Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Восточно-Сибирский государственный университет технологий и управления»

На правах рукописи

Кравченко Вячеслав Александрович

ЛОГИКО-МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ ДИНАМИЧЕСКИХ СИСТЕМ С ИСПОЛЬЗОВАНИЕМ АППАРАТА ФУНКЦИОНАЛЬНЫХ ГРАММАТИК

05.13.18 – «Математическое моделирование, численные методы
и комплексы программ»

Диссертация на соискание
ученой степени кандидата технических наук

Научный руководитель:
доктор физико-математических
наук, профессор Ширапов
Дашадондок Шагдарович

Улан-Удэ
2017

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность логико-математического моделирования заключается в важности его прикладного значения, которое состоит в возможности трансформации логико-математической модели в предметно-математическую модель в виде программы для ЭВМ, осуществляющей компьютерное моделирование рассматриваемой системы. Совершенствование методов логико-математического моделирования динамических систем способствует развитию автоматизированного построения алгоритмов моделирования.

Первоначальные идеи логико-математического моделирования, предназначенного для автоматизированного синтеза программ были сформулированы в начале 70-х гг. XX века Э.Х. Тыугу в Институте кибернетики АН Эстонской ССР (г. Таллин) и Г.Е. Минцем в Ленинградском отделении математического института АН СССР [88]. В частности, был выдвинут тезис о том, что автоматический синтез программ по спецификациям задач должен быть основан на конструктивном доказательстве теоремы существования решения задачи. Таким образом, была поставлена задача разработки методов логико-математического моделирования.

С конца 70-х гг. до середины 80-х гг. методы логико-математического моделирования активно разрабатывались С.С. Лавровым в Институте теоретической астрономии АН СССР. Результатом является система СПОРА, построенная на исчислении высказываний [57].

На основе идей С.С. Лаврова в 80-е гг. было сделано несколько десятков реализаций систем логико-математического моделирования в разных организациях, как в форме систем автоматического синтеза программ общего назначения, так и в форме конкретных предметно-ориентированных пакетов прикладных программ. Но наибольшую известность получили работы Э.Х. Тыугу [31, 97], в которых логико-математическое моделирование является частью «концептуального программирования». Результатом работы научной группы Э.Х.

Тыугу является мощная система автоматизированного синтеза программ решения инженерных задач ПРИЗ.

В 90-х гг. имело место некоторое забвение данной темы в связи с развитием объектно-ориентированного программирования. Начиная с 2000-х гг. вновь появляется интерес к концептуальному программированию. Современные работы в этом направление ведутся в Московском институте электроники и математики (ВШЭ) [60, 79], в Ядерном университете МИФИ [15, 17, 28], в Московском государственном университете [33-37], в Институте проблем информатики РАН [24-27], в Институте системного программирования РАН [20, 55], в Томском политехническом университете [70, 71, 73], в Иркутском государственном университете путей сообщения [2-5, 112] в Институте динамики систем и теории управления СО РАН [13, 59, 69, 72, 100-102]. В качестве примера зарубежных работ необходимо указать результаты француза К. Динечина [113].

Подавляющее большинство перечисленных работ опирается на подходы, заложенные Э.Х. Тыугу, основным из которых является представление знаний семантическими сетями, которые описываются или могут быть описаны атрибутными контекстно-свободными грамматиками.

Атрибутные грамматики были введены Д.Э. Кнутом в далёком 1968 году [114]. С тех пор они были усовершенствованы и используются в системах программирования и инженерии знаний [99]. На основе атрибутных грамматик в 1980 году В.А. Тузовым из Ленинградского государственного университета были введены функциональные контекстно-свободные грамматики [93]. В монографии [94] показаны преимущества функциональных грамматик над атрибутными при построении программных модулей. Также В.А. Тузовым была указана возможность использования функциональных грамматик в представлении знаний [95]. Но дальнейшие его работы и работы его учеников лежат в смежной области – они направлены на использование функциональных грамматик в построении математической модели естественного русского языка [63, 64, 96].

В данной работе расширяется использование функциональных грамматик на область логико-математического моделирования динамических систем.

Целью диссертационной работы является разработка единой методики и программного алгоритма логико-математического моделирования динамических систем из разных предметных областей на основе аппарата функциональных контекстно-свободных грамматик.

Для достижения поставленной цели решаются следующие **задачи**:

1. Анализ методов логико-математического моделирования динамических систем.

2. Определение контекстно-свободных функциональных грамматик.

3. Разработка способа логико-математического представления знаний выбранной предметной области, содержащей законы функционирования динамической системы, на основе аппарата функциональных контекстно-свободных грамматик.

4. Разработка логико-математической модели решения прямых и обратных задач моделирования динамических систем в терминах функциональных грамматик.

5. Разработка алгоритма программного комплекса логико-математического моделирования динамических систем на основе аппарата функциональных контекстно-свободных грамматик.

6. Апробация методов логико-математического моделирования динамических систем и алгоритма программного комплекса в теории линейных стационарных радиотехнических систем с применением численных методов.

Методы исследования. При выполнении диссертационной работы применялись методы формального описания математических моделей, методы формальных грамматик, λ-исчисление, методы представления знаний, эвристические методы поиска решений, методы концептуального программирования, численные методы, временные и спектральные методы анализа радиотехнических цепей. Для описания алгоритмов программной реализации использован функциональный язык программирования Лисп.

