

СПЕЦИФИКА ВЫПОЛНЕНИЯ ТЕОРЕТИКО-МНОЖЕСТВЕННЫХ ОПЕРАЦИЙ НАД КОНТЕКСТНО-СВОБОДНЫМИ ГРАММАТИКАМИ В УСЛОВИЯХ РАЗЛИЧНЫХ ФОРМ ДОПОЛНИТЕЛЬНЫХ СЕМАНТИЧЕСКИХ ПРАВИЛ В СЕМИОТИЧЕСКОЙ МОДЕЛИ ИНТЕЛЛЕКТУАЛЬНЫХ САПР

Григорьев А.В.
Кафедра ПМиИ, ДонНТУ
grigorie@r5.dgtu.donetsk.ua

Abstract

Grigoriev A.V. Implementation Specific character of tch theoretical plural operations over context-free grammars In conditions of diverse forms of supplementary semantic regulations in semiotics model intellectual CAD. In work consider the implementation peculiarities of tch theoretical -plural operations in conditions: 1) presence of usual semantic inference regulations; 2) presence in grammars elements of supplementary prototypes lists. Consider the examples, a formal not formal algorithms record leads.

Введение

Ранее в работах [1, 2, 3] автором была описана семиотическая модель интеллектуальных САПР. В рассматриваемой модели САПР имеется два языка – язык описаний текстов прототипов и язык диалога синтеза прототипов.

Данные языки соответствуют в модифицированной теории сложности САУ [3] двум компонентам – целевому пространству систем (ЦПС), т.е. множеству решений – прототипов и пространству обликов систем (ПОС), т.е. множеству технических заданий. Соответственно, имеются две различные грамматики – грамматика описаний текстов прототипов в ЦПС и грамматика диалога синтеза прототипов в ПОС.

В грамматике описания множества прототипов имеется возможность построить любое из возможных описаний прототипов на формальном внутреннем языке представления моделей некоторой САПР.

В грамматике диалога каждый ИЛИ-синтерм трактуется как возможный вопрос к пользователю при синтезе решений и имеет дополнительно пустой элемент и возможность выбрать любое подмножество элементов, включая все (полный элемент).

ТМО над грамматиками позволяют обеспечить процесс обучения БЗн САПР, т.е. автоматическое построение ЦПС, а так же процесс синтеза решений (прототипов) по ПОС, сформированному автоматически по ЦПС или же напрямую построенному пользователем.

Ранее в работах [5, 6] детально рассматривался алгоритм выполнения ТМО над порождающими КС-грамматиками, но без учета наличия семантических правил вывода. В соответствии с классификацией семантических правил вывода, введенной в работах [7], различаются:

- неявные продукции, предполагающие наличие списков прототипов у терминальных и нетерминальных символов контекстно-свободной (КС) грамматики, призванные обеспечить выбор только семантически возможных решений;

- явные продукции, вводимые пользователем над прямо введенными КС порождающими грамматиками и призванные зафиксировать возможные и невозможные комбинации альтернатив в ИЛИ-синтермах (нетерминалах).

Достоинства и недостатки двух подходов:

Неявные продукции – это средство задания полной контекстной зависимости.

Достоинства:

- высокая степень автоматизации процессов обучения и вывода в базе знаний;

- не требуется высокий уровень квалификации пользователя (эксперт в предметной области может заменить эксперта по знаниям);

- не требуется верификация базы знаний.

Недостатки:

- необходимость иметь достаточно большое число проверенных на практике текстов описания объектов подходящего назначения;

- жесткость процесса создания базы знаний, отсутствие гибкости, невозможность для эксперта вводить свое понимание методики проектирования.

Явные продукции – это средство задания частичной контекстной зависимости, в той степени, которая соответствует знаниям пользователя.

Достоинства:

- быстрый ввод базы знаний;

- гибкость, возможность для эксперта вводить свое понимание методики проектирования.

Недостатки:

- низкая степень автоматизации;

- требуется высокая квалификация эксперта (т.е. дополнительно нужен эксперт по знаниям);

- требуется верификация базы знаний.

Ранее в работе [8] были определены основные принципы:

- организации диалога синтеза текстов прототипов;
- манипуляции списками прототипов в грамматике при синтезе.

