УДК 004.896

Комплекс средств и методов работы с формальными грамматиками в семиотической концептуальной модели предметной области интеллектуальных САПР

А.В. Григорьев
1) Донецкий национальный технический университет
grigorievalvl@gmail.com

Григорьев А.В. Комплекс средств и методов работы с формальными грамматиками в семиотической концептуальной модели предметной области интеллектуальных САПР. *Рассмотрен комплекс методов работы с формальными грамматиками, определенных в рамках концептуальной модели предметной области инструментальной оболочки для автоматизации построения интеллектуальных САПР ограниченного класса. Работа носит итоговый характер.*

Ключевые слова: *формальные грамматики, концептуальная модель предметной области, семиотическая модель, САПР.*

Введение

САПР относятся к важнейшим информационным системам. Развитие производства и общества в целом не мыслимо без применения САПР. Лавинообразное увеличение числа предметных областей (ПрО), где требуется создание САПР, ставит проблему автоматизации их создания. Методы и средства систем искусственного интеллекта (СИИ), проникая, в том числе, и в область создания САПР, позволяют сделать решение этой проблемы реальностью. Данная тенденция, к сожалению, пока не получила своего полного воплощения на практике. Т.о., *актуальным* является построение унифицированных средств и методов построения САПР, адаптируемых на технологии проектирования в требуемой ПрО, обеспечивающих более высокую эффективность процесса создания и функционирования как новых интеллектуальных САПР (И САПР), так и модификации существующих САПР до уровня гибридных. Проведенный ранее анализ [] тенденций развития САПР и СИИ в современных условиях позволяет сделать вывод, что *актуальной становится задача* разработки комплексной КМ ПрО САПР как основы для создания специализированной инструментальной оболочки (ИО), предназначенной для построения интеллектуальных САПР уровня АРМ, учитывающей тенденции развития САПР и СИИ и являющейся средством предметной и проблемной адаптации в современных условиях.

При этом данная КМ должна обеспечивать следующие возможности предметной и проблемной адаптации: 1) ИО должна быть способна настраиваться на любую техническую или не техническую ПрО, обладающей достаточно формализованным описанием объекта и методами построения объекта с требуемыми характеристиками; 2) Быть ориентирована на эксперта в ПрО, выполняющего адаптацию САПР, как источник знаний при построении моделей объекта проектирования и проектных процедур и быть способной адаптироваться на специфический уровень его квалификации; 3) Адаптироваться на данную ПрО с характерными для нее типами фазовых переменных и координат взаимодействия; 4) Быть ориентированной на построение набора требуемых типов изделий в данной ПрО; 5) Формировать состав модельных АУ требуемой полноты; 6) В рамках каждого АУ формировать набор моделей, проектных процедур и критериев требуемой полноты; 7) ИО должна при построении САПР на каждом АУ, задавая набор процедур, моделей и критериев, обеспечить способность задать иерархические, регулярные структуры, реализующие регулярные же, иерархически организованные функции, специфичные для данной ПрО; 8) Иметь возможность адаптироваться на достигнутый уровень воплощения методик проектирования для существующих САПР в данной ПрО, т.е. – быть ориентирована либо на построение нового автономного интеллектуального САПР (И САПР) в случае отсутствия П САПР, либо – интеллектуальной надстройки над существующим САПР, добавив в нее те или иные проектные процедуры, повысив тем самым уровень воплощения методик проектирования; 9) ИО должна обеспечивать построение либо всех процедур САПР целиком в форме ЭС, либо - построение САПР как гибридной.

Построение данной КМ выполнено автором в рамках работ [1-30]. Данная КМ является основой построения соответствующего инструментального комплекса по автоматизации построения интеллектуальных САПР некоторого ограниченного класса – мета-эвристической оболочки (МЭО).

Наибольшее влияние на методы построения КМ оказала выбранная форма ее представления – семиотическая модель. А так как главный компонент СМ – это формальные грамматики, то без преувеличения можно сказать, что суть КМ – это набор методов работы с формальными грамматиками, определенными в рамках СМ.

Цель работы: описать структуру средств и методов работы с формальными грамматиками в рамках С КМ ПрО МЭО.

1. Общие принципы построения С КМ ПрОб МЭО

Ранее в работах [1,2] был определен ряд принципов построения инструментальной мета-эвристической оболочки (МЭО), предназначенной для построения интеллектуальных САПР. К главным особенностям МЭО относится:

1) работа с моделями [3,4,5] сложных объектов различной степени недоопределенности.

2) ориентация при обучении базы знаний комплекса на ограниченное количество имеющихся в наличии апробированных на практике моделей сложных объектов данной предметной области;

Ограниченное число моделей сложных объектов предполагает необходимость автоматизации синтеза возможных решений с различной степенью их определенности с последующим отсевом выбором имеющих смысл решений с целью ускорить их ввод в базу знаний САПР.

Структура МЭО и технология работы с МЭО проиллюстрированы на рис. 1.

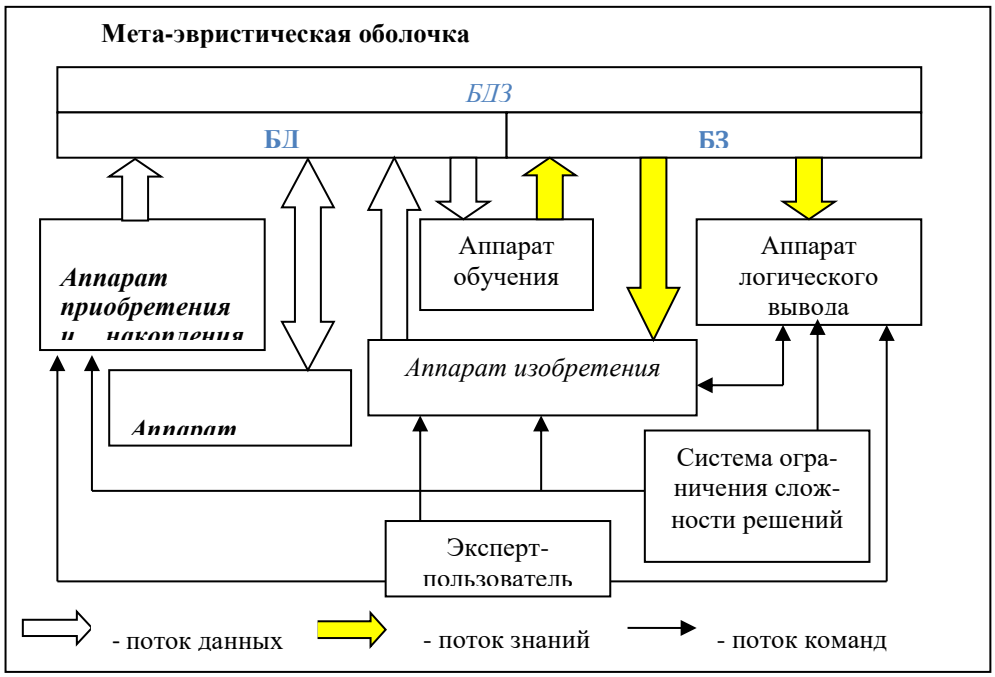

Рисунок 1 - Структура мета-эвристической оболочки

Кратко опишем назначение некоторых элементов комплекса:

1) Аппарат моделирования – обеспечивает нахождение недоопределенной информации предметной области, путем рассмотрения данной задачи на динамических недоопределенных вычислительных моделях [31,32,33].

2) Аппарат изобретения – обеспечивает создание новых объектов некоторого заданного типа.

3) Система оценки и ограничения сложности моделей – оценивает и ограничивает сложность объектов для любых предметных областей (ПрОб) как при вводе апробированных (достоверных) объектов их проблемно-ориентированных САПР или при прямом вводе объектов пользователем, а так же при синтезе (изобретении) объектов. Запрещает рассмотрение не имеющих смысла или слишком сложных для анализа данным пользователем объектов, повышая тем самым скорость процесса изобретения.

Общие принципы построения С КМ

ПрОб МЭО:

1. Построение САПР решения типичных задач проектирования, где имеет место большое, но ограниченное множество возможных решений;
2. Физическая семантика ПрОб;
3. И-ИЛИ-дерево как форма контекстно-независимой грамматики описания множества решений структур или функций объекта некоторого типа;
4. Семиотическая модель как форма представления КМ ПрОб;

Рассмотрим данные принципы детальнее.

1.1. Построение САПР решения типичных задач проектирования

Существует ряд определений САПР, что связанно со сложностью этого понятия. Примеры определений САПР: как комплекс средств, как совокупность проектных процедур, как набор этапов разработки документации и т.д. [34]. Однако, в качестве главного определения можно рассматривать процедурное определение САПР. В соответствии с ним САПР представляет собой ряд моделей объекта проектирования возрастающих уровней абстракции (АУ), связанных проектными процедурами, модифицирующими данные модели под управлением соответствующих критериев (см. рис. 2).

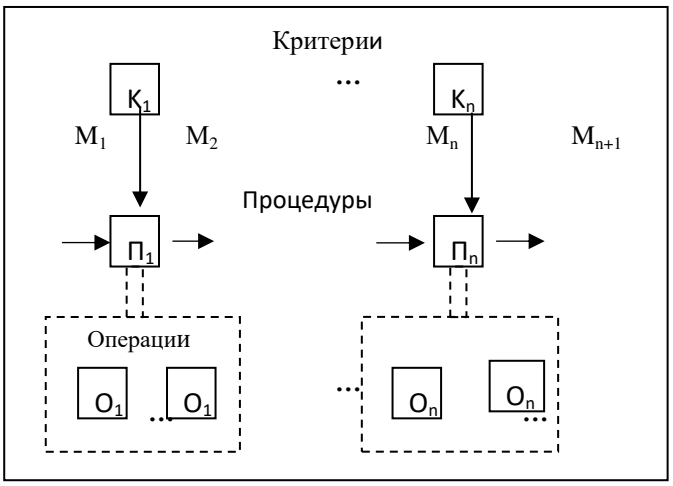

Рисунок 2 - Структура САПР

Совокупность критериев составляет в целом ТЗ на изделие. Набор процедур определяет методику проектирования. Каждая процедура есть комплекс проектных операций. Каждому АУ соответствует некоторое подмножество процедур и связанных с ними форм представления моделей объекта проектирования. Типичные проектные процедуры, относящиеся к любому АУ, это: синтез (выбор) модели, редактирование, верификация (моделирование), документирование. Типичные формы представления моделей объекта проектирования: система уравнений, документ, графический образ, таблица, текст на некотором языке программирования и т.д. В САПР используются такие АУ моделей проектирования и, соответственно, уровней наборов проектных процедур:

- структурный (полное отсутствие модели функций в модели объекта проектирования);
- системный (уровень потоков заявок как модели функций объекта);
- функционально-логический (уровень дискретных, логических уравнений, составляющих модель объекта);
- количественный, включая подуровень макро-моделей, т.е. моделей с сосредоточенными параметрами (уравнений Кирхгофа) и подуровень микро-моделей, т.е. моделей с рассосредоточенными параметрами (уровень дифференциальных уравнений в частных производных как моделей объекта).