Научная новизна результатов, выносимых на защиту:

1. Впервые разработан метод логико-математического моделирования динамических систем на основе применения аппарата функциональных грамматик.

2. При описании законов функционирования линейных стационарных радиотехнических систем использована новая форма представления знаний – продукционная система, описанная неполной функциональной контекстно-свободной грамматикой.

3. Созданы новые алгоритмы на основе вывода суперпозиции функций для построения программного комплекса логико-математического моделирования динамических систем.

Теоретическая и практическая значимость результатов. Полученные в работе результаты могут быть использованы для логико-математического моделирования динамических систем из разных предметных областей. Разработанный алгоритм программного комплекса осуществляет автоматизированный вывод логико-математической модели и построение на её основе программы математического моделирования заданной системы. Используемые приемы логико-математического моделирования позволяют эффективно встраивать численные методы для расчета характеристик динамических систем. Разработанная база знаний по радиотехнике может быть применена для анализа и синтеза пассивных радиотехнических фильтров, или усовершенствована для работы с другими видами радиотехнических систем.

Достоверность полученных результатов подтверждается успешным созданием базы знаний по радиотехнике линейных стационарных систем и использованием её для решения прямых и обратных задач логико-математического моделирования в указанной предметной области.

Полученные в работе результаты соответствуют трем областям исследования паспорта специальности 05.13.18 – Математическое моделирование, численные методы и комплексы программ:

п.2. Развитие качественных и приближенных аналитических методов исследования математических моделей.

п.3. Разработка, обоснование и тестирование эффективных вычислительных методов с применением современных компьютерных технологий.

п.4. Реализация эффективных численных методов и алгоритмов в виде комплексов проблемно-ориентированных программ для проведения вычислительного эксперимента.

Апробация результатов. Результаты работы были представлены в виде докладов и обсуждались на научных конференциях:

1. III международная конференция «Инфокоммуникационные и вычислительные технологии и системы» (ИКВТС-2010), г. Улан-Удэ (оз. Байкал), 6-11 сентября 2010 г.

2. Конференция PAE «Функциональные и прикладные исследования. Образование, экономика и право», Италия (Рим, Флоренция), 12-19 сентября 2011 г.

3. XI Всероссийская научно-техническая конференция «Теоретические и прикладные вопросы современных информационных технологий» (ТиПВСИТ-2012), г. Улан-Удэ (оз. Байкал), 13-20 августа 2012 г.

4. Всероссийская научная конференция «Компьютерные технологии в науке, в технике, в искусстве», г. Таганрог, 15 июня 2013 г.

5. II Всероссийская научная интернет-конференция с международным участием «Современные системы искусственного интеллекта и их приложения в науке», г. Казань, 14 мая 2014 г.

6. Международная конференция «Дифференциальные уравнения и математическое моделирование» (ДУММ-2015), г. Улан-Удэ (оз. Байкал), 22-27 июня 2015 г.

7. XII Международная научно-практическая конференция «Инновации на основе информационных и коммуникационных технологий» (Инфо-2015), г. Сочи, 1-10 октября 2015 г.

8. XIII Международная научно-практическая конференция «Инновационные, информационные и коммуникационные технологии» (Инфо-2016), г. Сочи, 1-10 октября 2016 г.

9. X Международная научно-практическая конференция «Техника и технология: новые перспективы развития», г. Самара, 25 сентября 2017 г.

10. X Международная научно-практическая конференция «Современные технологии в мировом научном пространстве», г. Уфа, 28 сентября 2017 г.

11. XIV Международная научно-практическая конференция «Инновационные, информационные и коммуникационные технологии» (Инфо-2017), г. Сочи, 1-10 октября 2017 г.

12. Ежегодные научно-практические конференции преподавателей, научных сотрудников и аспирантов Восточно-Сибирского государственного университета технологий и управления в 2010–2017 гг.

Публикации. Основные результаты диссертации опубликованы в 19 работах, включая статьи в журналах и материалах конференций [7, 38-53, 108], и Свидетельство о регистрации программы для ЭВМ [83]. В том числе 5 статей в рецензируемых журналах, включенных в Перечень ВАК [38, 40, 43, 47, 49].

В статьях [7, 108], выполненных в соавторстве, диссертант является полноценным соавтором и участвовал от постановки задачи до получения результатов. В статьях [38, 41-43, 46-53], в которых диссертант является первым автором, его вклад – основной.

Структура и объем диссертации. Диссертация состоит из введения, четырех глав, заключения, списка литературы и приложений. Объем работы составляет 126 страниц машинописного текста. Количество рисунков – 23, таблиц – 11, наименований списка литературы – 114.

Глава 1 посвящена применению функциональной парадигмы в логико-математическом моделировании динамических систем.

В параграфе 1.1 определена задача логико-математического моделирования, перечислены методы построения логико-математической модели и рассмотрены этапы дедуктивного синтеза.

В параграфе 1.2 приводится формальное описание задачи логико-математического моделирования динамических систем, указаны возможные

варианты прямых и обратных задач моделирования. Приведена блок-схема логико-математического моделирования динамических систем.

В параграфе 1.3 показаны преимущества применения аппарата функциональных грамматик над применением аппарата атрибутных грамматик, поставлена цель применения функциональных грамматик в логико-математическом моделировании.

В параграфе 1.4 разработан последовательно-параллельный метод декомпозиции оператора логического вывода, необходимый для применения аппарата функциональных грамматик. Показаны преимущества последовательно-параллельного метода перед последовательным.

