АЛГОРИТМ ВЫПОЛНЕНИЯ ТЕОРЕТИКО-МНОЖЕСТВЕННЫХ ОПЕРАЦИЙ НАД ГРАММАТИКАМИ В СРЕДЕ СПЕЦИАЛИЗИРОВАННОЙ ОБОЛОЧКИ ДЛЯ СОЗДАНИЯ ИНТЕЛЛЕКТУАЛЬНЫХ САПР

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

Abstract

Grigoriev A.V. Implementation algorithm of plural operations over grammars in wednesday of specialized envelope for creation intellectual CAD. *In work is offered an implementation algorithm of plural operations over originative grammars. A Given algorithm clever to provide work with knowledges in wednesday of instrumental envelope for creation intellectual CAD.*

Введение

Специфика предлагаемого подхода к представлению знаний в специализированной инструментальной оболочке для создания интеллектуальных САПР – мета-эвристической оболочке (МЭО) описана ранее в [1-6] и кратко может быть охарактеризована следующим образом. База знаний представляет собой И-ИЛИ-дерево с определенными отношениями (продукциями) над ИЛИ-синтермами. Цель вывода в базе знаний - обеспечение выбора требуемого прототипа по техническому заданию (ТЗ) как подмножеству значений ИЛИ-синтермов, т.е.:

1) Отношения между ИЛИ-узлами связывают те термы, комбинация которых принадлежит некоторому непустому множеству семантически верных (проверенных) прототипов, имеющих место в И-ИЛИ-дереве;

2) Аксиомы, или прототипы есть основа построения И-ИЛИ-дерева;

3) И-ИЛИ-дерево есть средство для компактной записи множества известных прототипов и порождения гипотез о возможных новых прототипах;

4) И-ИЛИ-дерево – это множество синтаксически правильных выражений;

5) Продукции определены над И-ИЛИ-деревом и задают правила вывода, которые в совокупности позволяют вычленить из И-ИЛИ-дерева семантически верное подмножество, т.е. те же самые аксиомы-прототипы.

При работе в открытой базы знаний в среде мета-эвристической оболочки возникают следующие задачи:

- обобщение множеств прототипов по структуре и по функциям в пределах отдельного модуля знаний, т.е. создание И-ИЛИ-дерева;
- аппарат вывода в смысле теории сложности САУ, по базе знаний – как целевого пространства систем (ЦПС) [7] на базе ТЗ [2], аппарат преобразования И-ИЛИ-деревьев.

Как в том, так и ином случае основу аппарата составляют теоретико-множественные операции (ТМО) над И-ИЛИ-деревом, и, в частности, такие операции, как пересечение, объединение, разность и дополнение множеств прототипов, хранящихся в И-ИЛИ-деревьях.

Все прототипы как модели объектов имеют в МЭО следующую иерархию описаний:

1) внешнее описание прототипа - ТЗ; назначение - внешняя идентификация, выбор прототипа из множества всех прототипов; интерфейс с малоквалифицированным пользователем;

2) описание на языке внутреннего представления, имеющего определенною грамматику: блоки, массивы блоков, свойства блоков, значения свойств, массивы свойств, связи, массивы связей; назначение - явное текстовое писание модели; интерфейс с высококвалифицированным пользователем;

3) табличную форму записи и хранения модели; назначение - внутренне, инструментальное представление прототипов в форме, позволяющей обеспечить оптимальную форму для хранения и преобразования; интерфейс с пользователем в этом случае - не предусматривается.

Табличная форма записи прототипов есть преобразованная форма представления на языке внутреннего представления.

И-ИЛИ-дерево как форма обобщения прототипов имеет место на любом уровне представления моделей. Форма Бэкуса-Наура (БНФ) может выступать как форма представления И-ИЛИ-дерева, свойственная грамматическому характеру описаний моделей на уровнях 1 и 2. С другой стороны, БНФ всегда можно соотнести с И-ИЛИ-деревом в любой форме представления, в том числе и табличной, свойственной уровню 3. Т.о. средства выполнения ТМО над БНФ может рассматриваться как универсальный аппарат работы с И-ИЛИ-деревьми в любой форме представления - от уровня 1 до уровня 3.