Недостатком данных работ является отсутствие детального описания алгоритма ТМО над КС-грамматиками в случае наличия:

- списков прототипов в грамматике, т.е. неявных продукций;
- явных продукций в грамматике.

Целью данной работы является устранение названных недостатков. Учитывая отсутствие подобных решений в литературе, можно отметить актуальность и важность решения данной задачи.

1. Характеристика ТМО над КС-грамматиками как инструментального средства в СМ

Рассматривается класс контекстно-свободных грамматик, эквивалентный И-ИЛИ-дереву. С точки зрения специфики грамматики это означает: 1) отсутствие рекурсии; 2) наличие либо чисто дизъюнктивных, либо чисто конъюнктивных правил вывода; 3) синтермы с разными именами могут описывать частично совпадающее множество слов и т.д. [9]. Предлагаемый алгоритм теоретико-множественных операций над грамматиками ориентирован на специфику данного подкласса КС-грамматик и имеет такие особенности [5,6]:

- задача алгоритма – сравнить две грамматики и построить четыре новые грамматики, соответствующие результатам ТМО – объединение, разница, дополнение, пересечение;
- алгоритм выполняет сравнение грамматик как двух множеств текстов (слов), представленных в сравниваемых грамматиках;
- две грамматики имеют разное назначение – первая, ведущая, а вторая - ведомая;
- ведущая грамматика просматривается по всем возможным ветвям, а – ведомая просматривается многократным проходом вперед и назад с целью поиска совпавших слов или групп слов, заданных синтермами в текущем месте ведущей грамматики;
- при сравнении алгоритм делит исходные синтермы в обеих грамматиках на две части – общая (совпавшая), и – отличная (не совпавшая);
- создаются новые оригинальные имена новых синтермов, как совпавших так и не совпавших (амперсенд четный и – амперсенд нечетный);
- обход обеих грамматик выполняется слева - направо и сверху (от корня) – вниз (к ветвям);
- как для ведомого, так и для ведущего дерева имеется прямой и обратный ход алгоритма;
- прямой ход – проверка некоторой ветви дерева;

- обратный – возврат к месту ветвления с параллельным формированием четного и нечетного амперсандов в каждом из пройденных ранее синтермов;
- прямой ход – разворачивание синтермов на компоненты по ходу просмотра;
- обратный ход – сворачивание развернутых ранее синтермов, при автоматическом создании новых синтермов – совпавшей и – не совпавшей части, на которые разбиваются «старые» ИЛИ-синтермы; каждая новая часть содержит подмножество альтернатив из старого ИЛИ-синтерма;
- используются 2 пары стеков для хранения цепочек разложенных синтермов;
- знак «&» или «\» в верхушке стека имеет смысл граничного символа разложения;
- синтерм считается не совпавшим, если хотя бы один его компонент не совпал;
- не совпавшие синтермы раскрываются далее, с целью поиска либо совпадающих синтермов, либо с целью деления нижележащего множества цепочек терминальных символов на совпавшую и не совпавшую часть.

Таким образом, предложенный алгоритм является комплексным, т.е. включает в себя компоненты практически всех известных алгоритмов грамматического разбора текстов.

Упомянув об этом, необходимо кратко рассмотреть трактовку типичных этапов работы компилятора с точки зрения задач работы с текстами в СМ. Практически во всех трансляторах (и в компиляторах, и в интерпретаторах) в том или ином виде присутствует большая часть перечисленных ниже процессов: лексический анализ; синтаксический анализ; семантический анализ; генерация внутреннего представления программы; оптимизация; генерация объектной программы. В данном случае данные этапы работы приобретают новую трактовку.

Синтаксический анализ.

Выполняется всегда, но роль его другая – найти несовпадения, которые трактуются не как ошибки, а как – отличие, позволяющее выделить некоторое подмножество прототипов или технических заданий (ТЗ).

Лексический анализ.

- лексический анализ для ПОС может выполняться при наличии знания о грамматике описания ТЗ у пользователя, так, например, существует вариант САПР, разработанный по данной методике - система принятия решений по выбору мобильных телефонов по их характеристикам;

- лексический анализ для ЦПС так же может выполняться, смотри, например, интеллектуальную надстройку над языком проектирования аппаратуры вычислительной техники VHDL, где выделяется заголовок процедуры, описание переменных и тело процедуры [10].