Перечисленные АУ уровни упорядочены по возрастанию сложности и отличаются:

- полнотой представления модели пространства и времени (наличием или отсутствием, дискретным или непрерывным представлением);
- полнотой отражения совокупности фазовых переменных (поток и потенциал) и координат взаимодействия (емкость, индуктивность и т.д.);
- полнотой состава дополнительных свойств и отношений (функций), отделенных в модели объекта (топологические,

функциональные, энергетические и т.д.);

- полнотой представления законов сохранения (энергии, вещества и т.д.), выраженных посредством набора отношений.

На практике это приводит к тому, что САПРы могут решать два класса задач, отличающихся размером пространства поиска решений проектных процедур:

- задачи типичного проектирования, т.е. – выбора подходящего решения из ряда известных;

- задачи поискового конструирования, т.е. - поиска неизвестного ранее решения (изобретения).

1.2. Физическая семантика КМ ПрОб МЭО

Содержательная формулировка КМ ПрОб МЭО, соответствующая физической семантике ПрОб:

1) жизненный цикл объекта представляется как совокупность линейных векторных пространств, соответствующих числу необходимых тактов времени;

2) линейные векторные пространства связаны набором "временных" связей;

3) элементами линейных векторных пространств являются пространственные координаты, связанные совокупностью "пространственных" связей, определенных внутри "временных" связей;

4) пространственная координата может рассматриваться как физическая точка (ФТ), в случае, если состав и значения ее свойств отличается от "неопределено";

5) свойства ФТ возникают только как факт отражения существования "простых" связей между ФТ внутри "пространственных" связей;

6) "собственное" свойство - идентификатор ФТ есть некоторая фазовая переменная данной ПрОб, задающая потенциал точки для некоторой субстанции; задается посредством "кольцевого" отношение над ФТ;

7) совокупность "собственных" свойств ФТ определяется уровнем представления модели и ПрОб;

8) "чужое" свойство ФТ - задание факта наличия связи - отношения, задающего меру влияния потенциала другой ФТ на потенциал данной ФТ; задается посредством "простой" связи над парой ФТ;

9) модель функции ФТ задается табличным образом как множество возможных комбинаций значений свойств-потенциалов ФТ (по всем временным, пространственным и "простым" связям), когда-либо имевшим место на практике;

10) отношение зависимости связанной пары потенциалов "свой" - "чужой", выделенное внутри модели функции ФТ, определяет меру влияния "чужого" потенциала на "свой" потенциал ФТ и задает поток, направленный на уравнивание потенциалов во времени; данное отношение задается как функция "простой" связи, определенной внутри "пространственной" связи;

11) отсутствие в наборе свойств некоторой ФТ1 "чужого" свойства ФТ2 говорит о невозможности влияния потенциала ФТ2 на потенциал ФТ1 (отсутствующая или однонаправленная связь);

12) функция "простой" связи определяется таблично как совокупность историй поведения ее во времени и косвенно задает параметры связи (емкость, сопротивление и т.д.);

13) модель функции ФТ строится в соответствии с законами сохранения движения и энергии;

14) модели функций ФТ и "простых" связей задают функциональные и энергетические связи, простые" связи задают структурные и причинно-следственные связи, пространственные связи задаются явно (см. выше);

15) вещественные связи, связанные с переносом вещества, определяются как изменение ФТ своих пространственных координат и строятся в соответствии с законом сохранения массы;

16) переход на новый уровень представления модели предполагает не только декомпозицию избранного набора ФТ (блоков) и свойств на более мелкие, но и увеличение числа этих точек, т.е. изменение (расширение) вышележащих уровней декомпозиции.

1.3. И-ИЛИ-дерево как форма контекстно-независимой грамматики описания множества решений структур или функций объекта некоторого типа;

1. И-ИЛИ-дерево как аппарат представления БЗ

Опишем принятый в среде мета-эвристической оболочки метод преставления знаний [1]. Основой данного метода является использование И-ИЛИ-дерева как аппарата представления БЗ.

Базой знаний в системе является:

- И-ИЛИ-дерево, то есть упорядоченное множество вершин по «И» и по «ИЛИ», при этом все вершины имеют свой адрес в И-ИЛИ-дереве;
- Множество семантических двунаправленных зависимостей между ИЛИ-вершинами, описывающими их совместимость

между собой, то есть множество правил вывода или продукций.

Представление данных в виде И-ИЛИ-дерева необходимо для выполнения теоретико-множественных операций над БЗ системы в процессе обучения и вывода. Поиск решений в системе ведется по И/ИЛИ-дереву, которое состоит из единиц информации, представленных некоторой структурой.

Под И-ИЛИ-деревом понимается некоторый связный граф, не содержащий циклов и имеющий иерархическую многоуровневую структуру. Вершинам И-ИЛИ-дерева может быть дана следующая трактовка, выполненная с точки зрения терминов теории формальных грамматик [2]. Любая вершина рассматривается либо как терм либо как синтерм. Терм определяется как элементарный терминальный символ множества и включает оригинальный фрагмент описания некоторого объекта(ов) на некотором языке. Термы соединяются между собой только посредством операции "И" (&). Синтерм, т.е. нетерминальный символ, задается как имя множества, которое может в дальнейшем раскладываться. Элементами разложения могут быть как термы, так и синтермы, соединенные посредством операции "И" (&) или "ИЛИ" (V). Синтермы всегда записываются в соответствии с нотацией Бекуса-Наура только в угловых скобках "<>". Очевидно, что термы будут являться только листьями дерева, которые, не могут иметь “сыновей” в данном дереве.

Для избранного подхода к представлению И-ИЛИ-деревьев справедливы следующие утверждения:

- Каждая вершина или узел дерева представляет собой описание фрагмента объекта;
- Каждая полная цепочка от вершины дерева до некоторого листа есть объект, состоящий из фрагментов, которые представлены узлами входящими в данную цепочку;
- Синтермы, имеющие одного «отца», является альтернативами по отношению к друг другу;
- Отношения между ИЛИ связывает те термы, комбинация которых принадлежат некоторому непустому множеству семантически верных (проверенных) объектов, имеющих место в И-ИЛИ-дереве;
- Прототипы или верифицированные объекты есть основа построения И-ИЛИ-дерева;
- И-ИЛИ-дерево есть средство:

- компактной записи множества известных объектов;
- порождения гипотез о возможных новых объектах;

- И-ИЛИ-дерево – это множество синтаксически правильных выражений;
- Продукции определены над И-ИЛИ-деревом и задают правила вывода, которые в совокупности позволяют вычленить из И-ИЛИ-дерева семантически верное подмножество, то есть те же самые аксиомы-объекты.

1.4. СМ как форма представления КМ ПрОб

Средства построения формальной модели.

Открытый характер базы знаний САПР требует использовать для ее построения семиотическую модель (СМ). СМ представляет собой открытую формальную систему и имеет форму восьмерки [36]:

F = <T,C,A,П,r,b,g,d>, где:

T - множество базовых элементов системы, на которых строятся все выражения в ***F;***

C - множество правил построения синтаксически правильных формул, определяющих среди всех возможных выражений из базовых элементов те, которые синтаксически правильны;

A - множество аксиом ***F***, образующее подмножество в множестве синтаксически правильных формул, которым априорно присваивается статус истинности;

П - множество правил вывода, или семантические правила, (позволяющие получать из аксиом новые синтаксически правильные формулы, которым можно приписывать статус истинности);

r, ***b***, ***g***, ***d*** - правила изменения, соответственно для ***T***, ***C***, ***A*** и ***П***.

Для конструктивности семиотической модели требуется реализация следующих классов процедур:

П1 - определения принадлежности данного элемента множеству ***T;***

П2 - идентификации различия элементов множества **T**;

П3 - определения синтаксической корректности элементов, построенных посредством правил ***C***.

Процедуры ***П1***, ***П2*** и ***П3*** должны быть конструктивными, т.е. завершаться через определенное число шагов.

Конструктивная СМ является разрешимой, если существует конструктивная процедура ***П4***, дающая однозначный ответ на вопрос - является ли данный синтаксически корректный элемент семантически верным.

СМ может рассматриваться как форма представления концепции "возможных миров" Крипке. Проблема построения разрешимой СМ

в общем случае пока не решена.

2. Общая структура средств и методов работы с формальными грамматиками в рамках СМ

Общая структура включает в себя следующие методы и средства работы с формальными грамматиками в рамках СМ:

1) Методы построения **системы вложенных формальных проблемно-независимых языков спецификаций** на базе физической семантики ПрОб с привлечением аппарата НЕ-факторов;

2) Комплекс **методов задания семантических зависимостей** над контекстно-свободными грамматиками, адаптированные к различным условиям их применения;

3) Комплекс алгоритмов **выполнения теоретико-множественных операций** над грамматиками, адаптированные к различным условиям их применения;

4) **Методы обучения,** т.е. – построения базы знаний, на базе ТМО, адаптированные к различным условиям их применения;

5) Методы организации **логического продукционного вывода** на базе ТМО, адаптированные к различным условиям их применения;

6) Метод **оценки сложности** выполнения теоретико-множественных алгоритмов над грамматиками различных типов;

7) Методы преобразования грамматического описания объекта, имеющего **не допустимую когнитивную сложность** представления, к виду, имеющую допустимую форму представления когнитивной сложности;

8) **Изобретение новых решений** – как инструмент сужения числа аксиом грамматики, построенной путем обобщения прототипов;

9) Построение системы **интерфейсов** «Язык предметной области ↔ Язык формальных спецификаций соответствующего уровня абстракции» с цель обеспечения.

Рассмотрим их детальнее перечисленные компоненты.

3. Методы построения системы вложенных формальных проблемно-независимых языков спецификаций на базе физической семантики ПрОб с привлечением аппарата НЕ-факторов

СМ КМ ПрОб включает 6 обязательных уровней: 1) исходной модели; 2) задания времени как блока и свойства; 3) значений свойства времени и моделей пространств; 4) пространственных точек и их идентификаторов; 5) «простых» свойств и внутренних функций блоков; 6) значений "простых" свойств и "кортежей" функций. Все уровни описываются, исходя из общих аксиом, сформулированных в [10]. Рассмотрим последовательно все перечисленные уровни. Правила преобразования семантически верных выражений (правила **g** для **A**), соответствующих **Ti-**му отношению сигнатуры, будем обозначать как **Gi** и формулировать их по мере описания системы уровней.