Т.о. необходимо создать аппарат ТМО над БНФ. Для чего необходимо определить:

- класс грамматик, представленных в форме БНФ;
- возможные ограничения на алгоритм, т.е. область его применения и возможности;
- выбрать оптимальный набор инструментальных средств его представления, определяемый спецификой задачи.

И-ИЛИ-деревья, представленные в виде БНФ, которые могут в зависимости от способа решения задачи обобщения и изобретения иметь две альтернативных формы представления:

1) Всякая альтернатива в ИЛИ-синтермах связана впрямую со списком идентификации ряда прототипов (исходная версия). Множество прототипов есть подмножество множества синтаксически верных выражений СМ. Грамматическая трактовка: вариант контекстной зависимости в грамматиках, заданных идентификацией прототипов.

2) Всякая альтернатива в ИЛИ-синтермах не связана впрямую со списком идентификации ряда прототипов. Множество прототипов совпадает с множеством синтаксически верных выражений СМ. Грамматическая трактовка: вариант контекстной независимости в грамматиках.

В работе [8] сделан обзор существующих и возможных методов решения данной задачи, определен класс грамматик, представленных в форме БНФ, введены возможные ограничения на алгоритм, т.е. область его применения и возможности и доказана возможность построения необходимого алгоритма.

В предлагаемой статье решается задача выбора оптимального набора инструментальных средств его представления, определяемый спецификой задачи и выполняется собственно построение алгоритма для второй формы представления И-ИЛИ-деревьев, предполагающей контекстную независимость в грамматиках.

1. Выбор средств реализации алгоритма

Для задания алгоритма возможно применение различных прогрессивных форм представления из среды CASE-технологий или объектно-ориентированного программирования.

Наиболее оптимальным путем есть применение идей и средств R-технологии автоматизации проектирования программ [9] по причине:

1) явного включения средств, ориентированных на работу с текстом;
2) удобная графическая форма представления алгоритма.

Пусть дано два множества текстов описания моделей, заданных БНФ. Необходимо сформировать алгоритм выполнения над ними теоретико-множественных операций (пересечение, объединение, разность, дополнение).

Предлагается следующий алгоритм сравнения двух множеств, порождающий на выходе результаты указанных теоретико-множественных операций (пересечение, объединение, разность, дополнение).

2. Основные определения, соглашения и ограничения

2.1. Основные определения

Определение 1.

Термом называется элементарный символ множества. Термы соединяются между собой только посредством операции "И" (&).

Определение 2.

Синтермом называется имя множества, которое может раскладываться. Элементами разложения могут быть как термы, так и синтермы, соединенные посредством операции "И" (&) или "ИЛИ" (V). Синтермы всегда записываются только в угловых скобках "< >".

Определение 3.

Если два множества совпадают по имени, то это означает, что они эквивалентны и по структуре, т.е. одно и то же имя означает одно и тоже множество.

Определение 4.

Если два множества совпадают по структуре, то это значит, что имена у них разные, а подмножества и способ их объединения одинаковый, т.е. одна и та же структура может иметь много разных форм записей, но при полном разложении этих форм записи мы в результате получим одно и то же.

2.2. Основные соглашения

Соглашение 1.

Элементы каждого отдельного множества должны соединяться только по "И", или только по "ИЛИ". Совместное использование этих двух знаков операций при записи разложения отдельного множества не допустимо.

Соглашение 2.

Не существует двух разных путей, порождающих одну и ту же цепочку термов.

Соглашение 3.

Каждый элемент по "ИЛИ" можно рассматривать как символ и как множество цепочек символов, начинающихся с этого символа. Если "ИЛИ" выступает как символ, присущие ему признаки помещаются в графе символа "СИМВ", в противном случае - в графе множества "МН".

Соглашение 4.

Если при движении по некоторому пути выяснилось, что данный символ "ИЛИ" представляет собой полностью просмотренное множество символов, то при движении по любому другому пути, приводящего к этому же символу, множество цепочек, начинающихся с этого символа, будет иметь тот же признак.

Соглашение 5.

Каждая строка вновь вводимого множества должна заканчиваться признаком конца "%". Знак "%" в конце строки играет служебную роль и в конечный текст не входит.

Соглашение 6.

Признаком "ЗНАК" (Z) помечаются все знаки множества как в основном и во вспомогательном стеках.

Соглашение 7.