Глава 2 посвящена теоретическому описанию применения аппарата функциональных грамматик в логико-математическом моделировании.

В параграфе 2.1 приводятся общие элементы теории формальных грамматик. Введено понятие дерева перебора.

В параграфе 2.2 содержится определение функциональных грамматик и показано получение функциональной грамматики из контекстно-свободной. Указано взаимоотношение функциональной и атрибутной грамматик.

В параграфе 2.3 даны определения неполных контекстно-свободной и функциональной грамматик, описано представление знаний в виде неполной функциональной грамматики. Приведен пример на основе законов механики. Сформулированы этапы составления базы знаний. Получено описание метода представления знаний на языке чистого лямбда-исчисления.

Параграф 2.4 посвящен описанию представления условия задачи в виде полной функциональной грамматики и описанию вывода решения в виде суперпозиции функций на основе дерева перебора. Получены правила роста дерева, ограничивающие его разрастание и сужающие поиск. Приведен пример поиска решения задачи. Рассмотрено логико-математическое описание метода на языке чистого лямбда исчисления. Приведена общая схема логико-математического моделирования динамических систем с помощью аппарата функциональных грамматик.

Глава 3 посвящена построению базы знаний по радиотехнике линейных стационарных систем в виде неполной функциональной грамматики.

В параграфе 3.1 теория радиотехнических систем ограничена линейной стационарной пассивной радиотехнической цепью в виде *Г*-образного четырёхполюсника из *R, L* и *C* элементов, на входе и выходе которой действуют условные дискретные сигналы, подчиняющиеся теореме Котельникова.

В параграфе 3.2 выявлен набор понятий и получен алфавит символов функциональной грамматики. Для каждого символа получена контекстно-свободная грамматика, описывающая синтаксис его значения.

В параграфе 3.3 выявлены отношения между понятиями теории и построена система правил.

В параграфе 3.4 определены функциональные зависимости, входящие в состав правил функциональной грамматики.

Параграф 3.5 содержит два примера решения задач моделирования, одна из которых является прямой, а вторая – обратной.

Глава 4 посвящена разработке алгоритмов построения программного комплекса, осуществляющего логико-математическое моделирование динамических систем.

В параграфе 4.1 рассмотрены категории знаний и соответствующие им типы программных комплексов. В качестве типа программного комплекса определена прикладная система с элементами искусственного интеллекта. Приведена структурная схема программного комплекса. Объяснён выбор языка программирования Лисп.

В параграфе 4.2 приведены методы построения программного модуля ввода базы знаний и получения задачи моделирования динамических систем.

В параграфе 4.3 рассмотрены методы построения решателя (механизма вывода) задач моделирования.

В параграфе 4.4 приведены возможности использования результата логико-математического моделирования в виде суперпозиции функций. На примере

показаны преимущества суперпозиции функций над программным результатом классического метода.

В параграфе 4.5 перечислены символьные и численные методы, использовавшиеся при программировании функций базы знаний по радиотехнике линейных стационарных систем.

В заключении представлены основные научные результаты работы.

В приложении приведены листинг программного комплекса логико-математического моделирования и фрагмент базы знаний по радиотехнике линейных стационарных систем.

ГЛАВА 1. ФУНКЦИОНАЛЬНАЯ ПАРАДИГМА В ЛОГИКО-МАТЕМАТИЧЕСКОМ МОДЕЛИРОВАНИИ ДИНАМИЧЕСКИХ СИСТЕМ

1.1 Задача логико-математического моделирования

Под логико-математической моделью системы понимается совокупность логических отношений, принадлежащих определенной формальной теории и связывающих характеристики состояния рассматриваемой системы с её параметрами, исходной информацией и начальными условиями.

Системы математического моделирования, реализуемые на ЭВМ, функционируют в четыре этапа:

этап 1: формулировка задачи моделирования и разработка математической модели исследуемой системы;

этап 2: программная реализация математической модели на ЭВМ с использованием численных методов и методов символьной обработки информации;

этап 3: проведение реализации моделирования для заданных ситуаций и оформление результатов моделирования;

этап 4: интерпретация результатов моделирования для получения новой информации об объекте исследования.

Большинство современных систем математического моделирования предполагают ручную разработку модели для каждой задачи моделирования (этап 1), ручную разработку программы или пакета программ для полученной модели (этап 2), автоматизированную реализацию моделирования разных ситуаций (этап 3) и автоматизированную интерпретацию результатов моделирования (этап 4).

Целью логико-математического моделирования является автоматизация первых двух этапов. Логико-математическая модель составляется автоматически для решения задач моделирования в определенной предметной области (этап 1). Её

прикладное значение заключается в возможности трансляции в предметно-математическую модель в виде программы для ЭВМ (этап 2), осуществляющей компьютерное моделирование рассматриваемой системы (этапы 3 и 4).

Таким образом, логико-математическое моделирование является инструментом автоматизированного синтеза программ моделирования. Существуют три различных подхода логико-математического моделирования, использующихся в синтезе программ [97]:

1. Дедуктивный подход, при котором построение модели проводится на основе описания ее цели, заданной в виде спецификации (описания задачи). При таком подходе используется конструктивное доказательство утверждения о том, что решение задачи моделирования существует, и из него извлекается требуемая программа.

2. Индуктивный подход, при котором программа строится по примерам, непосредственно задающим ответ для некоторых исходных данных.

