если требуется активация другой системы продукций, то чтение текущей онтологии SP_i приостанавливается и осуществляется переход на чтение онтологии системы продукций SP_k. После окончания чтения онтологии SP_k, происходит возврат к чтению онтологии SP_j.

3. В каждом узле сети знаков-фреймов из подсистемы обработки знаков-фреймов вызываются соответствующие механизмы. Каждая категория знаков-фреймов имеет различную структуру и поэтому обрабатывается соответствующим образом. Например, если знак-фрейм относится к категории «Понятие», то необходимо вызвать конкретный механизм для его обработки. И так для каждой из перечисленных выше категорий концептуальных объектов.

4. Если в слоте знака-фрейма прописан вызов присоединенной процедуры, то происходит ее вызов. После выполнения присоединенной процедуры обработка знака-фрейма продолжается дальше, но уже после точки вызова процедуры. Результатом вызова процедуры может быть:

- переход на обработку другого знака-фрейма;
- активация другой системы продукций, что означает выполнение п.2;
- построение структуры рабочей памяти;
- заполнение таблиц рабочей памяти в результате лексического анализа текста или извлечения морфем;
- заполнение таблиц рабочей памяти в результате работы машины логического вывода.

Исходя из приведенного описания поведения СУ АСП, следует, что данная система обладает сложным поведением. Это подтверждается зависимостью выходных данных не только от значений входных параметров, но и от результатов вычислительных процессов внутри системы.

Для моделирования систем со сложным поведением в [3] предлагается автоматное программирование, которое и будет использоваться в настоящей работе. Суть данного программирования [3] заключается в представлении сущностей со сложным поведением в виде автоматизированных

объектов управления, которые состоят из управляемой и управляющей частей. Управляемая часть состоит из объекта управления, который отвечает за выполнение действий, выбранных для выполнения управляющей частью, и, возможно, за формирование некоторых компонентов входных воздействий для управляющей части – обратных связей. Управляющая часть отвечает за логику поведения – выбор выполняемых действий, зависящий от текущего состояния и входных воздействий, а также за переход в новое состояние. Управляющая часть представляет собой управляющий автомат или систему взаимодействующих автоматов. Формально автоматизированный объект управления – это пара вида [3]:

$$AOU =< A, O >, \quad (1)$$

где A — управляющий автомат, O — объект управления.

Управляющий автомат — это пятерка вида:

$$A =< X, Y, Z, y_0, \varphi, \delta >, \quad (2)$$

где $X = X_E \times X_O$ — конечное множество входных воздействий, причем каждое входное воздействие x состоит из компоненты x_E, порождаемой внешней средой, и компоненты x_O, порождаемой объектом управления;

Y — конечное множество управляющих состояний;

Z — конечное множество выходных воздействий;

$y_0 \in Y$ — начальное состояние;

$\varphi = \varphi' \times \varphi''$ — функция выходов (выходных воздействий), состоящая, в общем случае, из двух компонент: функции выходных воздействий в состояниях φ': $Y \to Z$ и функции выходных воздействий на переходах φ'': $X \times Y \to Z$;

δ: $X \times Y \to Y$ — функция переходов.

Объект управления – это тройка вида:

$$O =< V, f_c, f_c >, \quad (3)$$

где V — потенциально бесконечное множество вычислительных состояний (или значений);

f_q: $V \to X_O$ — функция, сопоставляющая входное воздействие вычислительному состоянию;

f_c: $Z \times V \to V$ — функция, изменяющая вычислительное состояние в зависимости от выходного воздействия.

Функции f_q и f_c являются математическими эквивалентами набора запросов и набора команд соответственно.

2. Автоматная модель СУ АСП

Таким образом, используя автоматную модель СУ АСП можно описать как множество взаимодействующих между собой автоматизированных объектов, которое имеет вид:

$$Control\_Activation\_SP = \{AOU_1, AOU_2, AOU_3, AOU_4, AOU_5, AOU_6, AOU_7\}. \quad (4)$$

Автоматизированный объект управления AOU_1, отвечает за чтение онтологии системы продукций и работу системы в целом имеет вид:

$$AOU_1 =< ReadOntlogy, < GUI, ControlOntology >>, \quad (5)$$

где *ReadOntology* — автомат, реализующий алгоритм обхода сети знаков-фреймов, образующих онтологию заданной пользователем системы продукций и работу интерфейса программы;

GUI — объект управления, отвечающий за интерфейс программы;

ControlOntology — объект управления, содержащий механизмы, требуемые для чтения онтологии системы продукций.

Управляющий автомат *ReadOntology* — это пятерка вида:

$$ReadOntology =< ont\_X, ont\_Y, ont\_Z, ont\_y_0, ont\_\varphi, ont\_\delta >, \quad (6)$$

где $ont\_X = X_E \times X_O = \{p1.e_1\} \times \{Gui.x_1\} = \{p1.e_1, Gui.x_1\}$ — множество входных воздействий: $p1.e_1$ – компонент, порождаемый поставщиком событий $p1$ и означающий событие «Система продукций выбрана пользователем»; $Gui.x_1$ — компонент, порождаемый объектом управления *Gui* и отвечающий на вопрос: «Найден ли центральный знак-фрейм онтологии системы продукций?».

$ont\_Y = \{ont\_y_0, ont\_y_1, ont\_y_2, ont\_y_3\}$ — конечное множество управляющих состояний:

ont_y_0 – «Получение входных данных»;
ont_y_1 – «Чтение онтологии СП»;
ont_y_2 – «Вызов автомата *ReadConcept*»;
ont_y_3 – «Активация системы продукций завершена».