Признаком "НЕОТРАБОТАН" (NO) помечаются все не отработанные элементы множества, соединенные по "ИЛИ" во вспомогательном стеке.

Соглашение 8.

Признаком "РАСКЛАДЫВАЕТСЯ" (R) помечаются все синтермы во вспомогательном стеке, которые подверглись разложению.

Соглашение 9.

Признаком "СОВПАДЕНИЕ" (S) помечаются все элементы множества (термы или синтермы) во вспомогательном стеке, которые совпали со сравниваемыми элементами другого множества.

Соглашение10.

Признаком "НЕСОВПАДЕНИЕ" (NS) помечаются все элементы множества (термы или синтермы) во вспомогательном стеке, которые не совпали со сравниваемыми элементами другого множества.

Соглашение11.

При описании алгоритмов использованы сокращения:

ZAKON - регистр, куда вписывается правая часть множества S="операция"(a1,a2,a3),

ТОРМ - таблица определения идентификаторов множеств (синтермов), каждая строка по структуре имеет вид S=&(a1,a2,a3),

THNA - таблица счетчиков номеров амперсенодов;

STEK - основной стек, имеется два основных стека STEK1 и STEK2, предназначенных для левого и правого множества,

VSTEK - вспомогательный стек, имеется два вспомогательных стека VSTEK1 и VSTEK2, предназначенных для левого и правого множества, элементы - имена IM2 или IM2, имеющие:

- признаки: "NO" - неотработан;

- графу "МН" (множество), где возможны признаки - "NS, если оно не совпало и "S" - если совпало;

- графу "СИМВ" признак "R" - развернут, признак "S" - свернут;

PRST - признак стека, для левого стека принимает значение "L", для правого - "P";

PRS - признак сворачивания или несворачивания стека, принимает значение "1" и "0" соответственно;

RSR - регистр, хранящий синтерм из VSTEK с признаком "NS" в графе "МН", если он не совпал и "S" - если совпал;

IM - регистр для хранения идентификатора множества, выбираемого из стека для сравнения, имеется IM1 и IM2 исходя из типа стека;

ТСН - текущее значение счетчика "амперсендов" - промежуточных служебных множеств - синтермов, порождаемых алгоритмом; имеет начальное значение 1;

"@четный номер" и "@нечетный номер" - имена "амперсендов"; тут "четный номер " и "нечетный номер " - значения ТСН; синтермы @четный и @нечетный позволяют объединять и пересекать множества, имеющие разные синтермы, но одни и те же базовые термы, т.е. собственно порождаемое описание.

Соглашение 12.

В одном выражении можно использовать совместно термы и синтермы.

2.3. Основные ограничения

Ограничение 1.

Запрещено определять синтермы через самих себя.

Ограничение2.

Порядок просмотра термов и синтермов при последовательном разложении БНФ одинаков как в первом так и во втором множестве.

3. Алгоритм ТМО над БНФ

3.1. Алгоритм записи в стек

Вход: Правая часть множества, записанная в регистр ZAKON. Выход: STEK.

Метод: При записи правой части множества в стек каждый терм записывается в отдельную ячейку стека и совокупность термов накрывается знаком "&". Каждый синтерм записывается так же в отдельную ячейку стека. Знак операции, посредством которого соединены подмножества - в вершине стека.

3.2. Алгоритм разложения

Вход: Элемент множества. Выход: ТОРМ.

Метод: Поиск элемента множества в ТОРМ. Если нашли, то запись правой части в STEK в соответствии с алгоритмом записи в стек, затем дозапись STEK в STEK1 или STEK2, а синтерма, который разложился - в VSTEK1 или в VSTEK2 соответственно, с признаком "R".

Если не нашли, то выдача сообщения, что данное имя - терм.

3.3. Алгоритм сворачивания (восстановления)

Вход: STEK, TCH, TOPM, PRST, VSTEK.

Выход: STEK, THNA.

Метод:

1) Если PRST = L , то наращиваем ТСН на 1 (получаем @нечетный номер), затем наращиваем ТСН еще на 1 (получаем @четный номер). Таким образом, имеем два регистра: @четный номер и @нечетный номер. Смотрим в STEK:

- идут подряд два знака, то первый знак поднимаем, а начиная со второго, анализируем;

- стоит один знак, начинаем анализ этого знака:

а) знак "&" - анализируем признаки группы идентификаторов до следующего знака:

- если все идентификаторы имеют признак "S", то эта группа

уничтожается, синтерм в RSR приобретает признак "S" в графе "МН", и помещается в STEK под первый знак, если он есть или в STEK, если его там нет;

- если встретился хоть один идентификатор с признаком "NS" в графе "МН", то группа идентификаторов уничтожается, синтерм в RSR приобретает признак "NS" в графе "МН" и помещается в STEK под первый знак или в STEK.