3. Трансформационный подход, где программа получается путем преобразования исходного описания задачи по правилам, совокупность которых представляет знания о решении задач. Трансформационный синтез позволяет получать из менее эффективных программ эквивалентные им, но более эффективные.

Наиболее распространенным способом логико-математического моделирования является дедуктивный синтез. Согласно нему, логико-математическое моделирование имеет несколько этапов [97]:

1) формулировка задачи моделирования в логическом виде;

2) доказательство решения задачи моделирования в виде логического вывода (проверка возможности декомпозиции задачи моделирования);

3) получение логико-математической модели из доказательства решения задачи (декомпозиция задачи моделирования на элементарные операции);

4) программная реализация логико-математической модели.

Для формулировки задачи (этап 1) необходимо выбрать способ представления знаний о предметной области, для доказательства решения и

построения модели (этапы 2 и 3) – математический аппарат логического вывода, а для программной реализации – язык программирования.

В работах Э.Х. Тыугу [31, 97] и современных работах, опирающихся на подходы концептуального программирования, для представления знаний используются семантические сети, основанные на атрибутных грамматиках; для доказательства решения и построения модели – математическая логика, а для программной реализации – специально созданный язык программирования.

Для применения параллельных вычислений и эффективного использования функциональных языков программирования в логико-математическом моделировании динамических систем автором предлагается вместо атрибутных грамматик использовать функциональные грамматики, введенные профессором В.А. Тузовым.

1.2 Формальное описание логико-математического моделирования динамических систем

Математическая модель динамической системы [12, 87, 89] представляет собой совокупность:

$$M = \{\boldsymbol{U},\ \boldsymbol{T}\}, \tag{1}$$

в состав которой входит множество параметров:

$$\boldsymbol{U} = \{u_i\}, \quad i = 1,\ 2,\ \ldots,\ n \tag{2}$$

и базис операторов:

$$\boldsymbol{T} = \{T_j\}, \quad j = 1,\ 2,\ \ldots,\ m. \tag{3}$$

Параметры множества (2) можно разбить на четыре подмножества:

$$\boldsymbol{U} = \{\boldsymbol{V},\ \alpha,\ \beta,\ \gamma\}, \tag{4}$$

где $\boldsymbol{V}$ – фазовые переменные (координаты), определяющие состояние системы в любой заданный момент времени *t:*

$$\boldsymbol{V} = \{v_h(t)\}, \quad h = 1,\ 2,\ \ldots,\ n_1; \tag{5}$$

α – внешние параметры:

$$\alpha = \{\alpha_i\}, \quad i = 1,\ 2,\ \ldots,\ n_2;$$

β – внутренние параметры:

$$\beta = \{\beta_j\}, \quad j = 1,\ 2,\ \ldots,\ n_3;$$

γ – выходные параметры:

$$\gamma = \{\gamma_l\}, \quad l = 1,\ 2,\ \ldots,\ n_4;$$

$$n_1 + n_2 + n_3 + n_4 = n.$$

В состав множества фазовых переменных (5) входит три подмножества:

$$\boldsymbol{V} = \{\boldsymbol{X},\ \boldsymbol{Y},\ \boldsymbol{Z}\}, \tag{6}$$

где $\boldsymbol{X}$ – подмножество входных (внешних) фазовых переменных, образующих вектор входных воздействий:

$$\boldsymbol{X} = \{x_i(t)\}, \quad i = 1,\ 2,\ \ldots,\ n_{11};$$

$\boldsymbol{Y}$ – подмножество выходных фазовых переменных, образующих вектор реакций системы:

$$\boldsymbol{Y} = \{y_j(t)\}, \quad j = 1,\ 2,\ \ldots,\ n_{12};$$

$\boldsymbol{Z}$ – подмножество внутренних фазовых переменных:

$$\boldsymbol{Z} = \{z_l(t)\}, \quad l = 1,\ 2,\ \ldots,\ n_{13};$$

$$n_{11} + n_{12} + n_{13} = n_1.$$

Таким образом, множество параметров (2) с учетом (4) и (6) может быть записано следующим образом:

$$\boldsymbol{U} = \{\boldsymbol{V},\ \alpha,\ \beta,\ \gamma\} = \{\boldsymbol{X},\ \boldsymbol{Y},\ \boldsymbol{Z},\ \alpha,\ \beta,\ \gamma\}, \tag{7}$$

а математическая модель динамической системы (1) с учетом (7) – представлена в виде совокупности:

$$M = \{\boldsymbol{X},\ \boldsymbol{Y},\ \boldsymbol{Z},\ \alpha,\ \beta,\ \gamma,\ \boldsymbol{T}\}. \tag{8}$$

Базис операторов $\boldsymbol{T}$ (3) определяет закон функционирования системы, однозначно связывающий подмножество $\boldsymbol{X}$ входных фазовых переменных с подмножеством $\boldsymbol{Y}$ выходных фазовых переменных:

$$\boldsymbol{Y} = \boldsymbol{T}\boldsymbol{X}. \tag{9}$$

В зависимости от соотношений между $\boldsymbol{X}$, $\boldsymbol{T}$ и $\boldsymbol{Y}$ возможны несколько частных случаев функционирования математической модели системы:

1) если $\boldsymbol{T} = 0$ и при $\boldsymbol{X}$ 0 $\boldsymbol{Y} = 0$, то между подмножеством входов системы $\boldsymbol{X}$ и подмножеством выходов $\boldsymbol{Y}$ имеет место полный функциональный разрыв;

2) если $\boldsymbol{T} = 1$, то $\boldsymbol{Y} = \boldsymbol{X}$, т.е. имеет место тождественное преобразование подмножества входных фазовых переменных в подмножество выходов выходных фазовых переменных;

3) если $\boldsymbol{X} = 0$ и при любом $\boldsymbol{T} \neq 0$ $\boldsymbol{Y} \neq 0$, то элемент функциональной схемы, описываемый оператором $\boldsymbol{T}$, содержит внутренний источник энергии или информации;

4) если $\boldsymbol{X} = 0$ и при любом $\boldsymbol{T} \neq 0$ $\boldsymbol{Y} = 0$, то элемент функциональной схемы, описываемый оператором $\boldsymbol{T}$, не содержит внутренний источник энергии или информации.