3.1. Уровень исходной модели

Исходная модель предмета уровня 1 (рис. 3) включает блок с именем, но без внутренней структуры, имеющим единственное недоопределенное свойство без имени и структуры, единственную "круговую" связь, замыкающую блок сам на себя. Зададим состав семантически верных отношений исходного уровня (аксиомы ЛА1-ЛА4), задающих описание глобальной аксиомы - *прототипа*, описывая одновременно синтаксис отношений:

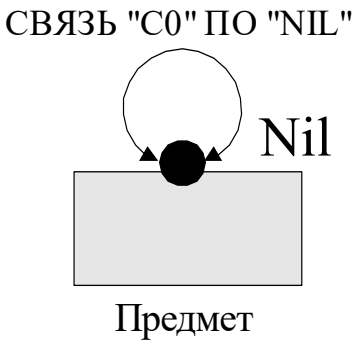

Рисунок 3 - Изначальное представление модели предмета

ЛА1. Единственный недоопределенный блок – модель предмета, имеет идентификатор "***П***", но не имеет структуры - $\boldsymbol{Б = \&\{\, П \,\}}$.

ЛА2. Единственное недоопределенное свойство не имеет идентификатора и структуры: $\boldsymbol{Д = \&\{\, Nil_1^Д \,\}}$.

ЛА3. Внешняя граница блока ***П***: $\boldsymbol{G[\, П \,] = \{\, П.Nil_1^Д \,\}}$, где точка обозначает отношение принадлежности свойства блоку.

ЛА4. Среда или множество *связей* уровня (рис. 1) имеет вид: $\boldsymbol{L_П = \&\{\, l_1^П \,\}}$. Структура связи: $\boldsymbol{l_1^П = П.Nil_1^Д \Leftrightarrow П.Nil_1^Д}$, где $\Leftrightarrow$ обозначает двунаправленное отношение передачи информации между блоками через свойства их внешних границ. Будем обозначать связь $\boldsymbol{l_1^П}$ как С0. Принятая выше грамматика описания отношений в ЛА1-ЛА4 входит в *множество синтаксически верных отношений,* составляющих прототип.

ЛА5. Модель прототипа уровня 1 не имеет альтернативных форм представления.

ЛА6. Перечисленный состав блоков, свойств, границ и связей задает набор системообразующих элементов модели данного уровня.

3.2. Уровень задания времени - блока и свойства

Уровень 2 предполагает декомпозицию исходного свойства $\boldsymbol{Nil_1^Д}$ блока ***П*** на три подсвойства - обратный элемент, время ***Т*** и неопределенность. Одновременно выполняется декомпозиция блока ***П*** на обратный элемент, блок ***Время*** (в дальнейшем просто *"**В**"*) и неопределенность (рис. 4). Тут ***В*** - собственное свойство блока ***Т***.

Множество связей $\boldsymbol{L_B = \&\{\, l_k^B \,\}_{k=1}^6}$, является декомпозицией связи $\boldsymbol{l_1^П}$. Структура связи $\boldsymbol{l_1^П}$ предполагает связь подблока ***В*** с внутренней границей блока ***П*** (т.е. $\underline{\boldsymbol{П}}$), затем связь между внутренней и внешней границами блока ***П*** и затем связь блока ***П*** самого с собой (внутри "старой" связи С0):

$$\begin{aligned} &\boldsymbol{l_1^B = П(\,B\,).(\,Nil_1^Д(\,T\,)) \Leftrightarrow П(\,\underline{П}\,).(\,Nil_1^Д(\,T\,)) \Leftrightarrow П.(\,Nil_1^Д(\,T\,))} \\ &\boldsymbol{\Leftrightarrow П.(\,Nil_1^Д(\,T\,)) \Leftrightarrow П(\,\underline{П}\,).(\,Nil_1^Д(\,T\,)) \Leftrightarrow П(\,B\,).(\,Nil_1^Д(\,T\,))} \end{aligned} . \quad (1)$$

Упустив промежуточные этапы, выражение (1) можно переписать как:

$$\boldsymbol{l_1^B = П(\,B\,).(\,Nil_1^Д(\,T\,)) \Leftrightarrow П(\,B\,).(\,Nil_1^Д(\,T\,))}. \quad (2)$$

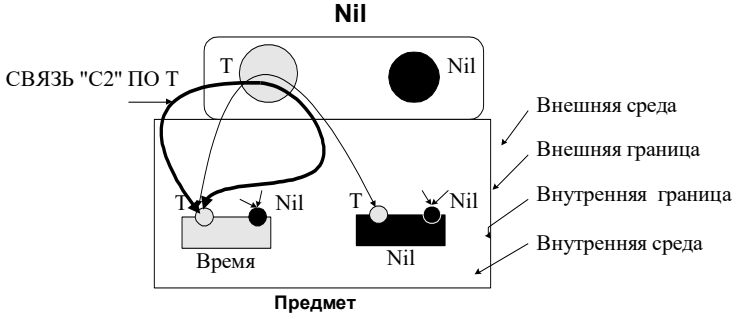

Рисунок 4 - Уровень ввода времени как свойства и блока

3.3. Уровень значений свойства времени и моделей пространств

Уровень 3 предполагает декомпозицию свойства ***Т*** на совокупность дискретных значений времени и одновременную декомпозиция блока ***В*** на совокупность моделей пространств, соответствующих отдельным значения времени (рис. 5).

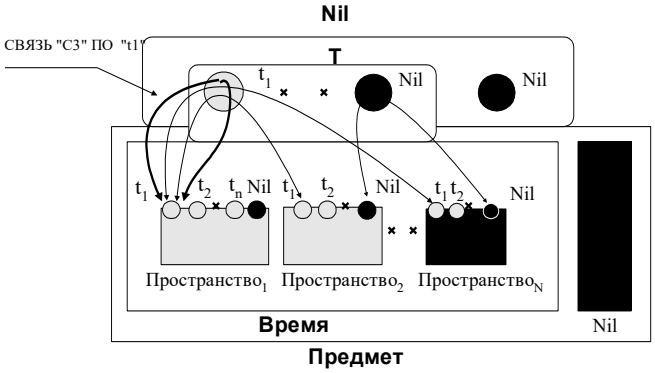

Рисунок 5 - Уровень определения моделей пространств

3.4. Уровень пространственных точек и их идентификаторов

Уровень 4 предполагает перевод всех значений свойства ***T*** в разряд новых свойств и дальнейшую их декомпозицию на ряд собственных значений, в качестве которых выступают идентификаторы физических точек пространства. Одновременно выполняется декомпозиция моделей пространств ***P*** на блоки - физические точки пространств (рис. 6).

Структура пространственной связи:

$$l_{kv}^{Tx} = P_{i1}^{k}(X_{r1}^{k}).t_{c}^{k}(x_{u}^{v}) \Leftrightarrow P_{i2}^{k}(X_{r2}^{k}).t_{c}^{k}(x_{u}^{v}); \forall r1, r2. \quad (3)$$

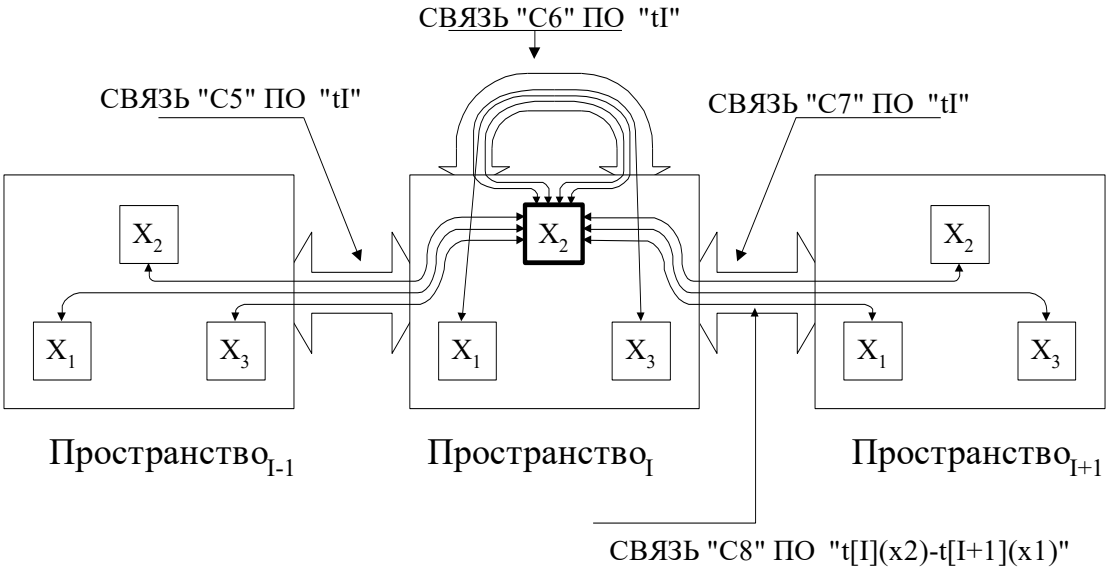

Рисунок 6 - Фрагмент совокупности связей ПТ

3.5. Уровень «простых» свойств и внутренних функций блоков

Уровень 5 предполагает перевод всех идентификаторов пространственных координат из разряда значений в разряд новых свойств и дальнейшую их декомпозицию на ряд собственных значений - идентификаторов "простых" свойств физических точек пространства, а так же декомпозицию моделей блоков - физических точек пространств на множество блоков - носителей простых свойств или внутренних функций (ВФ). Каждое "простое" свойство есть потенциал ПТ в данной предметной области (гидро, тепло и т.д.). ВФ наследует все свойства и связи ПТ. Совокупность свойств внешней границы отдельной ПТ определяется множеством ее связей (табл. 1).