б) знак "V":

- если встречаем идентификаторы с признаком "S", то засылаем их в @четный номер. Они должны соединяться посредством знака операции "V", и когда группа идентификаторов просмотрена, то цепочка этих идентификаторов должна заканчиваться признаком конца "%";

- если встречаем идентификаторы с признаком "NS", то засылает их в @нечетный номер аналогично.

После того, как группа "ИЛИ" в STEK1 иссякла, анализируем регистры с амперсендами:

- если @четный номер пуст, а @нечетный номер полон, то уничтожает эти "ИЛИ", присваиваем синтерму из RSR признак "NS" в графе "МН" и засылаем в STEK1 под знак, или в вершину стека;

- если @четный номер полон, а @нечетный номер пуст, то

уничтожаем эти "ИЛИ", присваиваем синтерму из RSR признак "S" в графе "МН" и засылаем в STEK1 под первый знак, или в вершину стека, если знак отсутствует;

- если оба амперсенды не пусты, то записываем в ТОРМ сформировавшиеся четные и нечетные амперсенды, а затем

<имя синтерма> = @нечетный номер V @четный номер %,

где <имя синтерма> - синтерм, который разложился на совпадающие и несовпадающие "ИЛИ", попавшие в соответствующий амперсенд.

Затем эти амперсенды записываются в STEK1 со следующими признаками в графе "МН":

- @четный - с признаком "S";
- @нечетный - с признаком "NS".

Значение счетчиков записываются в таблицу хранения номеров амперсенда: сначала нечетный, а затем четный.

2) Если PRST = P, то

выбираем первый элемент из вершины VSTEK.

Если это знак, то засылаем его в основной стек STEK и выбираем следующий элемент из VSTEK.

Если это не синтерм, или синтерм, который не раскладывался (т.е. не имеет признака "R"), то:

- если PRS = 1, то поднимает знак, находящийся в вершине STEK и переписываем в него все идентификаторы из VSTEK до первого знака или до синтерма с признаком "R";

- если PRS = 0, то все идентификаторы до первого знака или до синтерма с признаком "R" переписываем в STEK.

Все идентификаторы в STEK переписываем с признаками, присущими им в VSTEK. Если производится сворачивание, признаки уничтожаются.

Если это синтерм с признаком "R", то:

- если PRS = 0, дописываем этот синтерм в STEK и заканчиваем работу;
- если PRS = 1, смотрим в VSTEK:

а) идут подряд два знака: первый знак поднимаем, а начиная со второго

уничтожаем все идентификаторы до следующего знака, а вместо них под поднятый знак помещаем элемент из VSTEK, который раскладывался;

б) стоит один знак: тогда уничтожаем все идентификаторы, начиная с этого знака до следующего. Следующий знак поднимаем и под него записываем синтерм; конец работы.

3.4. Алгоритм прямого хода.

Вход: STEK1, STEK2, VSTEK1, VSTEK2, TOPM.

Выход: STEK1, STEK2, VSTEK1, VSTEK2.

Метод: Сравниваем имена множеств.

Если IM1 = IM2 , то множества совпали полностью.

K0:

Выбираем сообщения, что множество 1 совпало с множеством 2 и оканчиваем работу.

Если IM1 # IM2, то раскладываем IM2 и анализируем знак в вершине STEK2:

- знак & - идем на RIM1;
- знак V -

K1:

выбираем первый идентификатор из вершины STEK2, засылаем его в IM2 и сравниваем:

1) если IM1 = IM2, то на K0;

иначе раскладываем IM2 и анализируем знак в STEK2:

K2:

- если & - сворачиваем цепочку по "И" в соответствии с алгоритмом сворачивания и засылаем синтерм в VSTEK2, и на K1;
- знак V - на K1;

2) если IM2 - терм, то анализируем знак в STEK2:

- знак & - на K2;
- знак V - IM2 засылаем в VSTEK2 с признаком "NO" и выбираем следующий идентификатор по "ИЛИ".