$ont\_Z = \{Gui.z_1, Gui.z_2, Ontology.z_1\}$ — конечное множество выходных воздействий:

$Gui.z_1$ — «Загрузить интерфейс»;
$Gui.z_2$ — «Определить категорию знака-фрейма»;
$Ontology.z_1$ — «Прочитать онтологии системы продукций».

$ont\_y_0 \in ont\_Y$ — начальное состояние.

$ont\_\varphi = ont\_\varphi' \times ont\_\varphi''$ — функция выходов (выходных воздействий): ont_φ': $ont\_Y \rightarrow ont\_Z$ — функция выходных воздействий в состояниях:

$ont\_\varphi'1: y0 \rightarrow Gui.z1,$
$ont\_\varphi'2: y1 \rightarrow Gui.z2, Ontology.z1,$

ont_φ'3: y2→ReadConcept.

ont_φ'': ont_X×ont_Y→ont_Z — функция выходных воздействий на переходах, для данного автомата не существует.

*ont_*φ={(y_0→*Gui*.z_1), (y_1→(*Gui*.z_2, *Ontology*.z_1)), (y_2→*ReadConcept*))}, где *Gui*.z_1, *Gui*.z_2, *Ontology*.z_1 из *ont_Z*, *ReadConcept* – управляющий автомат, реализующий алгоритм разбора знаков-фреймов категории «Понятие».

*ont_*δ: *ont_X*×*ont_Y*→ *ont_Y* — функция переходов, которая представлена ниже в таблице 1.

Таблица 1.

Функция переходов *ont_*δ автомата *ReadOntology*

№	*ont_X*	*ont_Y*	*ont_X*×*ont_Y*→*ont_Y*
1	$p1.e_1$,*Gui*.x_1	*ont_*y_0	*ont_*y_1
2	$p1.e_1$,!*Gui*.x_1	*ont_*y_0	*ont_*y_3
3	$p2.e_1$	*ont_*y_1	*ont_*y_2
4	$p2.e_2$	*ont_*y_1	*ont_*y_3
5	$p2.e_3$	*ont_*y_2	*ont_*y_3

Поставщик событий *p*2 описывает события, возникающие в процессе чтения онтологии системы продукций. Данные события передаются автоматам *ReadOntology* и *ReadConcept*. Ниже, в таблице 2 представлено описание событий из *p*2.

Таблица 2.

События поставщика *p*2

Событие	Описание	Комментарий
e_1	Онтология инициализирована	Событие генерируется в случае, если для указанной пользователем системы продукций найден центральный знак-фрейм
e_2	Онтология не инициализирована	Событие генерируется в случае, если для указанной пользователем системы продукций не найден центральный знак фрейм
e_3	Процесс активации завершен	Событие генерируется, когда обход сети знаков-фреймов завершен
e_4	Следующий такт автомата	Это событие генерируется при переходе автомата из одного состояния в другое

Объект управления *GUI* — это тройка вида:

$$GUI =< gui\_V, gui\_f_q, gui\_f_c >, \quad (7)$$

Для данного объекта *gui_V* = {*gui_v*1, *gui_v*2}: *gui_v*1 — «Загружен интерфейс»; *gui_v*2 — «Получены входные данные».

*gui_*f_q: *gui_V*→X_O — функция, сопоставляющая входное воздействие вычислительному состоянию:

*gui_*f_{q1}: *gui_v*1→ «Начало работы системы»;

gui_f_{q2}: $gui\_v2 \rightarrow$ «Данные введены».

gui_f_c: $Y_1 \times gui\_V \rightarrow gui\_V$ — функция, изменяющая вычислительное состояние в зависимости от выходного воздействия:

gui_f_{c1}: $Gui.z1 \times gui\_v_1 \rightarrow gui\_v_2$.

Объект управления *ControlOntology* — это тройка вида:

$$ControlOntology =< ont\_V, ont\_f_q, ont\_f_c >, \quad (8)$$

где $ont\_V = \{ont\_v1, ont\_v2\}$: ont_v1 — «Онтология заданной системы продукций прочитана»; ont_v2 — «Заданная система продукций активирована».

ont_f_{q1}: $ont\_v1 \rightarrow$ «Центральный знак-фрейм заданной системы продукций найден»;

ont_f_{c1}: «Завершение работы системы» $\times$ $ont\_v_1 \rightarrow ont\_v_2$.

Автоматизированный объект управления, отвечающий за обработку знаков-фреймов категории «Понятие»:

$$AOU_2 =< ReadConcept, ControlConcept >, \quad (9)$$

ReadConcept — автомат, реализующий алгоритм разбора знаков-фреймов категории «Понятие»;

ControlConcept — объект управления, хранящий механизмы, стандартные — для обработки всех знаков-фреймов и специальные — для обработки знаков-фреймов категории «Понятие».

Управляющий автомат *ReadConcept* — это пятерка вида:

$$ReadConcept =< Con\_X, Con\_Y, Con\_Z, Con\_y_0, Con\_\varphi, Con\_\delta >, \quad (10)$$

где $Con\_X = \{(p2.e_4, ControlConcept.x_1), (p2.e_4, ControlConcept.x_2), (p2.e_4, ControlConcept.x_3), (p2.e_4, ControlConcept.x_4), (p2.e_4, ControlConcept.x_5), (p2.e_4, ControlConcept.x_6), (p2.e_4, ControlConcept.x_7)\}$ — множество входных воздействий:

$p2.e_4$ — компонент, порождаемый поставщиком событий $p2$ (см. выше в таблице 1);

$ControlConcept.x_1$ — истинно, если рассмотренный слот знака-фрейма имеет вложенный слот;

$ControlConcept.x_2$ — истинно, если на текущем уровне знака-фрейма есть не обработанный слот;

$ControlConcept.x_3$ — истинно, если рассмотренный слот знака-фрейма является вложенным слотом;

$ControlConcept.x_4$ — истинно, если текущий слот необходимо обработать;

$ControlConcept.x_5$ — истинно, если вызванный автомат завершил работу;

$ControlConcept.x_6$ — возвращает код категории знака-фрейма от 1 до 6 (1 — Понятие, 2 — Действие и т.д.);

$ControlConcept.x_7$ — истинно, если в текущем слоте знака-фрейма прописан вызов присоединенной процедуры.