Таблица 1. Пример состава свойств ПТ X_2^0.

t_{-1}^1						t_0^2						t_{+1}^3					
x_1^1		x_2^2		x_3^3		x_1^4		x_2^5		x_3^6		x_1^7		x_2^8		x_3^9	
s_1^1	s_2^2	s_1^3	s_2^4	s_1^5	s_2^6	s_1^7	s_2^8	s_1^9	s_2^{10}	s_1^{11}	s_2^{12}	s_1^{13}	s_2^{14}	s_1^{15}	s_2^{16}	s_1^{17}	s_2^{18}

В этом примере верхние индексы нумеруют свойства в пределах всего множества свойств данного уровня, а нижние индексы - в пределах вышележащей структуры данных. Т.о., внешняя граница ВФ:

$$\forall i,j,c,u : G[P_i\ (X_j^i(F_z^{ij}))] = \&\{P_i\ (X_j^i(F_z^{ij}))t_c^j(x_u^c(s_h^u))\}_{h=1}^{N_h^{ijcu}}; \quad (4)$$

$$\{t_c^{ij}(x_u^c(s_h^u))\}_{h=1}^{N_h^{ijcu}} \subseteq t_c(x_u^c). \quad (5)$$

G4) Множество «простых» связей внутри пространственных связей: $l_{kv}^{Tx} = \{l_{kve}^{Txs}\}_{e=1}^{N_e^{kv}}$. Каждая отдельная "простая" связь имеет вид:

$$l_{kve}^{Txs} = P_{i1}^k(X_{r1}^k).t_c^k(x_u^v(s_b^e)) \Leftrightarrow P_{i2}^k(X_{r2}^k).t_c^k(x_u^v(s_b^e)); \forall r1, r2. \quad (6)$$

3.6. Уровень значений "простых" свойств и "кортежей" функций

Выполняется перевод всех идентификаторов "простых" свойств ПТ из разряда значений в разряд новых свойств и дальнейшую их декомпозицию на ряд собственных значений, т.е. идентификаторов значений "простых" свойств, а так же декомпозиция ВФ на блоки - носители значений "простых" свойств - "кортежи".

3.7. Выводы по разделу

Предложенная система моделей фактически задает систему вложенных языков формальных спецификаций описанного выше вида.

4. Комплекс методов задания семантических зависимостей над контекстно-свободными грамматиками, адаптированные к различным условиям их применения

В данный комплекс входят:

- **атрибутные грамматики**, где каждый терм или синтерм грамматики имеет список атрибутов, задающих его принадлежность к именованному прототипу решения; используются для задания множеств прототипов малой мощности;

- **отношения совместности-несовместности над ИЛИ-узлами** контекстно-свободной грамматики, моделирующие в обобщенном виде атрибутные грамматики, и косвенно моделирующих атрибутные грамматики; используются для задания множеств прототипов средней мощности;

- **контекстно-свободные грамматики с явным использованием НЕ-фактора**, идентифицирующего неопределенный компонент грамматики, играющие совместно с предикатом «Существует» и возможным отрицанием роль посылок и выводов в явно заданных продукционных зависимостях, так же косвенно моделирующих атрибутные грамматики; используются для задания множеств прототипов большой мощности.

Конкретные примеры некоторых грамматик будут показаны в рамках последующих разделов.

5. Комплекс алгоритмов выполнения теоретико-множественных операций над грамматиками, адаптированные к различным условиям их применения

Автором предложен целый комплекс алгоритмов выполнения теоретико-множественных операций над грамматиками, адаптированные к различным условиям их применения.

Условия применения включают:

3.1) Форму грамматик:

- атрибутные грамматики;

- грамматики с отношениями совместности-несовместности;

- грамматики с НЕ-фактором;

3.2) Семантику грамматик:

- грамматики с неизвестной семантикой;

- грамматики с известной семантикой;

Ниже, в качестве примера, приведен алгоритм выполнения ТМО над грамматиками с неизвестной семантикой и синтаксисом.

5.1. Специфика алгоритм выполнения ТМО над грамматиками с неизвестной семантикой и синтаксисом

Специфика предлагаемого подхода к представлению знаний в специализированной инструментальной оболочке для создания интеллектуальных САПР – мета-эвристической оболочке (МЭО) описана ранее в [10-22] 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.:

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

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

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

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

6) 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) [35] на базе ТЗ, 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 [13] 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.

Опишем метод решения задач 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.

5.2. 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 (Вельбицкий И.В.) 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).

5.3. Основные определения, соглашения и ограничения

5.3.1. 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.

5.3.2. 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),

TOPM - 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 "MH" (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 "MH", 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;

TCH - 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 TCH; 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.

5.4. 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.

5.5. Алгоритм ТМО над БНФ

5 5.1. 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.

5.5.2. Алгоритм разложения

Вход: Элемент множества. Выход: TOPM.

Метод: Поиск элемента множества в TOPM. 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.

5.5.3. Алгоритм сворачивания (восстановления)

Вход: STEK, TCH, TOPM, PRST, VSTEK.

Выход: STEK, THNA.

Метод:

1) Если PRST = L , то наращиваем TCH на 1 (получаем @нечетный номер), затем наращиваем TCH 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 "MH", 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 "MH", то группа идентификаторов уничтожается, синтерм в RSR приобретает признак "NS" в графе "MH" 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.

5.5.4. Алгоритм прямого хода.

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

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 "CИMB"), 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.

5.5.5. 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 "CИMB" переписываем в STEK1 с признаком "S" в "MH". 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 "CИMB", приобретают "S" в графе "MH", а имеющие "NS" в графе "CИMB", приобретают "NS" в графе "MH". 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 "CИMB" приобретают такие же признаки в графе "MH".

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" и идем на 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 "MH" 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 "MH" 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 "MH", 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 "MO", затем все просмотренные, а потом этот IM1 с "C", 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 "MH", объединяя их между собой знаком "&". Цепочка должна оканчиваться признаком конца "%" и помещаться в TOPM. Когда VSTEK1 опустел в результате поиска следующего "S" в графе "MH", и все идентификаторы оказались в STEK1, переписываем содержимое STEK1 в VSTEK1.

Идем по VSTEK1 в поисках идентификатора, имеющего в графе "MH" признак "NS", переписывая все встретившиеся идентификаторы в STEK1, а все с признаками "S" в графе "MH" 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 "MH", 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.

6. Методы обучения, т.е. – построения базы знаний, на базе ТМО, адаптированные к различным условиям их применения

Назовем методы обучения, т.е. – построения базы знаний, на базе ТМО, адаптированные к различным условиям их применения:

- построение атрибутных грамматик путем автоматического выполнения теоретико-множественного объединения текстов отдельных решений - прототипов;
- прямого ввода контекстно-свободной грамматики и отношений совместности-несовместности для соответствующего типа грамматики;
- прямого ввода множеств операций пересечения, объединения, поиска разницы и дополнения для грамматик с НЕ-факторами – как способа построения пространства поиска решений.

Ниже приведен пример построения атрибутных грамматик путем автоматического выполнения теоретико-множественного объединения текстов отдельных решений – прототипов.

6.1. Построение атрибутных грамматик путем автоматического выполнения теоретико-множественного объединения текстов отдельных решений – прототипов

Пусть дано множество прототипов, входящих в тип блоков А: $A'=(P_1 \vee P_2 \vee P_3)$. На рис. 7 приведена обобщенная схема типа А, построенная в рамках предлагаемой концептуальной модели.

В табл. 2 представлено описание компонентов данной обобщенной модели как совокупности отдельных структурных связей между подблоками данного типа блоков.

Формат описания:

1) Общая структурная связь, принадлежащая всем вариантом структуры блока (системообразующий компонент) или
   - принадлежащий некоторому подмножеству вариантов структуры блока (прототипы.), т.е. – факультативный компонент.
2) Список принадлежности данного компонента тем или иным прототипам структуры блока, если компонент – факультативный.

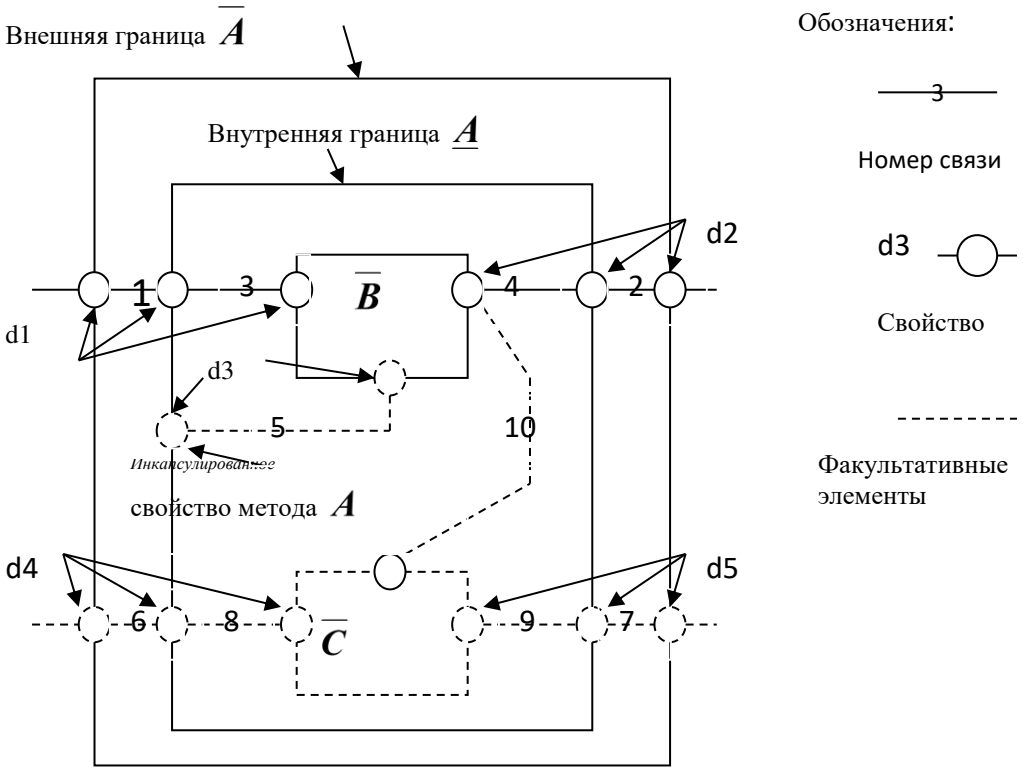

Рисунок 7 - Обобщенная схема типа А

Таблица 2 - Описание множества связей в типе блока.

Номер связи	Описание связи	Системобр./ Факульт-я.	Принадлежность прототипам
1	$\overline{\underline{A}} : d1 \leftrightarrow \underline{A} : d1$	С	1,2,3
2	$\overline{\underline{A}} : d2 \leftrightarrow \underline{A} : d2$	С	1,2,3
3	$\underline{A} : d1 \leftrightarrow \overline{\underline{B}} : d1$	С	1,2,3
4	$\underline{A} : d2 \leftrightarrow \overline{\underline{B}} : d2$	С	1,2,3
5	$\underline{A} : d3 \leftrightarrow \overline{\underline{B}} : d3$	Ф	1,2
6	$\overline{\underline{A}} : d4 \leftrightarrow \underline{A} : d4$	Ф	2,3
7	$\overline{\underline{A}} : d5 \leftrightarrow \underline{A} : d5$	Ф	2,3
8	$\underline{A} : d4 \leftrightarrow \overline{\underline{C}} : d4$	Ф	2,3
9	$\underline{A} : d5 \leftrightarrow \overline{\underline{C}} : d5$	Ф	2,3
10	$\overline{\underline{B}} : d3 \leftrightarrow \overline{\underline{C}} : d3$	Ф	2,3

Результат выполнения теоретико-множественных операций над совокупностями связей, образующими данные прототипы, показан на рис. 8.