Семантический анализ.

Ввод явных продукций экспертом – это правила семантического анализа, но используются они не для проверки изначально верных текстов, а - для синтеза семантически верных текстов в семиотической модели.

2. Алгоритм модификации явных продукций при выполнении ТМО

Рассмотрим роль ТМО над КС-грамматиками с дополнительными явными семантическими правилами вывода для различных случаев их использования.

1) Вывод и ТМО.

ТМО обеспечивает вывод в базе знаний, представленной как И-ИЛИ-дерево с определенными над ним дополнительными явными продукциями. В этом случае предполагается предварительный прямой экспертный ввод И-ИЛИ-дерева и дополнительных продукций над ним.

2) Обучение и ТМО.

Пусть есть множество баз знаний, т.е. И-ИЛИ-дереьев с определенными над ними дополнительными явными продукциями, построенными экспертным путем. В этом случае ТМО могут выполняться над отдельными И-ИЛИ-дереьями. Цель ТМО в этом случае – сформировать новое множество синтаксически верных выражений, более широкое или более узкое. Т.о., пользователь сам вводит И-ИЛИ-дереья, но может затем с ними выполнять любые действия для формирования нового И-ИЛИ-дерева, например:

- объединение двух И-ИЛИ-дереьев, в этом случае расширяется множество синтаксически верных выражений семиотической модели;
- вычитание одного И-ИЛИ-дерева из другого, в этом случае из одного И-ИЛИ-дерева удаляется некоторое подмножество синтаксически верных выражений и т.д.

Подведем итог двум приведенным случаям. С учетом условия, что синтермы с разными именами могут описывать частично совпадающее множество слов, алгоритм ТМО должен проводить анализ именно совпадения множества слов в двух деревьях, но при условии, что синтермы, имеющие одинаковые имена, совпадают и по структуре.

Продукции, определенные над одним из И-ИЛИ-дереьев, в этом случае *модифицируются по следующим правилам*. Поскольку исходные

синтермы после сравнения И-ИЛИ-деревьев делятся на три части – одна общая, совпавшая часть и две – различные, несовпавшие, то, соответственно, модифицируются и продукции, включающие некоторый исходный синтерм в посылке или выводе.

Если речь идет о посылке, то, соответственно, и посылка распадается на три условия. Если речь идет о выводе, то, соответственно, и вывод распадается на три действия. Т.о., продукция, имевшая, например, один синтерм в «посылке» и – один в «выводе», приобретает в самом сложном случае (при объединении) – три синтерма в посылке и – три – в выводе. При этом число результирующих продукций существенно увеличивается.

Рассмотрим случаи применения ТМО, а так же соответствие действий над грамматиками и продуктами:

1) При объединении грамматик в результирующий синтерм включаются все три части – результирующие грамматики, т.е. общая и две отличные. Старые продукции имеют отношение к общей части и отличной, «своей» части. Следовательно, фактически и продукции делятся на две части. Собственно продукции не изменяются, меняются их области определения.

2) При пересечении двух грамматик в результирующий синтерм включается только общая часть. Следовательно, соответствующим образом модифицируются и продукции. При этом, если новая посылка продукции становится изначально пуста – ее можно и удалить из рассмотрения. Но можно и оставить, учитывая, что она никогда не сработает, т.е. не мешает выводу, но может потребоваться в будущем.

3) При вычитании (или поиске дополнения) двух грамматик - в результирующий синтерм включается только одна из отличных частей. Следовательно, соответствующим образом модифицируются и продукции.

3. Специфика выполнения ТМО над КС-грамматиками с неявными продуктами

Рассмотрим отличия ТМО над грамматиками со списками от ТМО над грамматиками без списков. ТМО над грамматиками со списками прототипов предполагает ряд специфических условий, в частности:

- каждой грамматике соответствует свой набор прототипов, т.е. цепочек символов (слов-решений), составляющих описание прототипа на некотором формальном языке и идентифицируемых своими оригинальными идентификаторами, например – номерами прототипов;
- каждое текстовое отличие прототипа рассматривается как признак прототипа;
- каждый терминальный символ грамматики (терм) и не терминальный символ (синтерм) имеет список идентификаторов

прототипов, связанных соответственно либо с данным признаком, либо с группой признаков;

- номер или любой оригинальный идентификатор дается некоторому решению-прототипу однозначно, не зависимо от того, в какую грамматику входит данный прототип.