$Con\_Y = \{ Con\_y_0, Con\_y_1, Con\_y_2, Con\_y_3, Con\_y_4, Con\_y_5, Con\_y_6, Con\_y_7, Con\_y_8, Con\_y_9, Con\_y_{10}, Con\_y_{11}, Con\_y_{12}\}$ — конечное множество управляющих состояний:

Con_y_0 — «Инициализация знака-фрейма категории «Понятие»»;
Con_y_1 — «Переход к вложенному слоту»;
Con_y_2 — «Переход к следующему слоту»;
Con_y_3 — «Возврат к родительскому слоту»;
Con_y_4 — «Обработка текущего слота»;
Con_y_5 — «Завершение работы автомата»;
Con_y_6 — «Вызов присоединенной процедуры»;
Con_y_7 — «Вызов автомата *ReadConcept*»;
Con_y_8 — «Вызов автомата *ReadAction*»;
Con_y_9 — «Вызов автомата *ReadProperty*»;
Con_y_{10} — «Вызов автомата *ReadState*»;
Con_y_{11} — «Завершение работы присоединенной процедуры».

$Con\_Z = \{ControlConcept.z_1, ControlConcept.z_2, ControlConcept.z_3, ControlConcept.z_4, ControlConcept.z_5, ControlConcept.z_6, ControlConcept.z_7\}$ — конечное множество выходных воздействий:

$ControlConcept.z_1$ — «Обработать вложенный слот»;
$ControlConcept.z_2$ — «Обработать следующий слот»;
$ControlConcept.z_3$ — «Вернуться к родительскому слоту»;
$ControlConcept.z_4$ — «Выполнить присоединенную процедуру»;
$ControlConcept.z_5$ — «Передать управление вызвавшему автомату»;
$ControlConcept.z_6$ — «Завершить работу присоединенной процедуры»;
$ControlConcept.z_7$ — «Обработка знака-фрейма «Понятие»»;
$ControlConcept.z_8$ — «Предобработка знака-фрейма «Понятие»».

$Con\_y_0 \in Con\_Y$ — начальное состояние.

$Con\_\varphi = Con\_\varphi' \times Con\_\varphi''$ — функция выходов (выходных воздействий):

Con_φ': $Con\_Y \rightarrow Con\_Z$ — функция выходных воздействий в состояниях:

$Con\_\varphi'_1$: $Con\_y_0 \rightarrow ControlConcept.z_8$,
$Con\_\varphi'_2$: $Con\_y_1 \rightarrow ControlConcept.z_1$,
$Con\_\varphi'_3$: $Con\_y_2 \rightarrow ControlConcept.z_2$,
$Con\_\varphi'_4$: $Con\_y_3 \rightarrow ControlConcept.z_3$,
$Con\_\varphi'_5$: $Con\_y_4 \rightarrow ControlConcept.z_7$,
$Con\_\varphi'_6$: $Con\_y_5 \rightarrow ControlConcept.z_5$,
$Con\_\varphi'_7$: $Con\_y_6 \rightarrow ControlConcept.z_4$,
$Con\_\varphi'_8$: $Con\_y_7 \rightarrow ReadConcept$,

*Con_*φ′$_9$: *Con_y*$_8$→*ReadAction*,
*Con_*φ′$_{10}$: *Con_y*$_9$→*ReadProperty*,
*Con_*φ′$_{11}$: *Con_y*$_{10}$→*ReadState*,
*Con_*φ′$_{12}$: *Con_y*$_{11}$→*ControlConcept.z*$_6$.

*Con_*φ″: *Con_X*× *Con_Y*→*Con_Z* — функция выходных воздействий на переходах:

*Con_*φ″$_1$: (*p*2.*e*$_4$,*ControlConcept.z*$_8$)→ *Con_y*$_1$.

*Con_*φ={(*Con_y*$_0$→*ControlConcept.z*$_8$,
(*p*2.*e*$_4$,*ControlConcept.z*$_8$)→*Con_y*$_1$),
(*Con_y*$_1$→*ControlConcept.z*1,(*p*2.*e*$_4$,*ControlConcept.z*$_8$)→*Con_y*$_1$),
(*Con_y*$_2$→*ControlConcept.z*$_2$,(*p*2.*e*$_4$,*ControlConcept.z*$_8$)→*Con_y*$_1$),
(*Con_y*$_3$→*ControlConcept.z*$_3$, (*p*2.*e*$_4$,*ControlConcept.z*$_8$)→*Con_y*$_1$),
(*Con_y*$_4$→*ControlConcept.z*$_7$, (*p*2.*e*$_4$,*ControlConcept.z*$_8$)→*Con_y*$_1$),
(*Con_y*$_5$→*ControlConcept.z*$_5$, (*p*2.*e*$_4$,*ControlConcept.z*$_8$)→*Con_y*$_1$),
(*Con_y*$_6$→*ControlConcept.z*$_4$, (*p*2.*e*$_4$,*ControlConcept.z*$_8$)→*Con_y*$_1$),
(*Con_y*$_7$→*ReadConcept*,(*p*2.*e*$_4$,*ControlConcept.z*$_8$)→*Con_y*$_1$),
(*Con_y*$_8$→*ReadAction*, (*p*2.*e*$_4$,*ControlConcept.z*$_8$)→*Con_y*$_1$),
(*Con_y*$_9$→*ReadProperty*, (*p*2.*e*$_4$,*ControlConcept.z*$_8$)→*Con_y*$_1$),
(*Con_y*$_{10}$→*ReadState*, (*p*2.*e*$_4$,*ControlConcept.z*$_8$)→*Con_y*$_1$),
(*Con_y*$_{11}$→*ControlConcept.z*$_6$, (*p*2.*e*$_4$,*ControlConcept.z*$_8$)→*y*$_1$),

где *ReadAction*, *ReadProperty*, *ReadState* — вызываемые автоматы, отвечающие за обработку знаков-фреймов категорий «Действие», «Свойство» и «Состояние» соответственно.