Значения внешних параметров системы α определяют характеристики входных фазовых переменных. Поэтому вектор входных воздействий можно выразить следующим образом:

$$\boldsymbol{X} = \boldsymbol{X}(\alpha,\ t). \tag{10}$$

Значения внутренних параметров системы β характеризуют свойства функциональных подсистем, из которых состоит система. Поэтому параметры β связаны с операторами множества $\boldsymbol{T}$:

$$\boldsymbol{T} = \boldsymbol{T}(\beta). \tag{11}$$

Сформулированные выше соотношения (10) и (11) позволяют ввести математическую модель динамической системы. Так как формальное описание системы выражается операторным уравнением (9), то с учетом внешних (10) и внутренних (11) параметров это уравнение принимает вид:

$$\boldsymbol{Y}(t) = \boldsymbol{T}(\beta)\boldsymbol{X}(\alpha,\ t), \tag{12}$$

где α и β также могут быть функциями времени t.

Уравнение (12) может быть представлено в виде схемы (рисунок 1).

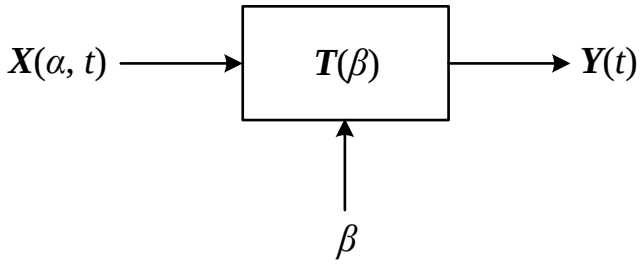

Рисунок 1 – Формальная схема математической модели динамической системы

Множество выходных параметров системы γ позволяет оценить функционирование системы, описанной выражением (12):

$$\gamma = F_1[\boldsymbol{Y}(t)] = F_1[\boldsymbol{T}(\beta)\boldsymbol{X}(\alpha,\ t)] = F_0(\boldsymbol{T},\ \alpha,\ \beta),$$

где F_1 и F_0 – определённые функциональные зависимости.

Для математической модели (12) могут быть заданы различные задачи моделирования, как прямые, так и обратные [23].

Прямая задача моделирования подразумевает отыскание следствий по известным причинам. В случае математического моделирования динамических систем, прямая задача – это анализ реакции системы на внешнее воздействие при известных внутренних параметрах системы. Прямая задача может быть описана выражениями:

$$\boldsymbol{Y}(t) = \boldsymbol{T}(\beta)\boldsymbol{X}(\alpha,\ t) \quad \text{или} \quad \gamma = F_1\left[\boldsymbol{T}(\beta)\boldsymbol{X}(\alpha,\ t)\right],$$

где F_1 – определённая функциональная зависимость.

Обратная задача рассматривает отыскание неизвестных причин по известным следствиям. Обратные задачи моделирования динамических систем могут быть двух типов:

1) анализ параметров внешнего воздействия по реакции системы на него и известных внутренних параметрах:

$$\boldsymbol{X}(\alpha,\ t) = \boldsymbol{T}^{-1}(\beta)\boldsymbol{Y}(t) \quad \text{или } \alpha = F_2\left[\boldsymbol{T}^{-1}(\beta)\boldsymbol{Y}(t)\right],$$

где F_2 – определённая функциональная зависимость.

2) анализ параметров моделируемой системы по известным внешнему

воздействию и реакции системы на него:

$$\boldsymbol{T}(\beta) = \boldsymbol{Y}(t)\boldsymbol{X}^{-1}(\alpha,\ t) \text{ или } \beta = F_3\left[\boldsymbol{Y}(t)\boldsymbol{X}^{-1}(\alpha,\ t)\right] \text{ или}$$

$$\boldsymbol{Z} = F_4\left[\boldsymbol{Y}(t)\boldsymbol{X}^{-1}(\alpha,\ t)\right],$$

где F_3 и F_4 – определённые функциональные зависимости.

Таким образом, целью логико-математического моделирования может являться любой из элементов математической модели (8): $\boldsymbol{X}$, $\boldsymbol{Y}$, $\boldsymbol{Z}$, α, β, γ, $\boldsymbol{T}$.

В общем виде задача логико-математического моделирования может быть задана совокупностью:

$$W = \{\boldsymbol{K},\ \boldsymbol{D},\ \boldsymbol{Q}\}, \tag{13}$$

состоящей из трех множеств:

1) множество понятий и отношений предметной области, составляющих базу знаний:

$$\boldsymbol{K} = \{k_l\},\quad l = 1,\ 2,\ \ldots,\ p;$$

2) множество входных переменных задачи, характеризующих исходные знания о модели:

$$\boldsymbol{D} = \{d_i\},\quad i = 1,\ 2,\ \ldots,\ r,$$

образующих вектор входных переменный задачи;

3) множество выходных переменных задачи, характеризующих цель моделирования:

$$\boldsymbol{Q} = \{q_j\},\quad j = 1,\ 2,\ \ldots,\ s,$$

образующих вектор выходных переменный задачи.