Отличия в алгоритме для ТМО со списками предполагает, что:

- каждый исходный синтерм или терм двух грамматик, подвергаемых ТМО, имеет список прототипов;

- соответственно, каждый вновь формируемый в алгоритме ТМО синтерм (амперсенд, четный или - нечетный) получает свой список прототипов по правилам, свойственным тому или иному типу грамматик (выбора и синтеза).

Т.о., нам необходимо определить правила, по которым формируются списки прототипов для синтермов, исходя из их состава. Предлагается следующее решение данной задачи. ТМО над грамматиками со списками выполняется различно над различными по типу грамматиками. В частности:

1) в грамматике порождения (синтеза): ИЛИ-синтерм - объединяет списки прототипов своих компонентов, а И-синтерм - так же объединяет;

2) в грамматике диалога (выбора): ИЛИ-синтерм - объединяет списки прототипов своих компонентов, а И-синтерм - пересекает списки.

Т.о., корневой или начальный символ порождающей грамматики синтеза имеет полный список всех возможных синтезируемых прототипов, а начальный символ грамматики выбора - может быть пустым или включать только некоторое подмножество исходных прототипов, точнее тех прототипов, которые имеют хотя бы одно из значений всех базовых ИЛИ-синтермов.

В грамматике выбора все прототипы рассматриваются альтернативно, т.о. начальный символ включает в свой список только выбранные пользователем прототипы. Список окончательно формируется в момент окончания задания пользователем ТЗ на нужный ему прототип. Подмножество признаков, оставленное пользователем в том или ином ИЛИ-синтерме, приобретает смысл И-синтерма, поскольку все данные признаки устраивают пользователя в равной мере, т.е. одновременно.

Т.о., ТМО, выполняемые над грамматиками на этапе обучения БЗн, т.е. над исходными грамматиками синтеза, предполагают только объединение списков прототипов.

ТМО, выполняемые над грамматиками на этапе выбора решений из БЗн, т.е. над грамматиками выбора предполагают пересечение списков прототипов для И-синтермов.

В качестве частного вывода следует сказать, что неявные продукции так же в свою очередь выполняют ТМО, но уже со списками прототипов, принадлежащих синтермам и термам.

Т.о., при наличии ТМО над списками прототипов можно выполнять их уже над результирующими множествами, получившимися после ТМО над КС-грамматиками, при условии, что базовые термы сохраняют свои списки прототипов.

4. Формальная запись грамматик, обеспечивающих учет наличия списка прототипов в КС-грамматиках

Анализ ряда грамматик [11, 12] позволил сделать вывод, что наиболее подходящей грамматикой для формализации рассмотренного выше алгоритма являются атрибутные грамматики, а точнее их современная разновидность – предикативные грамматики. Выполнив некоторую модификацию, их можно использовать для формальной записи алгоритма. В соответствии с данными работами приведем описание простых предикативных грамматик.

Пусть имеется набор $G = (F, N, T, P, S)$, где:

F – счётное множество термов, образованных с помощью конечного множества конструкторов C и объектных переменных из конечного множества W (значением объектной переменной могут быть только термы из F , сама переменная также является термом); алгебра термов может быть многосортной;

N – конечный алфавит нетерминальных символов, для каждого из которых задан тип, определяющий допустимый набор параметров из F ;

T – конечный алфавит терминальных символов, которые также могут иметь параметры из F в соответствии с заданным типом; терминальные символы с параметрами позволяют ввести лексические классы и их представителей, например, *number(5)*;

P – конечное множество правил (или продукций); правило имеет вид $A \rightarrow \alpha$, где $A \in N$, $\alpha \in (T \cup N)^*$, причем A и символы из α содержат необходимые параметры;

S – начальный нетерминальный символ, $S \in N$.