*Con_*δ: *Con_X* × *Con_Y*→*Con_Y* — функция переходов для автомата *ReadConcept*, представлена ниже в таблице 3.

Таблица 3. Функция переходов *Con_*δ автомата *ReadConcept*

№	*Con_X*	*Con_Y*	*Con_X*×*Con_Y* → *Con_Y*
1	*p*2.*e*$_4$	*Con_y*$_0$	*Con_y*$_1$
2	*p*2.*e*$_4$[!*ControlConcept.x*$_4$&&!*ControlConcept.x*$_1$ &&*ControlConcept.x*$_2$]	*Con_y*$_1$	*Con_y*$_2$
3	*p*2.*e*$_4$[!*ControlConcept.x*$_4$&&*ControlConcept.x*$_1$]	*Con_y*$_2$	*Con_y*$_1$
4	*p*2.*e*$_4$[!*ControlConcept.x*$_4$&&!*ControlConcept.x*$_1$ &&*ControlConcept.x*$_2$]	*Con_y*$_2$	*Con_y*$_2$
5	*p*2.*e*$_4$[!*ControlConcept.x*$_4$&&*ControlConcept.x*$_1$]	*Con_y*$_1$	*Con_y*$_1$
6	*p*2.*e*$_4$[!*ControlConcept.x*$_4$&&!*ControlConcept.x*$_1$ &&!*ControlConcept.x*$_2$]	*Con_y*$_1$	*Con_y*$_3$
7	*p*2.*e*$_4$[!*ControlConcept.x*$_4$&&!*ControlConcept.x*$_1$ && !*ControlConcept.x*$_2$]	*Con_y*$_2$	*Con_y*$_3$
8	*p*2.*e*$_4$[*ControlConcept.x*$_2$&&*ControlConcept.x*$_3$]	*Con_y*$_3$	*Con_y*$_2$
9	*p*2.*e*$_4$[!*ControlConcept.x*$_2$&&*ControlConcept.x*$_3$]	*Con_y*$_3$	*Con_y*$_3$
10	*p*2.*e*$_4$[!*ControlConcept.x*$_3$]	*Con_y*$_3$	*Con_y*$_5$
11	*p*2.*e*$_4$	*Con_y*$_5$	*Con_y*$_1$

12	$p2.e_4$[*ControlConcept*.x_4]	Con_y_1	Con_y_4
13	$p2.e_4$[*ControlConcept*.x_1&&!*ControlConcept*.x_7]	Con_y_4	Con_y_1
14	$p2.e_4$[*ControlConcept*.x_4]	Con_y_2	Con_y_4
15	$p2.e_4$[!*ControlConcept*.x_1&&*ControlConcept*.x_2 &&!*ControlConcept*.x_7]	Con_y_4	Con_y_2
16	$p2.e_4$[!*ControlConcept*.x_1&&!*ControlConcept*.x_2 &&!*ControlConcept*.x_7]	Con_y_4	Con_y_3
17	$p2.e_4$[*ControlConcept*.x_7]	Con_y_4	Con_y_6
18	$p2.e_4$[*ControlConcept*.x_6=1]	Con_y_6	Con_y_7
19	$p2.e_4$[*ControlConcept*.x_6=2]	Con_y_6	Con_y_8
20	$p2.e_4$[*ControlConcept*.x_6=3]	Con_y_6	Con_y_9
21	$p2.e_4$[*ControlConcept*.x_6=4]	Con_y_6	Con_y_{10}
22	$p2.e_4$[*ControlConcept*.x_6<1 || *ControlConcept*.x_6>4]	Con_y_6	Con_y_{11}
23	$p2.e_4$[!*ControlConcept*.x_1&&*ControlConcept*.x_2]	Con_y_{11}	Con_y_2
24	$p2.e_4$[!*ControlConcept*.x_1&&!*ControlConcept*.x_2]	Con_y_{11}	Con_y_3
25	$p2.e_4$[*ControlConcept*.x_1]	Con_y_{11}	Con_y_1

Объект управления *ControlConcept* — это тройка вида:

$$ControlConcept =< Con\_V, Con\_f_q, Con\_f_c >, \qquad (11)$$

где Con_V — это множество всех возможных конфигураций слотов знака-фрейма категории «Понятие». Для обработки слотов знаков-фреймов используются множество команд Con_f_q и множество запросов Con_f_c:

Con_f_q = {*ControlConcept*.z_1, *ControlConcept*.z_2, *ControlConcept*.z_3, *ControlConcept*.z_4, *ControlConcept*.z_5, *ControlConcept*.z_6, *ControlConcept*.z_7, *ControlConcept*.z_8}, подробное описание данных команд приведено выше в множестве Con_Y из (10).

Con_f_c = {*ControlConcept*.x_1, *ControlConcept*.x_2, *ControlConcept*.x_3, *ControlConcept*.x_4, *ControlConcept*.x_5, *ControlConcept*.x_6}, подробное описание данных запросов приведено выше в множестве Con_X из (10).