Если задача моделирования разрешима, то её можно представить в виде:

$$\left(\boldsymbol{K};\ d_1,\ d_2,\ \ldots,\ d_r \mapsto\ q_1,\ q_2,\ \ldots,\ q_s\ \right).$$

Таким образом, логико-математическое моделирование заключается в отображении вектора входных переменных $\boldsymbol{D}$ в вектор выходных переменных $\boldsymbol{Q}$ с помощью соотношений из базы знаний $\boldsymbol{K}$:

$$\boldsymbol{Q} = \boldsymbol{B}\boldsymbol{D}, \tag{14}$$

где $\boldsymbol{B}$ – оператор логического вывода.

Более подробно уравнение (13) может быть записано в следующем виде:

$$\{q_1,\ q_2,\ \dots,\ q_s\} = \boldsymbol{B}\left[\{d_1,\ d_2,\ \dots,\ d_r\}\right]. \qquad (15)$$

При этом, смысл множеств $\boldsymbol{Q}$, $\boldsymbol{D}$, $\boldsymbol{B}$ определяется условиями моделирования. Возможные варианты задач моделирования и способы логического вывода $\boldsymbol{B}$ цели моделирования $\boldsymbol{Q}$ через исходные данные $\boldsymbol{D}$ приведены в таблице 1.

Таблица 1 – Виды задач моделирования динамических систем

Вид задач моделирования	Цель моделирования $\boldsymbol{Q}$	Исходные данные моделирования $\boldsymbol{D}$	Оператор логического вывода $\boldsymbol{B}$
Прямые задачи	$\boldsymbol{Y}$	$\boldsymbol{T}, \boldsymbol{X}, \alpha, \beta$	$\boldsymbol{T}(\beta)\boldsymbol{X}(\alpha,\ t)$
	γ	$\boldsymbol{T}, \boldsymbol{X}, \alpha, \beta$	$F_1\left[\boldsymbol{T}(\beta)\boldsymbol{X}(\alpha,\ t)\right]$
Обратные задачи	$\boldsymbol{X}$	$\boldsymbol{T}, \boldsymbol{Y}, \beta$	$\boldsymbol{T}^{-1}(\beta)\boldsymbol{Y}(t)$
	α	$\boldsymbol{T}, \boldsymbol{Y}, \beta$	$F_2\left[\boldsymbol{T}^{-1}(\beta)\boldsymbol{Y}(t)\right]$
	$\boldsymbol{T}$	$\boldsymbol{X}, \boldsymbol{Y}, \alpha$	$\boldsymbol{Y}(t)\boldsymbol{X}^{-1}(\alpha,\ t)$
	β	$\boldsymbol{X}, \boldsymbol{Y}, \alpha$	$F_3\left[\boldsymbol{Y}(t)\boldsymbol{X}^{-1}(\alpha,\ t)\right]$
	$\boldsymbol{Z}$	$\boldsymbol{X}, \boldsymbol{Y}, \alpha$	$F_4\left[\boldsymbol{Y}(t)\boldsymbol{X}^{-1}(\alpha,\ t)\right]$

При подстановки фактических параметров $\boldsymbol{D}'$ в модель (15) будет получен результат $\boldsymbol{Q}'$. Часто необходимо, чтобы он был как можно ближе к какому-то требуемому значению $\boldsymbol{Q}^*$:

$$\boldsymbol{Q}' \to \boldsymbol{Q}^* \quad \text{или}$$

$$\left|\boldsymbol{Q}' - \boldsymbol{Q}^*\right| \to min. \qquad (16)$$

Таким образом, модель (14) с учетом условия (15) можно записать в виде:

$$\begin{cases} \boldsymbol{Q}' = \boldsymbol{B}\boldsymbol{D}'; \\ \boldsymbol{Q}' - \boldsymbol{Q}^* \to\ min. \end{cases}$$

Для выполнения системы (16) может понадобиться итерационная корректировка фактических параметров $\boldsymbol{D}'$ (рисунок 2).