Определим отношение выводимости на $(T \cup N)^*$. Из цепочки $\alpha A(\mathbf{t})\beta$, где $A \in N$, $\alpha, \beta \in (T \cup N)^*$, $\mathbf{t} = (t_1, \dots, t_n)$, $n \geq 0$, t_i – параметры A , непосредственно выводима цепочка $(\alpha\gamma\beta)\theta$, если есть правило $A(\mathbf{s}) \rightarrow \gamma$, где $\mathbf{s} = (s_1, \dots, s_n)$, s_i – параметры A , и θ – наиболее общее решение (НОР) системы уравнений $\mathbf{t} = \mathbf{s}$. Отношение обозначается $\alpha A(\mathbf{t})\beta \rightarrow_G (\alpha\gamma\beta)\theta$, а его рефлексивно-транзитивное замыкание \rightarrow_G^* называется отношением выводимости.

Формально опишем грамматику синтеза.

Будем рассматривать два сорта пустых цепочек символов:

- обычные, т.е. относящиеся к грамматике синтеза, будем обозначать как Θ ;
- пустые цепочки выбора - ε .

Множество W объектных переменных включает оригинальные идентификаторы (как возможный вариант - номера) прототипов. Имеется два типа синтермов, т.е. нетерминалов, - конъюнктивные и дизъюнктивные, соответственно A и B . Для типа A правила вывода имеют вид:

$$A(t_A) \rightarrow \& a_n(s_{a_n}); \text{ где}$$

$A \in N, a_n \in (T \cup N), t_A \subseteq F, s_{a_n} = (t_1, \dots, t_k), t_i \in F$. Соответствующая система уравнений для объектных переменных имеет вид:

$$t_A = \bigcup_{n=1}^N s_{a_n}, \text{ где под решением понимается состав списков}$$

переменных, удовлетворяющих данному уравнению. Аналогично для типа B :

$$B(s_B) \rightarrow \vee b_n(s_{b_n});$$

При этом допустимо, что $b_1 \rightarrow \Theta$, т.е. b_1 – пустой элемент списка.

Соответствующая система уравнений для объектных переменных имеет вид:

$$s_B = \bigcup_{n=1}^N s_{b_n}; s_{b_1} = s_S \setminus s_B; s_{b_2} = s_B;$$

Формально опишем грамматику выбора.

Так же рассматривается два типа синтермов, т.е. нетерминалов, конъюнктивные и дизъюнктивные, соответственно A и B . Для типа A правила вывода имеют вид:

$$A(t_A) \rightarrow \& a_n(s_{a_n}); \text{ где}$$

$$A \in N, a_n \in (T \cup N), t_A \subseteq F, s_{a_n} = (t_1, \dots, t_k), t_i \in F.$$

Соответствующая система уравнений для объектных переменных имеет вид:

$$t_A = \bigcup_{n=1}^N s_{a_n}$$

Аналогично для типа B :

$$B(s_B) \rightarrow \vee b_n(s_{b_n}); B \in N, b_n \in (T \cup N), t_B \subseteq F, s_{b_n} = (t_1, \dots, t_k), t_i \in F$$

При этом:

$$N \geq 3, \text{ а так же } b_1 \rightarrow \varepsilon; b_2 \rightarrow B; b_3 \rightarrow \Xi; \text{ где } \Xi \subset B.$$

Соответствующая система уравнений для объектных переменных имеет вид:

$$s_B = \bigcap_{n=1}^N s_{b_n}; s_{b_1} = s_S \setminus s_B; s_{b_2} = s_B.$$

Рассмотрим ряд дополнительных правил вывода, позволяющих динамически модифицировать грамматику, исходя из состояния изменившихся списков прототипов в ИЛИ-альтернативах, которые пока не рассматривались. В этом случае необходимо выделить:

- удаление "пустой" альтернативы из списка альтернатив;
- удаление из рассмотрения ИЛИ-синтермов, где отсутствуют "непустые" элементы.

Формальная запись первого случая имеет вид. Наличие "опустевшей" альтернативы предполагает, что $\exists b_i : b_i \rightarrow \varepsilon$ при том, что $s_{b_i} = []$, т.е. пустой элемент вывода требует пустого списка объектных переменных. Такие элементы грамматики можно назвать отвергнутыми альтернативами. Правила вывода:

$$(B(s_B) \rightarrow \bigvee_{n=1}^N b_n(s_{b_n})) \& (\exists b_i : b_i \rightarrow \varepsilon) \rightarrow B(s_B) \rightarrow \bigvee_{n=1}^{N-1} b_n(s_{b_n})$$

$$\text{для } \forall B \in (T \cup N) : s_B = s_B \setminus s_{b_i}, s_{b_n} = s_{b_n} \setminus s_{b_i}.$$

Аналогично может быть описан и второй случай.