Автоматизированные объекты управления AOU_3, AOU_4, AOU_5, AOU_6 и AOU_7, отвечающие за обработку знаков-фреймов категорий «Свойство», «Величина», «Действие», «Событие» и «Состояние» соответственно, организованы в соответствии с их назначением.

4. Реализация автоматной модели

Для программной реализации описанной выше автоматной модели СУ АСП используется технология автоматного программирования UniMod [3]. Согласно концепции автоматного программирования схема связей автоматов — это тройка вида:

$$D =< P, A, O >, \tag{12}$$

$$P = \{P_i \mid i = 1,.., k\}, \tag{13}$$

где P — множество источников событий, k — количество источников событий. Множество источников событий рассматриваемой системы имеет вид:

$$P = \{p_1, p_2\}, \tag{14}$$

$$p_1 = \{e_1\}, \tag{15}$$

$$p_2 = \{e_2, e_3, e_4, e_5\}. \tag{16}$$

$p1$ — поставщик событий, который генерирует событие $e1$; $p2$ — поставщик событий, возникающих в процессе чтения онтологии системы продукций. Описание этих событий приведено выше в таблице 2.

A — множество диаграмм состояний автоматов:

$$A = \{A_i \mid i = 1,.., g\}, \tag{17}$$

A_i —диаграмма состояний конкретного автомата;

g — количество диаграмм.

Ниже на рисунке 1 представлена построенная в среде UniMod схема связей автоматов.

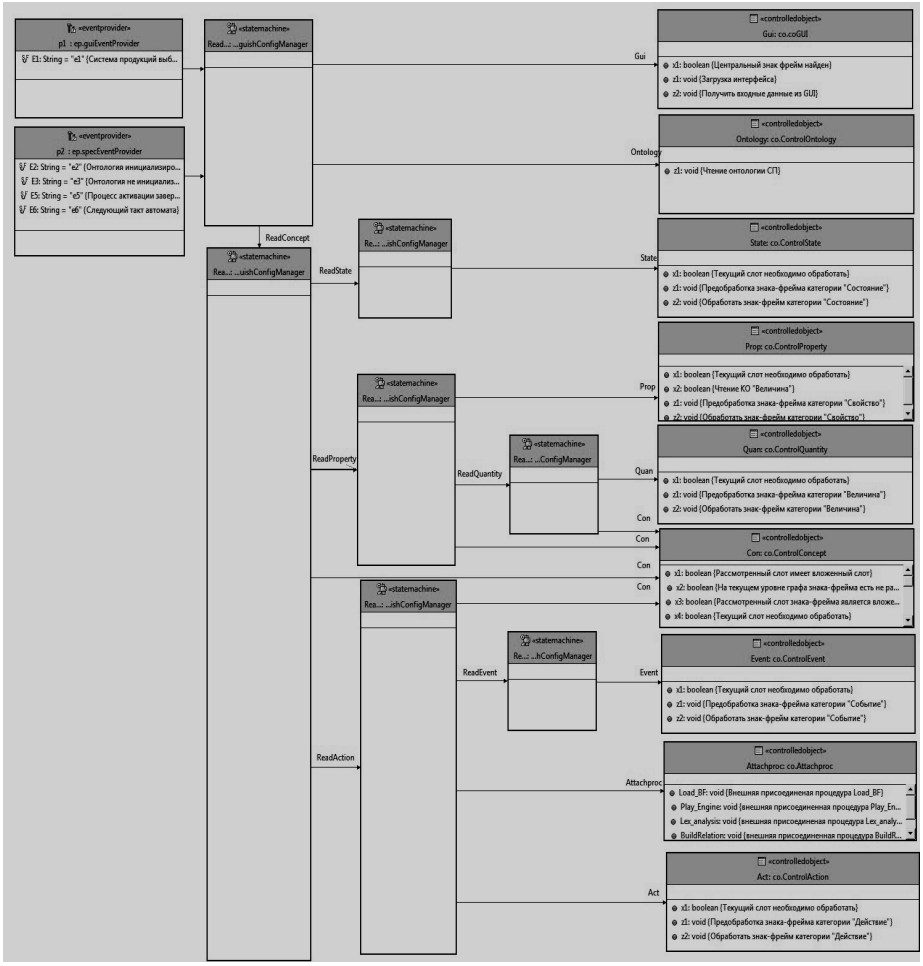

Рис. 1. Схема связей автоматов

Как видно из рисунка 1, схема связей автоматов содержит два поставщика событий $p1$ и $p2$, расположенных слева. В центре рисунка расположены семь автоматов. Между собой автоматы соединены стрелками, которые указывают на вызываемые автоматы. Справа на рисунке расположены объекты управления. Связь между автоматом и объектом управления показана стрелками.

В работе множество диаграмм состояний автоматов имеет вид:

$$A = \{ReadOntology, ReadConcept, ReadAction, ReadProperty, ReadQuantity, ReadState, ReadEvent\}, \quad (18)$$

ReadOntology, *ReadConcept*, *ReadAction*, *ReadProperty*, *ReadQuantity*, *ReadState* и *ReadEvent* автоматы математическое описание которых представлено выше.