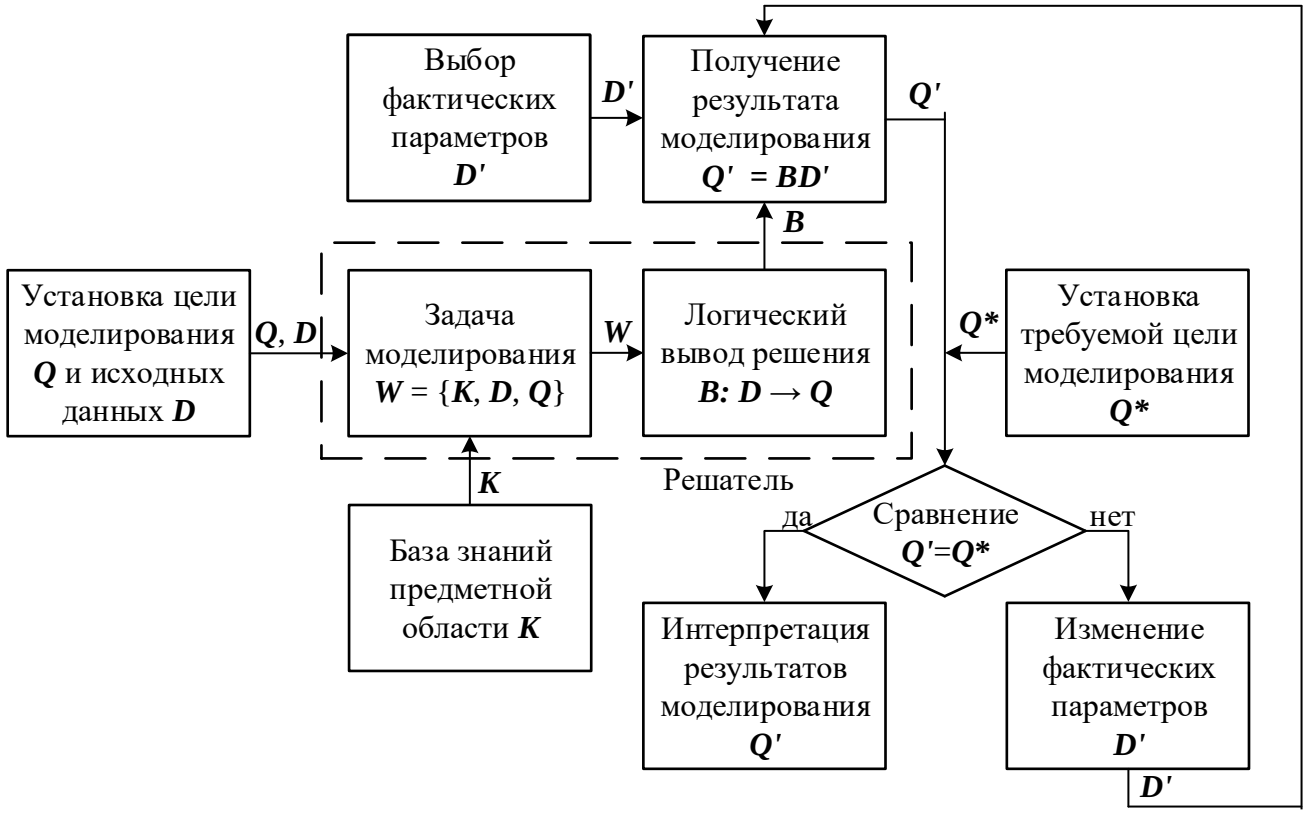

Рисунок 2 – Блок-схема логико-математического моделирования динамических систем

1.3 Обоснование применения функциональных грамматик в логико-математическом моделировании

При дедуктивном синтезе логико-математического моделирования используется представление знаний в виде семантических сетей, основанных на атрибутных грамматиках.

Как указано, например, в [84] формализм атрибутных грамматик является удобным средством для описания семантики языков программирования, но реализация вычислителей на атрибутных грамматиках сталкивается с большими трудностями. К таким трудностям можно отнести несоответствие чисто функциональной природы атрибутного вычислителя и связанной с ней неупорядоченностью процесса вычисления атрибутов и упорядоченностью элементов программы.

В работе [93] были введены функциональные грамматики. Они являются

дальнейшим развитием атрибутных грамматик и используются в математической лингвистике [63, 64, 96]. Автор предлагает использовать функциональные грамматики в логико-математическом моделировании [50, 51].

Преимущества функциональных грамматик над атрибутными грамматиками и другими методами при описании языков программирования было показано в монографии [94]. Данный сравнительный анализ вполне применим и при описании баз знаний ***K***, так как технически реализованная база знаний представляет собой модуль либо на имеющемся, либо на специально созданном языке программирования.

Результат сравнительного анализа показан в таблице 2. При этом знаком «+» указана принадлежность свойства методу, знаком «–» – отсутствие принадлежности, «0» – способность моделирования свойства другими инструментами метода.

Таблица 2 – Анализ методов описания языков программирования [94]

№	Свойство	Атрибутные грамматики	Венский метод	Грамматики ван Вейнгаардена	Продукционные системы	Аксиомати-ческий подход	Функциональные грамматики
1	Описание синтаксиса	+	+	+	+	–	+
2	Операция отложенного выполнения	–	–	–	–	–	+
3	Описание структуры данных	–	0	+	–	–	+
4	Операция отождествления	–	–	+	–	–	+
5	Память с варьируемой структурой	–	–	–	–	–	+
6	Память с постоянной структурой	0	+	0	+	–	+
7	Рекурсия	0	+	0	+	–	+
8	Альтернативная операция	–	+	–	+	–	+
9	Управляющая память	–	–	–	–	–	+
10	Система приведений	–	–	–	–	0	+
11	Многоуровневость	–	–	–	–	–	+

Согласно указанному анализу, аппарат функциональных грамматик дает самое полное описание систем описания баз знаний. Вообще, все виды грамматик изоморфно вкладываются в функциональную грамматику, т.е. функциональные грамматики являются обобщением всех методов, указанных выше, включая атрибутные грамматики. Таким образом, использование функциональных грамматик позволяет получить более гибкую систему, чем при использовании атрибутных грамматик.

Кроме того, в [94] были указаны такие недостатки атрибутных грамматик, как неполнота, сложность описания, отсутствие ясности в описании семантики, низкие способности обнаружения ошибок и освещения деталей.