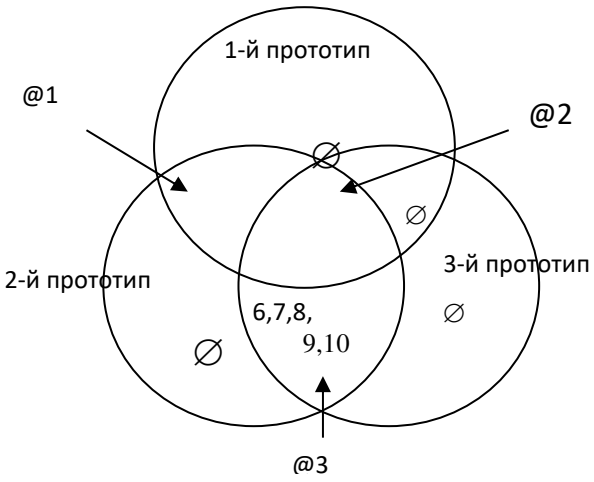

Рисунок 8 - Выделение частей прототипов

Тут: @j - некоторая часть внутренней среды прототипов; 1,2… - номера связей, ∅ - пустое множество связей. При этом:

P_1=@1&@2; P_2=@1&@2&@3; P_3=@2&@3;@1=5; @2=1&2&3&4; @3=6&7&8&9&10. (7)

Преобразуем множество прототипов А к форме И-ИЛИ-дерева:

$A' = (P_1 \vee P_2 \vee P_3)$ = @2 & H1; H1=@1$\vee$ @3 $\vee$ H2; H2=@1&@3.

На рис. 9 изображено полученное И/ИЛИ дерево. В скобках показаны номера прототипов, входящих в данную вершину, числами заданы номера связей, стрелками показан порядок декомпозиции узлов.

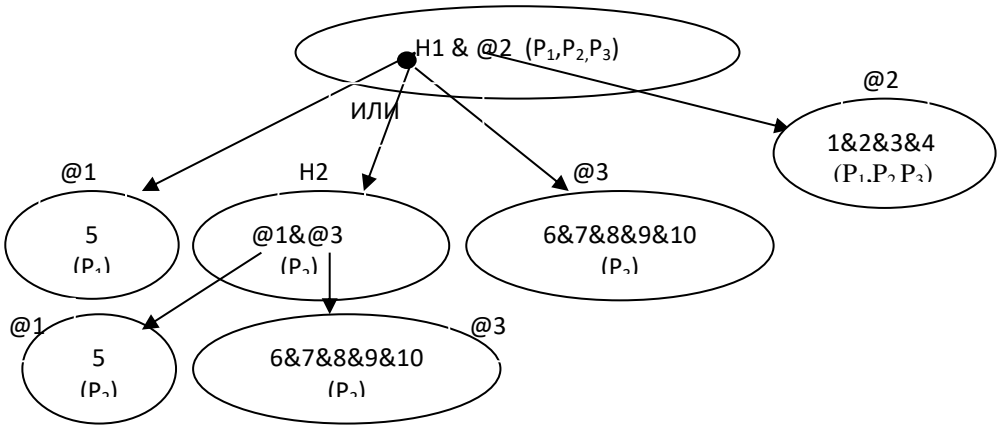

Рисунок 9 - Форма И-ИЛИ-дерева

7. Методы организации логического продукционного вывода на базе ТМО, адаптированные к различным условиям их применения

Структура комплекса методов использования ТМО такова:

- усечения атрибутных грамматик путем использования комплекса стандартных правил вывода;
- усечения грамматик с отношениями совместности – несовместности путем использования комплекса стандартных правил вывода;
- пересечения, объединения, поиска разницы и дополнения для грамматик с НЕ-факторами, выполняемых в рамках заданного множества продукций.

Фактически, речь идет реализации алгоритма функционирования процедуры П4 управления выводом в СМ. Ниже, в качестве примера, приведен вариант процедуры П4 для реализации пространственно-временной логики в рамках общего метода усечения атрибутных грамматик путем использования комплекса стандартных правил вывода.

7.1. Вариант П4, обобщенный: часть прототипов известна, а часть - нет.

Пусть дано:

1) ТЗ - часть значений X'_{ij} для ряда T_j; $X'_{ij} = \{ X'^{m}_{ij} \}^{M'ij}_{m=1}$, но собственно искомые прототипы - неизвестны;

2) Куча - совокупность $X_{ij} = \{ X^{m}_{ij} \}^{Mij}_{m=1}$, при том, что верно $X^{m}_{ij} \rightarrow \{ \Pi^{1}_{i},.., \Pi^{Ki}_{i}, Nil \}$, т.е. имеется частичное соответствие - "границы ПТ <-> состояния пространств".

Найти: все прочие X_{ij}.

Идем по пути 1, т.е. без счета прототипов, но удаляем неподходящие прототипы из числа известных, т.е. выполняем счет прототипов. Если мы сводим модель к одному из известных Π^{k}_{i}, то задача решена, т.к. траектория уже известна. Все новые состояния запоминаются как Π^{k}_{i} и как часть нового P^{k}.

Процедура П4:

1) Делаем текущим T_j начальное время T_1;

2) Выбираем один из X'_{ij} для данного T_j;

3) Выполняем пересечение X'_{ij} и X_{ij} по множеству $\{ X^{m}_{ij} \}$. Если пересечение пусто - выход по неуспеху, если иначе - идем на 4;

1) Сужаем X_{ij} на дополнение к X'_{ij}, т.е. удаляем из X_{ij} неиспользуемые $\{ X^{m}_{ij} \}$, а так же формируем список удаляемых состояний пространств Π^{k}_{i} (траекторий-прототипов P^{k});

4) Удаляем для данного T_j из всех прочих $\{ X^{m}_{ij} \}^{Mij}_{m=1}$ удаленные ранее состояния (прототипы), пустые X^{m}_{ij} - не

удаляются; затем последовательно проделываем то же самое для всех X_{ij}^{m} из T_{j+1}, T_{j+2} и т.д. до T_N;

5) Переходим на 7 по просмотру всех $\{X_{ij}\}$ в T_j, иначе идем на 2;

6) Выбираем множество продукций - связей $L_{ij} = \{l_{ij}^{k}\}_{k=1}^{Kij}$ для данного X_{ij} со всеми $\{X_{ij}^{m}\}_{m=1}^{Mij}$ в данном T_j, а так же множества связей со всеми $\{X_{ij}^{m}\}_{m=1}^{Mij}$ в пространствах T_{j+1}, T_{j+2} и т.д. до T_N;

7) Выполняем потоковый алгоритм Нариньяни [10] на множестве полученных связей, при том, что множество отношений интерпретации задано множеством границ; т.е.:

- выбираем одну из связей l_{ij}^{k} и отрабатываем ее, т.е. сужаем множество границ смежной по связи ПТ $X_{i'j} = \{M_{i'j}^{k}\}_{k'=1}^{K'ij}$ в данном T_j;
- далее переопределяем по связям $X_{i'j}$ множество возможных значений свойств у связанных с ней ПТ и т.д. во всех пространствах вплоть до окончания списка инициированных связей.

Потоковый алгоритм Нариньяни предполагает конечное число шагов [31];

8) Если область определения любого свойства любой ПТ оказалась пуста, то выход по неуспеху, иначе - идти на 10;

9) Если обнаружен конец списка $\{X_{ij}^{'}\}$ - то идти на 11, иначе - возврат на 2;

10) При $T_j = T_N$ - переход на 12; иначе - переходим к X_{ij}^{m}, часть X_{ij} которого уже определена на предшествующих шагах и далее переходим на 2;

11) Если хотя бы один X_{ij}^{m} из оставшихся не удаленными оказался пуст - то вводим новую идентификацию для данного нового прототипа и новых состояний в каждом T_j и запоминанием их; далее конец алгоритма по успеху.

8. Метод оценки сложности выполнения теоретико-множественных алгоритмов над грамматиками различных типов

Опишем метод оценки базы качества базы знаний, построенной на основе предлагаемого метода представления знаний

Все известные меры, которые возможно применить для оценки качества систем с базами знаний могут быть поделены на две группы:

1) Меры оценки механизмов логического вывода и структуры БЗ включают фактуальные оценки и технологические меры; например, фактуальные оценки включают: сложность БЗ, информативность БЗ, надежность вывода решения, достоверность выведенного решения, устойчивость БЗ, быстродействие системы представления и обработки знаний и т.д.

2) Меры оценки критериальных свойств для разработчиков - когнитологов и экспертов: релевантность знаний (их подтвержденность и достаточность), полнота умений (необходимость и достаточность процедур), проверенность знаний (их тестированность и целостность), уровень интеллектуальности (обучаемость, гибкость стратегий рассуждения и интерфейса), наличие мета-знаний и т.д.

Следует отметить громоздкость рассмотренной системы оценки качества БД. Мы будем рассматривать только главные выходные показатели эффективности вывода в базе знаний, задающие эффективность D-алгоритма синтеза логической области дедуктивной выводимости (ЛОДВ), соответствующего по структуре нашему алгоритму, а именно:

- L(S) – длину логического описания ОДВ (число конъюнктов в ДНФ ЛОДВ);
- M(S) – число замещений литералов в процессе работы алгоритма.

При этом, L(S) задает пространственную сложностью, M(S) – временную сложностью алгоритма. Временная и пространственная сложность вывода есть функции, зависящие от аргументов, являющимися структурными показателями сложности базы знаний:

- H(S) – высота И-ИЛИ-дерева, т.е. наибольший ярус дерева (ярус целевой вершины полагается нулевым).
- N(S) – число продукций в ЛСП S;
- q(S) – число нетерминальных литералов в ЛСП S, увеличенное на 1;
- x(s) – число терминальных литералов ЛСП S;
- $k_{\vee}$ - наибольшая степень ИЛИ-вершин И-ИЛИ-дерева, соответствующего ЛСП S;
- $k_{\&}$ - наибольшая степень И-вершин И-

ИЛИ-дерева, соответствующего ЛСП S.