O — множество объектов управления:

$$O = \{o_i \mid i = 1,..,r\}, \tag{19}$$

где r — количество объектов управления. Каждый элемент o_i содержит множество соответствующих ему входных и выходных воздействий, обозначаемых через множества X и Z соответственно:

$$o_i = \{X, Z\}, \tag{20}$$

$$X = \{x_i \mid i = 1,..,l\}, \tag{21}$$

$$Z = \{z_i \mid i = 1,..,m\}. \tag{22}$$

В данной системе множество объектов управления имеет вид:

$$O = \{Gui, ControlOntology, ControlConcept, ControlProperty, ControlAction, ControlQuantity, ControlState, ControlEvent, AttachProc\}, \tag{23}$$

Gui, *ControlOntology*, *ControlConcept*, *ControlProperty, ControlAction, ControlQuantity, ControlState, AttachProc* — объекты управления элементов из (4).

Автомат *ReadOntology*

Автомат *ReadOntology* — инициализирующий автомат, запускает приложение и задает начальные данные посредством взаимодействия пользователя с графическим интерфейсом. Автоматы *ReadConcept, ReadAction*, *ReadProperty*, *ReadQuantity*, *ReadStates*, *ReadEvents* реализуют логику работы с соответствующими объектами онтологии системы продукций.

Согласно рисунку 1 автомат *ReadOntology* взаимодействует с двумя объектами управления: *Gui* и *Con*. Также автомат *ReadOntology* осуществляет вызов автомата *ReadConcept*, задачей которого является разбор центрального знака-фрейма «Понятие». Граф переходов автомата *ReadOntology* представлен ниже на рисунке 2.

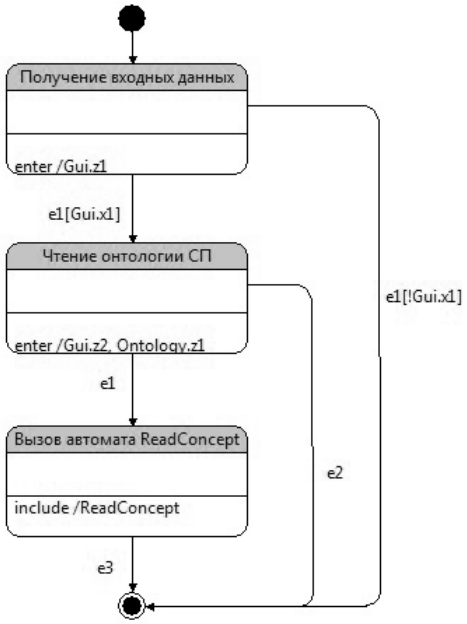

Рис. 2. Граф переходов автомата *ReadOntology*

Граф переходов автомата *ReadOntology* описывается четверкой вида:

$$ReadOntology =< Ont\_S, Ont\_E, Ont\_X, Ont\_Z >, \quad (24)$$

где *Ont_S* — множество возможных состояний автомата *ReadCO* из (6).

$$Ont\_E =< e_1, e_2, e_3 >, \quad (25)$$

элементы из множества *Ont_E* описаны выше в таблице 2.
Ont_X — множество входных воздействий из (6), *Ont_Z* — множество выходных воздействий из (6).

Автомат *ReadConcept*

Активация системы продукций начинается с чтения центрального знака-фрейма, который имеет структуру концептуального объекта «Понятие». Из рисунка 1 видно, что автомат *ReadConcept* взаимодействует с одним объектом управления *ControlConcept*, производит вызов автоматов *ReadProperty*, *ReadAction*, *ReadState*, а также вызывает рекурсивно самого себя. Ниже на рисунке 3 представлен граф переходов данного автомата.

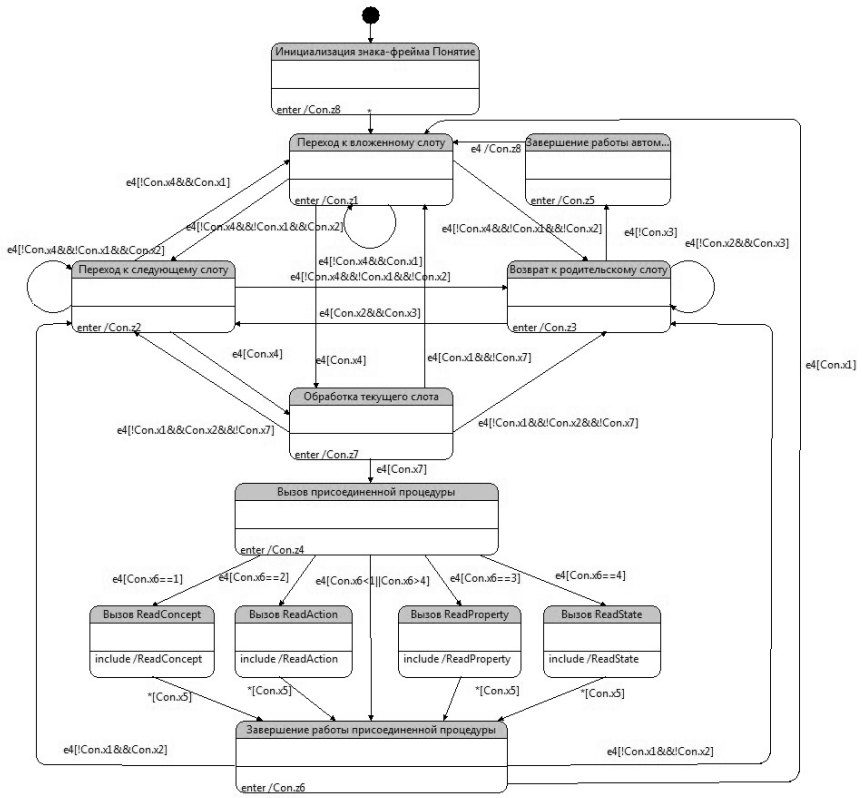

Рис. 3. Граф переходов автомата *ReadConcept*

Граф переходов автомата *ReadConcept* — это четверка вида:

$$ReadConcept =< Con\_S, Con\_E, Con\_X, Con\_Z >, \quad (26)$$

где *Con_S* — множество возможных состояний автомата *ReadConcept* из (10).

$$Con\_E =< e_4 >, \quad (27)$$

элементы из множества *Con_E* описаны выше в таблице 2.