5. Пример использования грамматики синтеза в форме грамматики выбора

Рассмотрим пример использования грамматики синтеза уже в качестве грамматики выбора. Согласно классификации САПР, представленной в [4], будем рассматривать вариант САПР - "есть ПОС, нет ЦПС". В данном случае отсутствует детальное описание прототипов в ЦПС как технических объектов, но каждый прототип имеет однозначно соответствующее ему полное ТЗ, играющее роль прототипа в ЦПС. ПОС будет включать "короткие" ТЗ, составляющие только значимые признаки. Пусть так же имеет место лексический анализ для множества ТЗ, позволяющий явно выделить отдельные компоненты ТЗ, т.е. отдельные признаки и их значения.

Рассмотрим простейшее множество прототипов "транспорт" со своими признаками:

Подмножество "Мотоцикл".

- 1 прототип - мото, базовый, 2 колеса, 2 пассажира, нет крыши;
- 2 прототип - мото, с коляской, 3 колеса, 3 пассажира, нет крыши.

Подмножество "Автомобиль".

- 3 прототип - авто, прогулочный , 4 колеса, 2 пассажира, есть крыша;

- 4 прототип - авто, базовый , 4 колеса, 4 пассажира, есть крыша.

Соответствующая *грамматика синтеза*, но без списков, имеет вид:

$\langle \text{транспорт} \rangle ::= \langle \text{колеса} \rangle | \langle \text{салон} \rangle$

$\langle \text{салон} \rangle ::= \langle \text{пассажиры} \rangle \& \langle \text{крыша} \rangle$

$\langle \text{колеса} \rangle ::= 2k | 3k | 4k$

$\langle \text{пассажиры} \rangle ::= 2p | 3p | 4p$

$\langle \text{крыша} \rangle ::= \emptyset | y$

Опишем списки прототипов для термов:

$2k \rightarrow (1)$

$3k \rightarrow (2)$

$4k \rightarrow (3,4)$

$2p \rightarrow (1,3)$

$3p \rightarrow (2)$

$4p \rightarrow (4)$

$\emptyset \rightarrow (1,2)$

$y \rightarrow (3,4)$

Синтермы грамматики на основе предложенного механизма получают свои списки прототипов, путем движения снизу-вверх:

$\langle \text{колеса} \rangle \rightarrow (1) \cup (2) \cup (3,4) = (1,2,3,4)$

$\langle \text{пассажиры} \rangle \rightarrow (1,3) \cup (2) \cup (4) = (1,2,3,4)$

$\langle \text{крыша} \rangle \rightarrow (1,2) \cup (3,4) = (1,2,3,4)$

$\langle \text{салон} \rangle \rightarrow (1,2,3,4) \cup (1,2,3,4) = (1,2,3,4)$

$\langle \text{транспорт} \rangle \rightarrow (1,2,3,4) \cup (1,2,3,4) = (1,2,3,4)$

Рассмотрим простейший вариант соотношения пары ЦПС-ПОС, т.е. будем считать, что ПОС автоматически формируется по данному ЦПС. Т.о., множество ИЛИ-синтермов грамматики синтеза составляет множество признаков прототипов, входящих в ПОС. Тогда соответствующая *грамматика выбора* будет иметь вид:

$\langle \text{транспорт} \rangle ::= \langle \text{колеса} \rangle | \langle \text{салон} \rangle$

$\langle \text{салон} \rangle ::= \langle \text{пассажиры} \rangle \& \langle \text{крыша} \rangle$

$\langle \text{колеса} \rangle ::= 2k | 3k | 4k | \emptyset | \Omega | \Xi$

$\langle \text{пассажиры} \rangle ::= 2p | 3p | 4p | \emptyset | \Omega | \Xi$

$\langle \text{крыша} \rangle ::= y | \emptyset | \Omega$

где:

\emptyset - пустой элемент, т.е. "ничто не подходит";

Ω - полное множество, т.е. "все подходит";

Ξ - произвольное непустое подмножество исходного синтерма, мощностью более одного элемента, т.е. "подходит несколько".