Длина ЛОДВ, построенного D-алгоритмом (ленточная сложность), не превысит

$$L_D(S) \le k_{\vee}^{k_{\&}} * k_{\vee}^{(q(S)-H(S)+1)*(H(S)-2)}. \quad (8)$$

Общее число построенных в процессе работы D-алгоритма конъюнктов не превысит (временная сложность):

$$M_D(S) \le k_{\vee} + k_{\vee}^{k_{\&}} * k_{\vee}^{\frac{k_{\vee}^{(q(S)-H(S)+1)*(H(S)-1)}-1}{k_{\vee}-1}}. \quad (9)$$

9. Методы преобразования грамматического описания объекта, имеющего не допустимую когнитивную сложность представления, к виду, имеющую допустимую форму представления когнитивной сложности

9.1.Краткая характеристика метода построения меры когнитивной сложности (КС)

Работа является последней в серии статей, посвященных когнитивной сложности моделей. Ранее были разработаны: 1) методика тестирования пользователей САПР для определения шкалы абсолютной КС моделей и границ допустимой КС представления моделей [24-26; 2) метод построения формальной меры для оценки когнитивной сложности моделей. Метод построения формальной меры для оценки когнитивной сложности моделей кратко может быть охарактеризован так. Упорядочивания по сложности совокупности отношений в пределах отдельно взятого формального описания прототипа выполняется с помощью модифицированного метода диаграмм Хассе [35].

Кратко изложим суть метода. *Пусть дана* некоторая замкнутая ограниченная модель внутренней среды прототипа $\boldsymbol{P}$, представленная множеством экземпляров отношений различных уровней общности. Данные отношения представлены таблично и могут быть упорядочены по взаимному включению в соответствии со шкалой $\boldsymbol{Ш_1}$. Вес несравнимых слов, задан системой неизвестных коэффициентов:

$$\boldsymbol{M} = \{\boldsymbol{M_i^m}\}_{\boldsymbol{i=\overline{1,I\,m}}}^{\boldsymbol{m=\overline{1,W}}}. \quad (10)$$

Здесь: $\boldsymbol{W}$ - число уровней типов отношений, имеющих место в прототипе; $\boldsymbol{m}$ - номер уровня иерархии отношений, $\boldsymbol{i}$ – номер отношения в пределах уровня; $\boldsymbol{Im}$ - число отношений на уровне. Общее число отношений в диаграмме будем обозначать как $\boldsymbol{I}$.

Тогда структурная (системная) сложность $\boldsymbol{S_l}$ для объекта-прототипа $\boldsymbol{P_l}$,$\boldsymbol{l=\overline{1,Np}}$ может быть вычислена по формуле:

$$\boldsymbol{S_l = S_1^W} \text{ для } \forall \boldsymbol{l : l = \overline{1,Np}}. \quad (11)$$

Здесь: $\boldsymbol{m = W}$ - номер верхнего уровня иерархии отношений, $\boldsymbol{i = 1}$ – номер единственного отношения верхнего уровня; $\boldsymbol{S_1^W}$ - сложность прототипа, представленного как замкнутая среда.

Сложность прототипа определяется по рекуррентному соотношению:

$$\boldsymbol{S_i^m = M_i^m * \sum_{j=1}^{K_i^{m-1}} S_j^{m-1}},\ \boldsymbol{i = \overline{1, I_i^m}}. \quad (12)$$

Тут: $\boldsymbol{i}$ - номер узла $\boldsymbol{m}$-го уровня; $\boldsymbol{j}$ - номер составляющего отношения в узле; $\boldsymbol{K_i^{m-1}}$ - число отношений в узле. Всякий узел содержит структуру ***Nil***. При этом для базового 1-го уровня (значений свойств) выполняется $\boldsymbol{S_j^1 = 1; \forall j}$.

Пример состава отношений приведен на рис. 10.

Построение меры КС проводится по следующей методике. *Пусть дано:*

1) $A=\{A_l\}_{l=\overline{1,Np}}$ - совокупность оценок абсолютной КС по шкале $Ш_a$ для ряда прототипов $P_o=\{P_l\}_{l=\overline{1,Np}}$, полученные путем тестирования пользователей; данные оценки должны отражать сложность восприятия, проектирования и контроля моделирования моделей объектов;

2) Мера структурной сложности с неизвестными весовыми коэффициентами M_i^m.

Расчет коэффициентов меры КС производится по набору весов КС ряда тестовых примеров. Конкретный метод решения задачи построения абсолютной КС для произвольных прототипов зависит от числа примеров в наборе тестов.

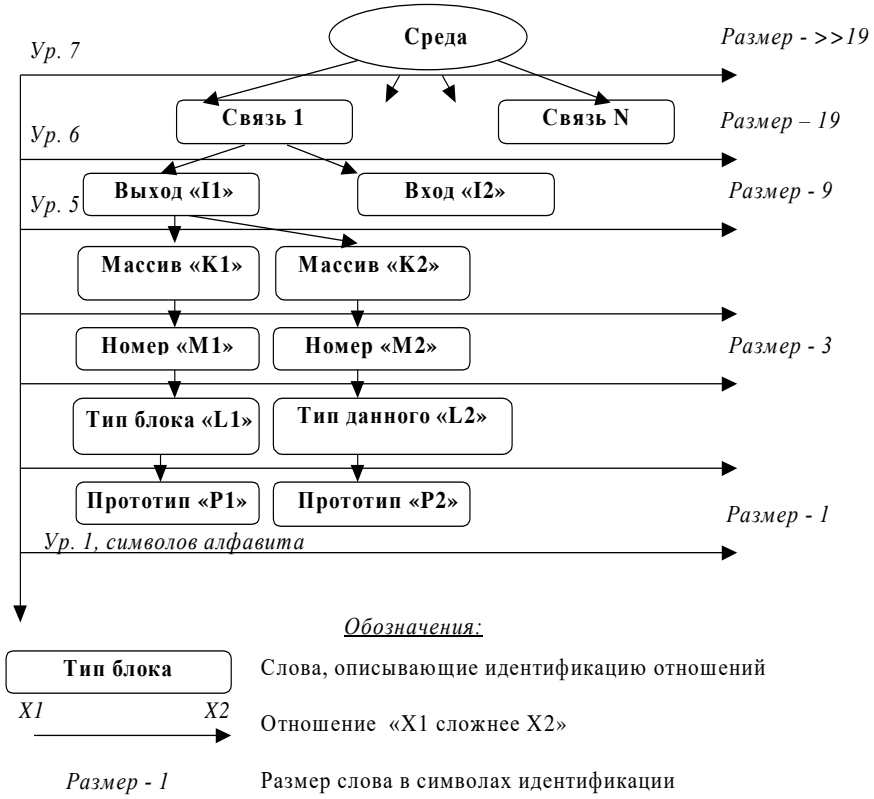

Рисунок 10 - Пример построения меры сложности на упрощенном описании внутренней среды блока-прототипа.

9.2. Общая постановка задачи ограничения КС моделей

Пусть дано:

1) Некоторая замкнутая ограниченная модель внутренней среды прототипа P, представленная множеством экземпляров отношений различных уровней общности. В данное множество отношений входит, в частности:

- Свойство как совокупность значений $Д_l=\{\partial_{li}\}_{i=1}^{N\partial}$, где: ∂_{li} - идентификаторы значений свойства $Д_l$;
- Подблоки данного блока-прототипа $Б_t \in Б=\{Б_t\}_t$, где: $Б_{jt}$ - идентификатор подблока $Б_t$, входящего в среду, t - номер в составе множества блоков $Б$, входящих в среду; $Б_1$ - внутренняя граница блока прототипа;
- Принадлежность свойства $Д_l$ границе $Б_t^e$ блока $Б_t$, заданное как отношение: $V_{tl}=(Б_t^e, Д_l)$;
- Граница блока внешняя (внутренняя) как множество отношений принадлежности свойств подблоку $Б_t$ среды S: $Б_t^e=\{V_{tl}^e\}_l$ ($Б_t^y=\{V_{tl}^y\}_l$);
- Связь s_i, т.е. причинно-

следственное отношение или отношение эквивалентности значений, определенное над парой отношений принадлежности свойств границам блоков: $s_i = (V_{ik}, V_{ip})$;

- Среда ***S***, т.е. множество причинно-следственная связей, определенных над множеством подблоков, входящих в среду: $S = \{s_i\}_{i=\overline{I,N}}$.

Примечание. Все перечисленные отношения принадлежат иерархии:

- Уровней их агрегации, что задается совокупностью имен в цепочке идентификации типа блоков и типов свойств;
- Уровней их принадлежности к вышележащим структурным единицам, что задается совокупностью имен в цепочке идентификации вхождений.

2) Мера абсолютной КСП, включающая:

- Собственно меру АКСП, определенную над множеством отношений;
- Интервал допустимой КСП $Э_5 \le KCp \le Э_2$, где $Э_5$ и $Э_2$ - точная нижняя и точная верхняя грань предельно допустимой КСП для данной предметной области, выбранного языка описания и конкретного пользователя (группы пользователей);
- ***N*** - возможная степень превышения КСП ***U***, обеспечивающая безошибочную работу оператора.

Получить: новую форму представления структуры модели ***P***, входящей в интервал допустимой КСП.

Процесс упрощения модели, заданной в такой форме представления, предполагает два последовательно выполняемых действия:

1) Упрощение модели за счет выявления в ее составе фрагментов, составляющих структуру известных решений, хранящихся в базе данных системы и изначально имеющих допустимую КС, с последующей заменой фрагмента соответствующим идентифицированным решением.

2) В случае, если после «действия 1» для оставшихся фрагментов модели нет известного решения, следует выполнить упрощение прототипа автоматически или с участием пользователя, ориентируясь на доступный уровень КС моделей.

Целью нашего изложения будет определения алгоритма выполнения второго из двух вышеописанных действий.

9.3. Алгоритм решения задачи

Предлагается следующий алгоритм обеспечения допустимой КС представления моделей. Суть алгоритма состоит в изменении структуры модели путем перемещения части подблоков прототипа во вновь создаваемые подблоки с целью снижения общей КС модели до уровня, не превышающего заданной предельной верхней границы КС. Для любого «не присоединенного» подблока будем определять дополнительно такие свойства:

- «ценность», исходя из критерия близости, определяющего меру КС его общей части с ранее выбранным набором подблоков;
- «вес», исходя из КС части подблока, отличной от ранее выбранного набора подблоков.

Критерий близости блоков есть функция, производная от числа эквивалентных отношений различных уровней иерархии. Структура критерия определяется составом и весом отношений, определенных на этапе построения меры КС. В частности, состав эквивалентных отношений может характеризоваться количеством:: K_s - связей по свойствам; K_t - имен в цепочке идентификации типа блока; K_d - наименования и типов свойств; K_z - значений свойств, заданных в порядке взаимного включения типов свойств (т.е.: время, пространство, все прочие свойства).

В этом случае критерий близости приобретет вид:

$$B = \sum_{i=1}^{N} K_i * W_i \qquad (13)$$

Тут: W_i - известные веса когнитивной сложности различных типов отношений. Исходя из данного критерия, может формироваться множество оценок близости для блоков, имеющихся в среде. Блок, имеющий наибольшую по величине меру близости с «пополняемым» блоком, объединяется с ним в единый логический блок.