где *Con_X*— множество входных воздействий из (10), *Con_Z* — множество выходных воздействий из (10).

Остальные автоматы *ReadProperty, ReadAction, ReadStates, ReadQuantity* и *ReadEvents* реализованы аналогично автомату *ReadConcept* с учетом особенностей структуры соответствующих концептуальных объектов.

5. Апробация

Полученное программное обеспечение (далее модуль активации систем продукций), реализующее СУ АСП, позволило провести серии вычислительных экспериментов на системах продукций из заданной базы знаний и словарных статьях из терминологического словаря[1].

В статье приведено краткое описание вычислительного эксперимента, проводимого на словарной статье: «ДЕНЬГИ — одно из важнейших понятий любого рыночного хозяйства: любые общепринятые средства платежа, которые могут обмениваться на товары и услуги и использоваться для оплаты долгов. Деньги — всеобщий эквивалент, особый товар, форма выражения стоимости всех других товаров. Деньги выполняют функции средства обмена, платежей, измерения стоимости, накопления богатства, мировых денег. Несмотря на то, что деньги возникли много тысяч лет назад, до сих пор в мире идут дискуссии о том, что такое деньги и почему они используются в хозяйстве» [5].

Выполнение *морфологического анализа* означает активацию системы продукций «Извлечение динамической морфологической информации». На рисунке 4 продемонстрирована структура базы данных aps, то есть рабочей памяти, сформированной для заданной системы продукций в автоматическом режиме.

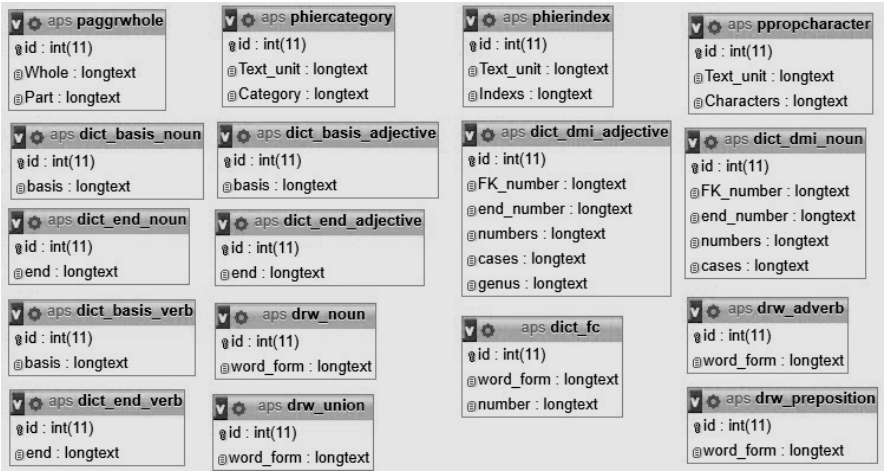

Рис. 4. Структура рабочей памяти «Извлечение динамической морфологической информации»

[1] Свидетельство об официальной регистрации программы для ЭВМ. Модуль активации систем продукций / заявитель и правообладатель ФГБОУ ВО ВСГУТУ. № 2018610271; заявл. 27.09.2017; опубл. 09.01.2018, Реестр программ для ЭВМ. 1 с.

Согласно рисунку 4 рабочая память содержит всего 17 таблиц, из которых 13 словарей — dict_basis_adjective, dict_basis_noun и т.д., и 4 таблицы paggrwhole, phiercategory, phierindex, ppropcharacter. После запуска машины логического вывода в таблицу PPropCharacter записывается морфологическая информация. Ниже представлен фрагмент содержимого данной таблицы.

id	Text_unit	Character
1	много	Наречие
…	…	…
4	и	Союз
…	…	…
8	деньги	Сущ.
…	…	…
57	деньги	148
…	…	…
107	деньги	Мн.ч.
…	…	…
44	деньги	Им.п.
…	…	…
351	любого	Муж.
…	…	…
364	такое	Ср.

Рис. 5. Фрагмент таблицы, содержащей динамическую морфологическую информацию о лексемах

Извлечение дефиниций из словарной статьи выполняется путем активации системы продукций «Извлечение дефиниции термина». На рисунке 6 представлена структура рабочей памяти aps, сформированной для заданной системы продукций в автоматическом режиме.

Рис. 6. Структура рабочей памяти для системы продукций «Извлечение дефиниции термина»

Рабочая память содержит всего 3 таблицы — paggrwhole, phiercategory и phierindex. После запуска машины логического вывода в таблицу phiercategory добавлена запись о дефиниции термина «Деньги» (рисунок 7).

Таблица phiercategory

id	Text_unit	Category
…	…	…
83	Деньги — одно из важнейших понятий любого рыночного хозяйства: любые общепринятые средства платежа, которые могут обмениваться на товары и услуги и использоваться для оплаты долгов.	дефиниция
84	Деньги	термин

Рис. 7. Фрагмент таблицы, содержащей текстовые единицы и их категории

В ходе проведенных вычислительных экспериментов были получены данные, которые соответствуют ожидаемым результатам тестирования, что подтверждает корректность предложенной автоматной модели СУ АСП.

Заключение

Описанная в статье автоматная модель системы управления активацией систем продукций была разработана в инструментальной среде UniMod, поддерживающей технологию автоматного программирования. Использование UniMod значительно снизило трудоемкость разработки системы как на этапе проектирования, валидации и отладки модели, так и на этапе ее программной реализации [3]. В дальнейшем в случае необходимости в модель и программу без особых сложностей могут быть внесены изменения.