Если STEK2 опустел, а IM1 # IM2, то восстанавливаем STEK2, сворачивая VSTEK2, выбираем IM2 из STEK2 и раскладываем.

RIM1:

раскладываем IM1 и анализируем знаки в STEK1 и STEK2:

если & - & - на YMN4,

иначе на INACH.

YMN:

выбираем идентификаторы из STEK1 и STEK2 и запоминаем их соответственно в регистрах IM1 и IM2 и сравниваем:

если IM1 = IM2, то засылаем их с признаками "S" в VSTEK1 и VSTEK2 соответственно и на YMN,

иначе

M3:

раскладываем IM2

- если IM2 - синтерм, то выбираем из STEK2 первый идентификатор, засылаем его в IM2 и на M0;
- если IM2 - терм, то смотрим, какой знак в вершине STEK2:

1) знак & - идем до первого "ИЛИ", или пока STEK2 не станет пуст, выбрасывая все встретившиеся идентификаторы в VSTEK2 с признаком "NS",

заслав туда предварительно IM2 с признаком "NS".

Если нашли "ИЛИ", выбираем первый идентификатор по "ИЛИ" и на M0, иначе

ZV:

восстанавливаем VSTEK2 до первого "S", или пока он не станет пуст (сворачивая). Затем выбираем идентификатор из ??TEK2, засылаем его в IM2, раскладываем IM1 и IM2 и на YMN.

2) знак V - засылаем этот идентификатор в VSTEK2 с признаком "NS" и выбираем следующее "ИЛИ". При этом все встретившиеся "И" засылаем в VSTEK2 с признаком "NO". Если "ИЛИ" уже нет, то на ZV.

M0:

если IM1 = IM2, то

M1:

выбрасываем в VSTEK2 все не просмотренные "ИЛИ" с признаком "NO" до первого знака, или пока STEK2 не станет пуст, затем содержимое IM2 с признаком "S" и все остальные "ИЛИ" до первого "И".

Из STEK1 также выбрасываем в VSTEK1 все не просмотренные "ИЛИ" с признаком "NO" до первого знака, а остальные со своими признаками, затем содержимое IM1 с признаком "S" и все остальные "ИЛИ" до первого знака и на YMN.

Если IM1 # IM2, то на M3.

При выборе идентификаторов из стеков в поиске "И" могут возникнуть ситуации:

1) STEK1 пуст и STEK2 пуст - работает алгоритм обратного хода (идем на OBHOD);

2) STEK1 пуст, а STEK2 не пуст - для STEK1 работает алгоритм сворачивания по "И", а в STEK2 сворачиваем все до идентификатора с признаком "S" (также работает алгоритм сворачивания);

3) STEK1 не пуст, а STEK2 пуст - идем на M;

4) STEK1 не пуст и STEK2 не пуст - продолжаем работу.

M: засылаем несовпавший по "И" идентификатор под знак в STEK1 и выбираем идентификатор из VSTEK1:

- если это терм, то поднимаем знак в вершине STEK1 и помещаем терм туда;

- если это синтерм с признаком "R", то работает алгоритм сворачивания;

- если это знак:

знак & - засылаем его в STEK1;

знак V - первому идентификатору по "ИЛИ" присваиваем признак "NS" в графе "MH" и засылаем его в STEK1, после чего начинает работать алгоритм обратного хода (идем на OBHOD).

При поиске этого первого "ИЛИ" (оно имеет признак "S" в графе "СИМВ"), работает алгоритм сворачивания.

INACH: выбираем идентификаторы из STEK1 и STEK2, засылаем их в IM1 и IM2 соответственно и сравниваем:

- если IM1 = IM2, то идем на M1;

иначе идем на M3.

Примечание: после того, как выбрали очередной идентификатор из под знака, нужно проверять : если следующий знак, то выбранный знак не засылаем снова в STEK, а засылаем в VSTEK после последнего выбранного из STEK идентификатора.

3.5. Алгоритм обратного хода

Вход: STEK1, STEK2, VSTEK1, VSTEK2, TOPM, THNA.

Выход: TOPM.

Метод:

OBHOD: Идем по VSTEK1 в поисках первого идентификатора с признаком "NO" после первого встретившегося идентификатора с признаком "S".