Т.о. решается оптимизационная задача, близкая по своей постановке к классической «задаче о рюкзаке». Отличие предлагаемой постановки данной задачи заключается в динамически изменяемых ценности и весов предметов, складываемых в рюкзак.

Описание алгоритма.

1) Проверка исходной схемы ***Р*** на предельно допустимую КС. Если условие $\boldsymbol{Э_5 \le КСр \le Э_2}$ не выполняется, то переходим на пункт 2, иначе – на конец алгоритма, т.е. пункт 12.

2) Формирование пустого списка ***S***п, предназначенного для внесения в него извлекаемых из тела прототипа ***Р*** подблоков.

3) Формирование на базе ***S***п нового «пустого» блока-аккумулятора ***П***, имеющего пустой список свойств, составляющих внешнюю и внутреннюю его границу, и внесение его в ***Б***.

4) Формирование списка запрещенных блоков ***S***z и внесение в него внутренней границы прототипа блока ***Б1*** и блока-аккумулятора ***П***;

5) Выбор из множества ***Б*** набора блоков ***{Бs},*** такого, что верно

$$\forall \boldsymbol{Б_S} \in \boldsymbol{Б} \setminus \boldsymbol{S_Z} \tag{14}$$

и выполняется условие максимальной близости ***Бs*** со списком ***Sn*** по критерию

$$\boldsymbol{K(Б_S \overline{\cap} S_П)} = \max_{\boldsymbol{Б_I \in Б}} \boldsymbol{K(Б_I \overline{\cap} S_П)} \tag{15}$$

где: $\overline{\cap}$ - операция определения общего подмножества отношений.

Если имеет место множество ***{Бs},*** для которых выполняется (6), например на первом шаге алгоритма, когда список ***П*** пуст, то выполняется переход на пункт 6, иначе на пункт 8.

6) Выбор из множества ***{Бs},*** для которых выполняется (2), блока ***Бs'***, для которого степень уменьшения КС блока ***Б***, получаемая за счет удаления блока ***Бs'*** максимальная среди всех ***Бi***, принадлежащих ***Б***, т.е.

$$\boldsymbol{K(Б \setminus Б'_S)} = \max_{\boldsymbol{Б_I \in Б}} \boldsymbol{K(Б \setminus Б_I)} \tag{16}$$

где: \ - операция удаления общего подмножества отношений.

При этом:

- степень увеличения КС блока ***П*** за счет блока ***Бs'*** максимальная среди всех ***Бi***, принадлежащих ***Б***, (см. формулу 11);
- критерий близости ***Бs*** с блоком ***П*** имеет максимальный вес (см. формулу 12);
- критерий близости ***Бs*** со списком ***Б*** имеет максимальный вес (см. формулу 13).

$$\boldsymbol{K(Б_S \cup П)} = \max_{\boldsymbol{Б_I \in Б}} \boldsymbol{K(Б_I \cup П)}; \tag{17}$$

$$\boldsymbol{K(Б_S \cap П)} = \max_{\boldsymbol{Б_I \in Б}} \boldsymbol{K(Б_I \cap П)}; \tag{18}$$

$$\boldsymbol{K(Б_S \cap (Б \setminus Б_S))} = \max_{\boldsymbol{Б_I \in Б}} \boldsymbol{K(Б_I \cap (Б \setminus Б_I))}. \tag{19}$$

6) Внесение блока ***Бs*** в состав списка ***Sn***.

7) Формирование на базе ***Sn*** нового блока-аккумулятора ***П***, имеющего список свойств, составляющих внешнюю и внутреннюю его границу, и внесение его в ***Б***; при этом:

- формируется связи данного блока ***Бs*** с подблоками блока ***П***;
- внутренняя и внешняя граница блока ***П*** пополняется свойствами, посредством которых блок ***Бs*** связан с внешней средой;

8) Изменение состава блоков и связей исходного прототипа ***Р*** с учетом удаления из его состава найденного блока ***Бs*** и включения связей с новым блоком ***П***;

9) Если ***Sz=P*** то новый блок ***П*** есть искомый и далее переход на конец алгоритма, т.е. пункт 12;

10) Оценка на КС нового блока ***П***: если условие $\boldsymbol{Э_5}$<=КСр<=$\boldsymbol{Э_2}$ выполняется, то идти на выбор следующего вторичного блока, т.е. на пункт 5, иначе на пункт 11;

11) Т.к. выбранный нами новый подблок для ***П*** превысил необходимый уровень ($\boldsymbol{Э_5}$<КСр), то делается:

- возврат на предыдущий вариант ***Sn*** и соответствующему ему виду ***Р*** и ***П***;
- внесение блока ***Бs*** в список

запрещенных блоков ***Sz***;

- переход на выбор следующего вторичного блока, т.е. на пункт 5;

12) Формирование нового «пустого» блока ***П***, т.е. имеющего пустой список подблоков и свойств, составляющих внешнюю и внутреннюю его границу, и внесение его в ***Б***.

10. Изобретение новых решений – как инструмент сужения числа аксиом грамматики, построенной путем обобщения прототипов

Общий подход к синтезу гипотез предполагает, что при склеивании прототипов автоматически формируется декартово произведению всех составляющих всех ИЛИ-синтермов, входящих в И-ИЛИ-дерево.

Данное И-ИЛИ-дерево с определенными над ним продукционными зависимостями, составляет динамическую базу данных. Результатом ограничений на каждом этапе является все более суженное "полное" И-ИЛИ-дерево.

Увеличение числа вводимых прототипов ведет к возникновению все большего числа частей блоков и фрагментов. Можно сделать выводы, что:

1. Количество вариантов сред чрезвычайно большое и пользователь не в состоянии их проверить их на достоверность;

2. Нужны ограничения технологического и семантического характера, способные резко снизить число потенциально возникающих прототипов при присоединении нового прототипа, отсекая решения, не имеющие смысл и решения, проверить адекватность которых для данного пользователя не представляется возможным.

В связи с этим, для решения данной задачи предлагается технология отсечения слишком сложных и не имеющих смысл решений, схема которой показана на рис. 11.

11. Построение системы интерфейсов «Язык предметной области ↔ Язык формальных спецификаций соответствующего уровня абстракции»

Цель построения системы интерфейсов «Язык предметной области ↔ Язык формальных спецификаций соответствующего уровня абстракции»:

- обучения базы знаний методикам проектирования по прототипам, накопленным в проблемно-ориентированном САПР;

- передачи готового решения из инструментальной оболочки в проблемно-ориентированный САПР.

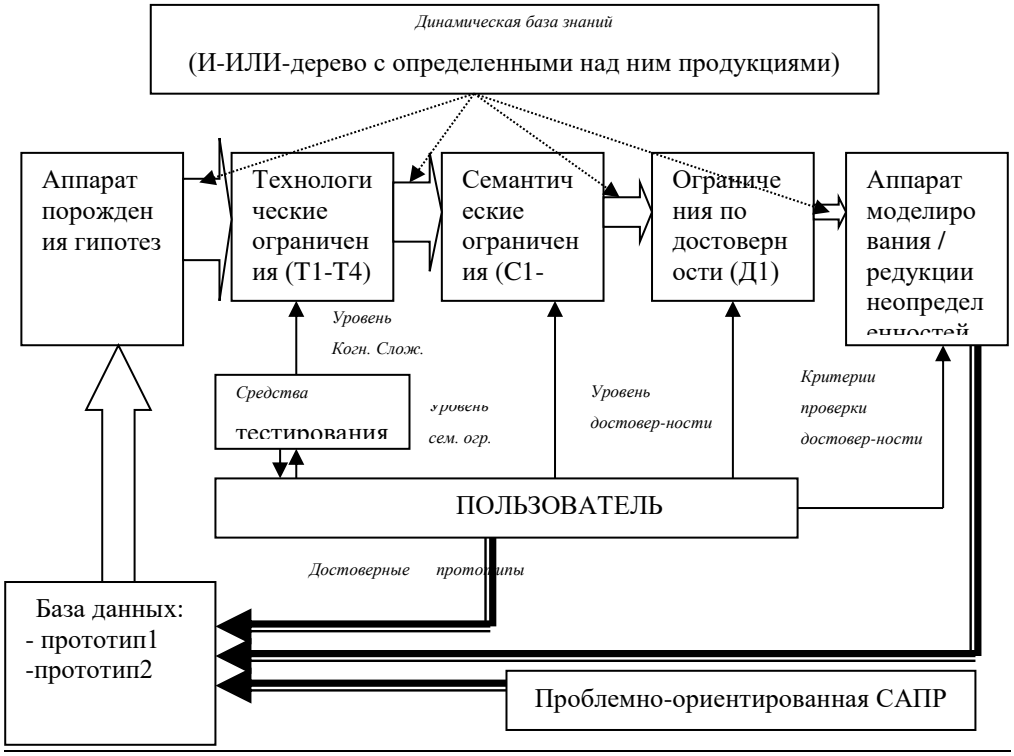

Рисунок 11 - Схема технологии синтеза и отбора гипотез в МЭО

Модель предлагаемого интерфейса может быть представлена следующим образом:

M=(Go,Fo,Mo,So,Gs,Fs,Ms,Ss,Pos,Pso),

где:

Go - грамматики языков представления моделей в ИО;

Fo - формат внутренних структур данных представления моделей в ИО;

Mo - описания прототипов в данной проблемной области на языках ИО (знания экспертов, проекты);

So - описания прототипов в данной проблемной области в формате внутренних структур данных МЭО;

Gs - грамматики языков представления моделей в других инструментальных средствах проектирования сложных систем (САПР);

Fs - внутренние структуры данных представления моделей других САПР в данной проблемной области;

Ms - описания прототипов в данной проблемной области на языках прочих САПР (знания экспертов, проекты);

Ss - описания прототипов в данной проблемной области в формате внутренних структур данных прочих САПР;

Pos - процедуры отображения Mo,So в Ms,Ss;

Pso - процедуры отображения Ms,Ss в Mo,So.

Охарактеризуем формы представления моделей в ИО:

1. Грамматика языка представления моделей в ИО (Go) описана в [7].

2. Модель структур во внутреннем формате (So), представляется в виде описаний библиотек, типов, массивов и т.п. в формате DBF.

3. Функциональные модели, задающие соответствия для базовых структурных блоков и связей, представляются в виде динамических недоопределенных вычислительных моделей, описание которых совмещено с описанием структур (So) и представлено в формате DBF.

Заключение

В работе описаны структура средств и методов работы с формальными грамматиками в рамках С КМ ПрО МЭО. Построение данной КМ выполнено автором в рамках работ [1-50]. Данная КМ является основой построения соответствующего инструментального комплекса по автоматизации построения интеллектуальных САПР некоторого ограниченного класса – мета-эвристической оболочки (МЭО).