Полученный в результате модуль управления активацией систем продукций является одним компонентов системы, основанной на правилах и предназначенной для извлечения знаний из текстов. Модуль позволяет выполнять в автоматическом режиме: выбор системы продукций для активации; построение рабочей памяти, соответствующей активной системе продукций; запуск машины логического вывода на активной системе продукций. Это стало возможным за счет применения онтологии с активной семантикой, представляющей знания о системе продукций [1].

Разработанный модуль позволит продолжить исследования по разработке методов автоматического извлечения знаний из текстов[2]. Однако проведенные вычислительные эксперименты показали, что полученное программное обеспечение может быть использовано для решения других задач. Для этого необходимо дополнить базу правил системами продукций, реализующими методы решения задачи, а также создать спецификации систем продукций в виде онтологий.

[2] Свидетельство об официальной регистрации программы для ЭВМ. Модуль активации систем продукций / заявитель и правообладатель ФГБОУ ВО ВСГУТУ. № 2018610271; заявл. 27.09.2017; опубл. 09.01.2018, Реестр программ для ЭВМ. 1 с.

Литература

1. Дармахеева Т. А., Хаптахаева Н. Б. Онтологическая модель представления знаний о системе продукций. «ЗНАНИЯ — ОНТОЛОГИИ — ТЕОРИИ» (ЗОНТ-2017): матер. всеросс. конф. с междунар. участием. Новосибирск: Изд-во СО РАН Институт математики им. С. Л. Соболева, 2017. С. 119–129.

2. Найханова Л. В. Технология создания методов автоматического построения онтологий с применением генетического и автоматного программирования: монография. 2-е изд., 2013. Улан-Удэ: Изд-во ВСГУТУ, 268 с.

3. Поликарпова Н. И., Шалыто А. А. Автоматное программирование. СПб.: Питер, 2009. 176 с.

4. Шатуева (Дармахеева) Т. А., Хаптахаева Н. Б. Формирование рабочей памяти продукционной системы, предназначенной для автоматического извлечения знаний из текстов // Вестник Восточно-Сибирского государственного университета технологий и управления. 2013. № 4 (43). С. 69–74.

5. Борисов А. Б. Большой экономический словарь. М.: Книжный мир, 2003. 895 с.

MODELING OF SYSTEM FOR CONTROLING ACTIVATION OF PRODUCTION RULES WITH THE USE OF AUTOMATIC PROGRAMMING

Tatyana A. Darmakheeva
Assistant, Department of systems of computer science,
East Siberia State University of Technology and Management
40V, Klyuchevskaya St., Ulan-Ude, 670013, Russia
E-mail: tania-shatyeva@mail.ru

Natalya B. Khaptakhaeva
Cand. Sci. (Engineering), A/Prof,
Head of the Department for training of higher qualification personnel,
East Siberia State University of Technology and Management
40V, Klyuchevskaya St., Ulan-Ude, 670013, Russia
E-mail: khapnb@gmail.com

The article presents the results of modeling the system for controlling activation of production systems. The activation of production systems is controlled through the use of knowledge of the system of production in the form of an ontology with active semantics. The behavior of the system that implements the control of the activation of production systems based on ontology with active semantics is complex. The theory of automata and the technology of automatic programming are used to simulate such a system. As a result, an automatic model of the system for controlling the activation of production systems was obtained and its software implementation was carried out. Testing of methods of natural language processing of the text and extraction of knowledge concerning terms and relations between them from the dictionary articles of the terminological dictionary were carried out by means of the obtained software. The methods are presented in the form of production systems.
Keywords: theory of automata; ontology with active semantics; modelling the systems with complex behavior; control the activation of production systems; system, based on knowledge; base of knowledge; production system.

References

1. Darmakheeva T. A., Khaptakhaeva N. B. Ontologicheskaya model' predstavleniya znanii o sisteme produktsii [Ontological Model of Presentation Knowledge on Production System]. *"Znaniya – Ontologii – Teorii" (ZONT - 2017) materialy Vserossiiskoi konferentsii s mezhdunarodnym uchastiem - "Knowledge –Ontologies – Theories (ZONT - 2017)": Proceedings of All-Russia conference with international participation*. Novosibirsk: SB RAS Institute of Mathematics named after S. L. Sobolev, 2017. Pp. 119–129.

2. Naikhanova L. V. *Tekhnologiya sozdaniya metodov avtomaticheskogo postroeniya ontologii s primeneniem geneticheskogo i avtomatnogo programmirovaniya* [Technology of Creating Methods for Automatic Construction of Ontologies with the Use of Genetic and Automatic Programming]. 2-nd edition. East Siberian State University of Technology and Management Publ., 2013. 268 p.

3. Polikarpova N. I., Shalyto A. A. *Avtomatnoe programmirovanie* [Automatic Programming]. St. Petersburg: Piter Publ., 2009. 176 p.

4. Shatueva (Darmakheeva) T. A., Khaptakhaeva N. B. Formirovanie rabochei pamyati produktsionnoi sistemy, prednaznachennoi dlya avtomaticheskogo izvlecheniya znanii iz tekstov [Formation of Working Memory of a Production System Designed to Automatically Extract Knowledge from Texts]. *Vestnik Vostochno-Sibirskogo gosudarstvennogo universiteta tekhnologii i upravleniya – Bulletin of the East Siberian State University of Technology and Management*. 2013. No. 4 (43). Pp. 69–74.

5. Borisov A. B. *Bol'shoi ekonomicheskii slovar* [Big Economic Dictionary]. Moscow: Knizhnyi Mir Publ., 2003. 895 p.