Следует отметить, что для синтерма "<крыша>" вариант Ξ - невозможен, поскольку список непустых элементов синтерма состоит только из одного элемента, а вариант \emptyset - не добавляется вновь, поскольку уже изначально присутствовал в синтерме.

Пустой элемент, добавляемый в ИЛИ-синтермы, приобретает собственный список прототипов, различный в зависимости от контекста, т.е. в различных ИЛИ-синтермах он формируется по разному, а именно - включает все возможные прототипы за исключением имеющихся в данном ИЛИ-синтерме. Соответственно в нашем случае они имеют такой вид:

$$[\emptyset_{\subset \langle \text{колеса} \rangle}] \rightarrow ()$$

$$[\emptyset_{\subset \langle \text{пассажиры} \rangle}] \rightarrow ()$$

$$[\emptyset_{\subset \langle \text{крыша} \rangle}] \rightarrow (1,2)$$

Рассмотрим вначале вариант грамматики выбора без дополнительных, семантических продукций, связанных со списками прототипов, т.е. неявных продукций вывода. В этом случае ответ, полученный на некоторый вопрос, не сужает автоматически тут же списки возможных альтернатив-признаков последующих вопросов.

Пусть пользователь определил следующий состав требований:

$$\langle \text{колеса} \rangle \Rightarrow \Xi (2k \mid 3k)$$

$$\langle \text{пассажиры} \rangle \Rightarrow 3p$$

$$\langle \text{крыша} \rangle \Rightarrow \Omega (y)$$

Тогда по итогам ввода требований синтермы получают следующие списки прототипов:

$$\langle \text{колеса} \rangle \rightarrow (1) \cup (2) = (1,2)$$

$$\langle \text{пассажиры} \rangle \rightarrow (2)$$

$$\langle \text{крыша} \rangle \rightarrow (1,2) \cup (3,4) = (1,2,3,4)$$

$$\langle \text{салон} \rangle \rightarrow (2) \cap (1,2,3,4) = (2)$$

$$\langle \text{транспорт} \rangle \rightarrow (1,2) \cap (2) = (2)$$

Рассмотрим далее вариант грамматики с дополнительными, семантическими продукциями. В этом случае ответ, полученный на некоторый вопрос, тут же сужает списки альтернатив-признаков для возможных последующих вопросов за счет неявных продукций.

Применение неявных продукций приводит к тому, что если в данном ответе мы отказались от ряда признаков, однозначно связанных с некоторыми прототипами, то данные прототипы удаляются во всех признаках-термах других вопросов. Кроме того, если список данного признака-альтернативы - опустел, то из диалога удаляется и сама альтернатива.

Пусть пользователь определил следующий состав требований:

1-й Вопрос:

$$\langle \text{колеса} \rangle ::= 2k \mid 3k \mid 4k \mid \emptyset \mid \Omega \mid \Xi$$

Ответ:

$\langle \text{колеса} \rangle \Rightarrow \Xi (2k | 3k)$

2-й Вопрос:

$\langle \text{пассажиры} \rangle ::= 2p | 3p | \emptyset | \Omega | \Xi$

Ответ:

$\langle \text{пассажиры} \rangle \Rightarrow 3p$

3-й Вопрос:

$\langle \text{крыша} \rangle ::= \emptyset$

Поскольку ответ на вопрос - однозначный, то пользователю он не задается и ответ, как и вопрос - отсутствует.

Тогда по итогам ввода требований синтермы получаются следующие списки прототипов:

$\langle \text{колеса} \rangle \rightarrow (2)$

$\langle \text{пассажиры} \rangle \rightarrow (2)$

$\langle \text{крыша} \rangle \rightarrow (1,2)$

$\langle \text{салон} \rangle \rightarrow (2) \cap (1,2) = (2)$

$\langle \text{транспорт} \rangle \rightarrow (2) \cap (2) = (2)$

Таким образом, искомым является второй прототип, т.е. «мотоцикл с коляской».