Наибольшее влияние на методы построения КМ оказала выбранная форма ее представления – семиотическая модель. А так как главный компонент СМ – это формальные грамматики, то без преувеличения можно сказать, что суть КМ – это набор методов работы с формальными грамматиками, определенными в рамках СМ.

Перспективой работы является дальнейшая работа над расширением предложного комплекса средств и методов.

Литература

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

2. Григорьев А.В. И/ИЛИ-дерево как средство абстрактного представления базы знаний // Наукові праці Донецького національного технічного університету. Серія: Інформатика, кібернетика та обчислювальна техніка, випуск 39: – Донецьк, ДонНТУ, 2002. - С. 36–42.

3. Григорьев А.В. Комплекс моделей САПР как система взаимосвязанных уровней о действительности. // Научные труды Донецкого государственного технического университета. Серия "Информатика, кибернетика и вычислительная техника", (ИКВТ-2000) выпуск 10. - Донецк, ДонГТУ, 2000. - С. 155-167.

4. 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.

5. Представление недоопределенности знаний в инструментальной оболочке для построения САПР. // Искусственный интеллект. N 6, 1999 , С. 56-66.

6. Григорьев А.В. Адаптивная система ограничений на сложность при синтезе новых решений в интеллектуальных САПР // Искусственный интеллект. – Донецк, 2001 – N 2 – С. 152 – 167.

7. Григорьев А.В. Унифицированная концептуальная модель предметной области. // Информатика, кибернетика и вычислительная техника (ИКВТ-97). Сборник трудов ДонГТУ, Выпуск 1. Донецк: ДонГТУ, 1997. С.218-224.

8. Григорьев А.В. Концептуальная модель оболочки для построения интеллектуальных САПР вычислительной техники как средство предметной адаптации.В кн. Наукові праці Донецького національного технічного університету. Серія „Інформатика,

кібернетика та обчислювальна техніка" (ІКОТ-2009). Випуск 10(153).- Донецьк: ДВНЗ «ДонНТУ». - 2009. – С.255-265.

9. Григорьев А.В. Вербальная модель предметной области для интеллектуальных САПР // Наукові праці Донецького Державного технічного університету. Серія "Обчислювальна техніка та автоматизація". Випуск 20. - Донецьк, ДонДТУ, 2000. - С. 171-180.

10. Григорьев А.В. Содержание некоторых категорий абстракций в теории построения интеллектуальных САПР. // Наукові праці Донецького національного технічного університету / Редкол.: Башков Є.О. та інші. Серія: "Обчислювальна техніка та автоматизація": Випуск 64.: Донецьк: Видавництво ДонНТУ, 2003 - С. 166-178.

11. Григорьев А.В. Семантика модели предметной области для интеллектуальных САПР. // Научные труды Донецкого государственного технического университета. Серия "Информатика, кибернетика и вычислительная техника", (ИКВТ-2000) выпуск 10. - Донецк, ДонГТУ, 2000. - С. 148-154.

12. Григорьев А.В. Специфика выполнения теоретико-множественных операций над контекстно-свободными грамматиками в условиях различных форм дополнительных семантических правил в семиотической модели интеллектуальных САПР Научные труды Донецкого национального технического университета. Серия «Проблемы моделирования и автоматизации проектирования динамических систем» (МАП – 2006). Выпуск 5 (116). – Донецк: ДонНТУ, 2006. – С. 91-104.

13. Григорьев А.В. Алгоритм выполнения теоретико-множественных операций над грамматиками в среде специализированной оболочки для создания интеллектуальных САПР // Наукові праці національного технічного університету. Серия «Проблемы моделирования и автоматизации проектирования динамических систем» (МАП - 2002). Выпуск 52: Донецк: ДонНТУ, 2002. - С.83-93.

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

15. Григорьев А.В. Методы построения функций в специализированной оболочке для создания интеллектуальных САПР // Искусственный интеллект. – Донецк, 2001 – №3 – С. 40–53.

16. Григорьев А.В. Обобщение знаний в интеллектуальной системе с семиотической моделью представления знаний // Научные труды Донецкого государственного технического университета. Серия: Проблемы моделирования и автоматизации проектирования динамических систем, выпуск 29. – Севастополь: «Вебер», 2001. – С. 114–120.

17. Григорьев А.В. Управление движением объектов в семиотической модели предметной области // Наукові праці національного технічного університету. Серия «Обчислювальна техніка та автоматизація». Выпуск 48: Донецк: ДонНТУ, 2002. - С.280-287

18. Григорьев А.В. Организация вывода решений в базе знаний инструментальной оболочки для создания интеллектуальных САПР / Наукові праці Донецького національного технічного університету. Серія "Проблеми моделювання та автоматизації проектування динамічних систем" (МАП-2005). Випуск: 78 - Донецьк: ДонНТУ. – 2005 – с. 171-182.

19. Григорьев А.В. Организация пространственного и временного логического вывода в концептуальной модели интеллектуальных САПР. В кн. Наукові праці Донецького національного технічного університету. Серія „Інформатика, кібернетика та обчислювальна техніка" (ІКОТ-2008). Випуск 9(132).- Донецьк: ДонНТУ. - 2008. – С.296-311.

20. Григорьев А.В. Обеспечение монотонности вывода и верификация баз знаний в инструментальной оболочке для создания интеллектуальных надстроек над САПР. // В кн. Наукові праці Донецького національного технічного університету. Серія „Інформатика, кібернетика та обчислювальна техніка" (ІКОТ-2010). Випуск 11(164).- Донецьк: ДВНЗ «ДонНТУ». - 2010. – С.161-164.

21. Григорьев А.В. Методы решения задачи структурного синтеза в интеллектуальных САПР, построенных на основе семиотической модели структур.В кн. Наукові праці Донецького національного технічного університету. Серія „Обчислювальна техніка та автоматизація". Випуск 171(19) / Редкол.: Башков Є.О. (голова) та ін. - Донецьк: ДонНТУ, 2010. – С.128-140.

22. Григорьев А.В. Система автоматизированного решения вычислительных задач в САПР, основанная на методе программирования в ограничениях. В кн. Наукові праці національного технічного університету. Серія «Проблеми моделювання та автоматизації проектування динамічних систем. (МАП-2011). Випуск 9 (129): Донецьк: ДонНТУ. - 2011. – С. 43-51.

23. Григорьев А.В. Логические системы представления знаний и вывода в искусственном интеллекте. // Научные труды Донецкого государственного технического университета. Серия "Информатика,

кибернетика и вычислительная техника", (ИКВТ-2000) выпуск 10. - Донецк, ДонГТУ, 2000. - С. 225-230.

24. Григорьев А.В. Ограничение когнитивной сложности моделей. // Прогрессивные технологии и системы машиностроения:: Международный сб. научных трудов. - Донецк: ДонГТУ, 2000. Выпуск. 10 - С. 49-58.

25. Григорьев А.В. Методика тестирования для определения когнитивной сложности моделей различных предметных областей. // Научные труды Донецкого государственного технического университета. Серия: Информатика, кибернетика и вычислительная техника, выпуск 6 (ИКВТ-99): - Донецк: ДонГТУ, 1999. - С. 246-251.

26. Григорьев А.В. Оценка когнитивной сложности моделей. // Научные труды Донецкого государственного технического университета. Серия: Информатика, кибернетика и вычислительная техника, выпуск 6 (ИКВТ-99): - Донецк: ДонГТУ, 1999. - С. 252-259.

27. Григорьев А.В. Изобретение как метод автоматизации процесса обучения методикам проектирования в семиотической модели САПР. // Информатика и кибернетика. – Д.: ДонНТУ, – 2015. – № 1. - С. 51- 66.

28. Григорьев А.В. Анализ эффективности и перспектив развития методов построения двусторонних трансляторов в задаче создания интеллектуальных надстроек над проблемно-ориентированными САПР.В кн. Наукові праці національного технічного університету. Серія «Обчислювальна техніка та автоматизація». Випуск 20 (182). –: Донецьк: ДонНТУ, 2011. – С. 116-129.

29. Григорьев А.В. Анализ существующих способов создания интерфейса «языки формальных спецификаций — проблемно-ориентированные языки». В кн. Серія „Інформатика, кібернетика та обчислювальна техніка” (ІКОТ-2011). Випуск 14(185).- Донецьк: ДВНЗ «ДонНТУ», 2011.270-275.

30. Григорьев А.В. Интерфейс табличного процессора EXCEL и специализированной оболочки для синтеза интеллектуальных САПР и АСНИ. // Информатика, кибернетика и вычислительная техника (ИКВТ-97). Сборник трудов ДонГТУ, Выпуск 1. Донецк: ДонГТУ, 1997. С. 229-238.

31. Нариньяни А.С. Недоопределенность в системах представления и обработки знаний. // Известия АН СРСР. Техническая кибернетика. - 1986.- № 5. – С. 3-28.

32. Д.В. Банасюкевич, И.Д. Гофман, Д.А. Инищев, А.С. Нариньяни. Интеллектуальная система планирования и управления проектами на базе недоопределенной математики. В кн.: “КИИ '2000 седьмая национальная конференция по искусственному интеллекту з международным участием”: Том 1. Москва, 2000, С.. 617 — 624.

33. Нариньяни А.С. Не-факторы: неточность и недоопределенность – расхождение и взаимосвязь. Известия академии наук. Теория и системы управления, 2000, №5, С. 44-56.

34. Норенков И.П. Разработка систем автоматизации проектирования. М.: МГТУ им. Э.Н.Баумана, 1994. – 207 с.

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

36. Поспелов Д.А. Ситуационное управление: теория и практика. М.: Наука, 1986. - 288 с.

Григорьев А.В. Комплекс средств и методов работы с формальными грамматиками в семиотической концептуальной модели предметной области интеллектуальных САПР. *Рассмотрен комплекс методов работы с формальными грамматиками, определенных в рамках концептуальной модели предметной области инструментальной оболочки для автоматизации построения интеллектуальных САПР ограниченного класса. Работа носит итоговый характер.*

Ключевые слова: *формальные грамматики, концептуальная модель предметной области, семиотическая модель, САПР.*

Grigoriev A.V. The complex of means and methods of working with a formal grammar in semiotic conceptual domain model of intelligent CAD. *The complex of methods of work with formal grammars defined within the conceptual model of the instrumental domain shell for building intelligent automation CAD limited class. The work is the final character.*

Keywords: *formal grammar, conceptual domain model, the semiotic model of CAD.*

Статья поступила в редакцию 14.2.2017
Рекомендована к публикации д-ром физ.-мат.. наук А.С. Миненко