Идентификаторы по "И" с признаком "S" в графе "СИМВ" переписываем в STEK1 с признаком "S" в "МН". При этом, отыскиваем такие же идентификаторы с признаком "S" в VSTEK2, сохраняя признак, а все предшествующие ему идентификаторы в том порядке, в котором анализировались, восстанавливаем в STEK2. Если встречаем синтерм с признаком "R", сворачивающий цепочку по "И", то сворачиваем в соответствии с алгоритмом сворачивания.

Если встретили знак "ИЛИ", переписываем все идентификаторы по "ИЛИ" из VSTEK1 до первого "S" в STEK1. Причем, идентификаторы, имеющие признак "S" в графе "СИМВ", приобретают "S" в графе "МН", а имеющие "NS" в графе "СИМВ", приобретают "NS" в графе "МН". Такие же идентификаторы с признаками "S" или "NS" из VSTEK2 засылаются в STEK2, попутно перебрасывая встретившиеся на пути к ним идентификаторы в STEK2. Идентификаторы с признаком "S" в "СИМВ" приобретают такие же признаки в графе "МН".

Когда встретили первый идентификатор с признаком "NO" после того, как первый встретившийся "S" заслали в STEK1, записываем его и все остальные до первого знака или синтерма с признаком "R" в STEK1, а в правом стеке переписываем все идентификаторы с сохранением присущих им признаков в STEK2 до первого идентификатора с признаком "S", или пока VSTEK2 не станет пуст. Если не нашли больше "S" в VSTEK2, то идем на POISK.

Затем выбираем первый идентификатор с "NO" из STEK1 и засылаем в IM1. Из STEK2 выбираем первый идентификатор с "NO" для сравнения, а все остальные, которые предшествуют ему, восстанавливаем в VSTEK2.

1) Если IM1 = IM2, то в левом стеке переписываем все идентификаторы по "ИЛИ" во вспомогательный стек из основного до первого знака, затем IM1 в VSTEK1 и IM2 в VSTEK2, а затем все остальные идентификаторы до первого "И", а в правом стеке = все идентификаторы до "ИЛИ" с "NO", затем все "NO" ил??ам IM2.

POISK: выбираем из STEK1 идентификатор с признаком "NO" и начинаем сравнивать его со всеми идентификаторами STEK2, пока не дойдем до идентификатора, имеющего признак "S".

2) Если IM1 # IM2, то

S: восстанавливаем STEK2, пока VSTEK2 не станет пуст без сворачивания, раскладываем IM1, выбираем первое идентификатор из STEK1, засылаем его в IM1 и снова сравниваем со всеми идентификаторами STEK2 до идентификатора, имеющего признак "S".

Если IM1 не раскладывается, и не нашлось одинакового ему в STEK2, то смотрим, в каком контексте стоит этот терм:

- если в контексте "И", то

K: сворачиваем эту цепочку по "И", а ближайшему "ИЛИ", т.е. синтерму, свернувшему эту цепочку, даем признак "NS" в "МН" и помещаем рядом с

просмотренными "ИЛИ" в STEK1 или в вершину стека и идем на POISK:

- если в контексте "ИЛИ", то

L: даем ему признак "NS" в "МН" и помещаем рядом с просмотренными "ИЛИ" в STEK1 или в вершину стека и на POISK.

Если IM1 = IM2, то смотрим, в каком контексте находятся IM1 и IM2:

- Если в STEK1 "&" и в STEK2 "&", то выбираем следующие идентификаторы из STEK1 и STEK2 и сравниваем, пока не исчерпаем всю цепочку "И". Если вся цепочка совпала, то сворачиваем в развернувший ее синтерм, даем ему признак "S" как множество, т.е. в "МН", и помещаем рядом просмотренными "ИЛИ" в STEK1 и на POISK. Если хоть одно "ИЛИ" этой цепочки не совпало, то на K.

- Если в STEK1 "V" и в STEK2 "V" , то засылаем в VSTEK1 все "ИЛИ" с "МО", затем все просмотренные, а потом этот IM1 с "С", в правом стеке аналогично и далее - на POISK.

- Если в STEK1 "&", а в STEK2 "V", то на K. - если в STEK1 "V", а в STEK2 "&", то на L.