Заключение

В предлагаемой работе решена задача определения специфики выполнения теоретико-множественных операций над контекстно-свободными грамматиками в условиях различных форм дополнительных семантических правил в семиотической модели интеллектуальных САПР.

В работе детально рассматриваются особенности выполнения теоретико-множественных операций в условиях:

- 1) наличия обычных семантических правил вывода;
- 2) наличия у элементов грамматик дополнительных списков прототипов.

Приведены примеры, приводится формальная и не формальная запись алгоритмов.

Полученные результаты позволяют в полной мере реализовать семиотическую модель САПР. Как перспективную задачу следует выделить детальную классификацию упомянутого алгоритма выполнения ТМО над КС-грамматиками с точки зрения существующих средств и методов грамматического разбора.

Литература

1. Григорьев А.В. Семиотическая модель базы знаний САПР. Научные труды Донецкого государственного университета. Серия

"Проблемы моделирования и автоматизации проектирования динамических систем". Выпуск 10: - Донецк: ДонГТУ, 1999. - С. 30-37.

2. Григорьев А.В., Каспаров А.А. Обобщение знаний в интеллектуальной системе с семиотической моделью предметной области. Научные труды Донецкого государственного университета. Серия "Проблемы моделирования и автоматизации проектирования динамических систем". Выпуск 29. - Севастополь, "Вебер", 2001. - С. 106-113.

3. Григорьев А.В. Упорядочивание обликов в семиотической модели САПР /Научно-теоретический журнал «Искусственный интеллект», №4, 2005. – Донецк: ИПИИ МОН и НАН Украины «Наука и образование», 2005. – с. 465–477.

4. Григорьев А.В. Пути создания интеллектуальных САПР при различных уровнях квалификации экспертов /Научно-теоретический журнал «Искусственный интеллект», №3, 2005. – Донецк: ИПИИ МОН и НАН Украины «Наука и образование», 2005. – с. 758–763.

5. Григорьев А.В. Теоретико-множественные операции над грамматиками как механизм работы со знаниями в интеллектуальных САПР. Вісник Східноукраїнського національного університету імені Володимира Даля, N 2(48). Луганск, ВУТУ, 2002. С. 186-194.

6. Григорьев А.В. Алгоритм выполнения теоретико-множественных операций над грамматиками в среде специализированной оболочки для создания интеллектуальных САПР. Наукові праці національного технічного університету. Серія «Проблеми моделювання і автоматизації проектування динамічних систем» (МАП -2002). Выпуск 52: Донецк: ДонНТУ, 2002. - С.83-93.

7. Григорьев А.В. Классификация типов продукций в интеллектуальных САПР / Наукові праці національного технічного університету. Серія «Обчислювальна техніка та автоматизація». Выпуск 88. -: Донецк: ДонНТУ, 2005. – с. 99-105.

8. Григорьев А.В. Принципы организации вывода решений в базе знаний инструментальной оболочки для создания интеллектуальных САПР. // Практика і перспективи розвитку інституційного партнерства». Вісник ДонГТУ – ТРТУ. Донецьк: РВА ДонНТУ, 2003 – С.96-106.

9. Григорьев А.В., Каспаров А.А. И/ИЛИ-дерево как средство абстрактного представления знаний. Наукові праці національного технічного університету. Серія «Інформатика, кібернетика та обчислювальна техніка». Выпуск 39: Донецк: ДонНТУ, 2002. - С.36-42.

10. Григорьев А.В., Кошелева Д.А. Интеллектуализация процесса проектирования аппаратуры средствами языка VHDL / Моделирование и компьютерная графика: Материалы 1-й международной научно-

технической конференции, г Донецк, 04-07 октября 2005 г. — Донецк, ДонНТУ, Министерство образования и науки Украины, 2005. – с. 110-116.
1-я международная научно-техническая конференция «Моделирование и компьютерная графика», 04-07 октября 2005 г., г. Донецк, Украина.

11. Серебряков В.А., Галочкин М.П. Основы конструирования компиляторов. УРСС, 2001. 224 с

12. А. Ахо, Дж. Ульман. Теория синтаксического анализа, перевода и компиляции. Том 1. Синтаксический анализ. М. Мир: 1978.

Дата надходження до редакції 18.10.2006 р.