Процесс прекращаем, когда вся цепочка идентификаторов по "ИЛИ" просмотрена и на P.

Если при поиске первого "NO" в VSTEK1 встречаем синтерм с признаком "R" (при анализе "ИЛИ").

P: ищем в THNA текущее значение счетчика амперсендов и начинает работать алгоритм сворачивания.

После работы алгоритма сворачивания содержимое STEK переписывается в STEK1.

Процесс прекращается, когда в VSTEK1 уже нет идентификаторов с признаком "NO", т.е. VSTEK1 пуст. При этом все идентификаторы с присущими им, или приобретенными при анализе их признаками, попадают в STEK1.

После чего восстанавливаем VSTEK1 и начинаем анализ VSTEK1.

В очередной @четный записываем цепочку идентификаторов, имеющих признак "S" в графе "МН", объединяя их между собой знаком "&". Цепочка должна оканчиваться признаком конца "%" и помещаться в TOPM. Когда VSTEK1 опустел в результате поиска следующего "S" в графе "МН", и все идентификаторы оказались в STEK1, переписываем содержимое STEK1 в VSTEK1.

Идем по VSTEK1 в поисках идентификатора, имеющего в графе "МН" признак "NS", переписывая все встретившиеся идентификаторы в STEK1, а все с признаками "S" в графе "МН" в очередной @нечетный. После того, как нашли первый идентификатор с признаком "NS" , записываем его в @нечетный, а в STEK1 не переписываем. Далее, в @нечетный дописываем все идентификаторы с признаком "S" в графе "МН", а в STEK1 все идентификаторы со своими признаками.

Когда VSTEK опустел, переписываем STEK1 в VSTEK1 и начинаем поиски следующего "NS", действуя аналогично.

Процесс прекращаем, когда в VSTEK1 нет идентификаторов с признаком "NS". После чего выдается на печать вся таблица TOPM. В ней: последний @четный номер - результат пересечения множеств, а последовательность последних @нечетных номеров результат разности множеств. Объединение множеств - это все первое множество плюс разность.

Заключение

В работе предложен алгоритм выполнения теоретико-множественных операций над контекстно-независимыми порождающими грамматиками. Данный алгоритм способен обеспечить работу со знаниями в среде инструментальной оболочки для создания интеллектуальных САПР, что показали реализации [6,3]. Следует отметить, что данный алгоритм имел в свое время полную, но автономную реализацию в среде Р-трана на УВК СМ-4.

В качестве перспективного направления работы следует указать построение алгоритма выполнения теоретико-множественных операций над контекстно-зависимыми порождающими грамматиками, где всякая альтернатива в ИЛИ-синтермах связана впрямую со списком идентификации ряда прототипов.

Литература

1. Григорьев А.В. Представление недоопределенности знаний в инструментальной оболочке для построения САПР. Искусственный интеллект. N 6, 1999, С. 56-66.

2. 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.

3. 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.

4. А.В. Григорьев. Построение процедуры П4 для семиотической модели пространства-времени. Труды Международной научно-практической конференции KDS-2001 "Знание-Диалог-Решение". KDS-2001, Том 1. Санкт-Петербург, издательство "Лань", 2001. - С. 169-177.

5. А.В. Григорьев. Методы построения функций в специализированной оболочке для создания интеллектуальных САПР. Искусственный интеллект. N 3, 2001, С. 40-53.

6. Григорьев А.В., Базалей А.О. Специализированная оболочка для синтеза интеллектуальных САПР и АСНИ. В кн. Информатика, кибернетика и вычислительная техника (ИКВТ-97). Сборник трудов ДонГТУ, Выпуск 1. Донецк: ДонГТУ, 1997. С. 225-228.

7. Солодовников В.В., Тумаркин В.И. Теория сложности и проектирование систем управления. - М.: Наука. 1990. - 186 с.

8. Григорьев А.В. Теоретико-множественные операции над грамматиками как механизм работы со знаниями в интеллектуальных САПР. Труды Восточно-украинского технического университета. Луганск, ВУТУ, 2002. С. 186-194.

9. И.В. Вельбицкий, В.Н. Ходаковский, Л.И. Шолмов. Технологический комплекс производства программ на машинах ЕС ЭВМ и БЭСМ-6. - Москва: Статистика, 1980. - 263 с., ил.

Поступила в редакційну колегію 28.12.2002