Согласно [31, 97], при использовании атрибутных грамматик результат логико-математического моделирования транслируется на специально созданный язык программирования. Это является недостатком, так как часто требуется, чтобы модель могла быть трансформирована в программу на необходимом языке программирования. При использовании функциональных грамматик логико-математическая модель представляет собой суперпозицию функций, которая может быть трансформировать в функциональную программу на выбранном языке.

Также в качестве недостатка логико-математического моделирования на основе атрибутных грамматик можно указать императивность получаемой модели и вследствие этого – последовательный порядок выполнения действий. Применение функциональных грамматик позволяет представлять математическую модель в функциональном виде, позволяющем использовать распараллеливание вычислений.

Таким образом, недостатками логико-математического моделирования на основе атрибутных грамматик перед логико-математическом моделировании на основе функциональных грамматик являются:

1) меньшая гибкость атрибутных грамматик по сравнению с функциональными грамматиками (таблица 2) при описании знаний;

2) ограниченность применения логико-математической модели только одним узкоспециализированным языком программирования;

3) императивность логико-математической модели и языка программирования и последовательный порядок выполнения действий.

Принимая во внимание перечисленные недостатки использования атрибутных грамматик, предлагается для логико-математического моделирования динамических систем использовать представление знаний в виде функциональных грамматик.

1.4 Метод декомпозиции задачи моделирования в условиях применения функциональной парадигмы

Главной проблемой логико-математического моделирования является декомпозиция оператора $\boldsymbol{B}$, т.е. представление его совокупностью элементарных операций, принадлежащих базе знаний $\boldsymbol{K}$:

$$B_i \in \boldsymbol{K},\ i = 1,\ 2,\ \ldots,\ t.$$

В работах [97, 31] для логического вывода (доказательства) решения используется последовательный способ декомпозиции.

Если каждая операция B_i оператора $\boldsymbol{B}$ будет описана уравнением, подобным (13), то доказательство решения задачи моделирования можно представить последовательным набором таких операций. Общее уравнение при этом будет иметь вид:

$$\boldsymbol{Q} = B_t B_{t-1} \ldots B_2 B_1 \boldsymbol{D}, \qquad (16)$$

где B_t, B_{t-1}, ... , B_2, B_1 – операторы элементарных операций.

Каждая элементарная операция B_i описывается своим операторным уравнением, подобным (13), векторы $\boldsymbol{D}$ и $\boldsymbol{Q}$ в котором заменены (рисунок 3) соответствующими векторами промежуточных переменных $\boldsymbol{C}_1$, $\boldsymbol{C}_2$, ... $\boldsymbol{C}_{t-2}$, $\boldsymbol{C}_{t-1}$:

$$\boldsymbol{C}_1 = B_1 \boldsymbol{D},\ \ \boldsymbol{C}_2 = B_2 \boldsymbol{C}_1,\ \ \ \boldsymbol{C}_3 = B_3 \boldsymbol{C}_2,\ \ \ldots\ ,\ \ \ \boldsymbol{C}_{t-1} = B_{t-1} \boldsymbol{C}_{t-2},\ \ \ \boldsymbol{Q} = B_t \boldsymbol{C}_{t-1}.$$

Или в более общем виде:

$$\boldsymbol{C}_i = B_i \boldsymbol{C}_{i-1},\ \ i = 1,\ 2,\ \ldots,\ t \ \text{ при } \ \boldsymbol{C}_0 = \boldsymbol{D},\ \ \boldsymbol{C}_t = \boldsymbol{Q}.$$

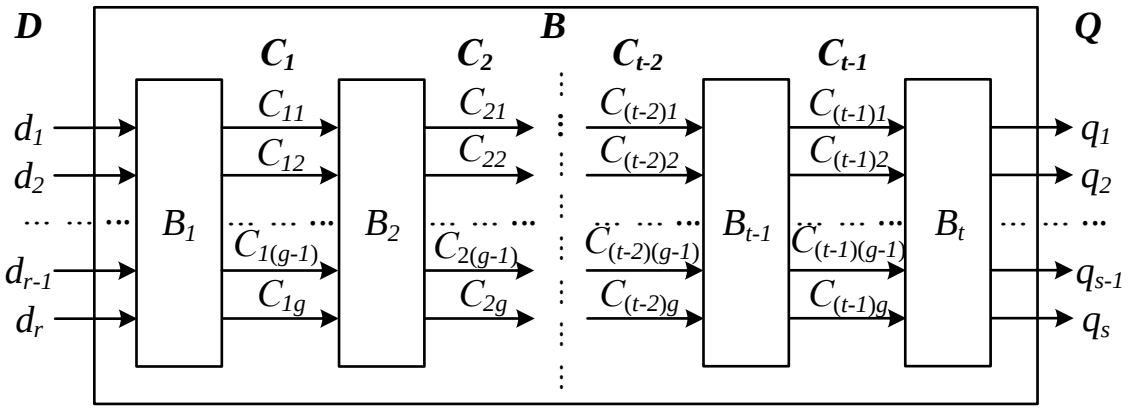

Рисунок 3 – Последовательный способ декомпозиции оператора ***B***

Каждый из векторов промежуточных переменных $\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots \boldsymbol{C}_{t-2}, \boldsymbol{C}_{t-1}$ максимально может содержать по g элементов. Так как каждый из выходов q_j может быть связан со всеми входами d_i, то число g равно произведению количества входных переменных r на количество выходных переменных s:

$$g = r \cdot s.$$

При логико-математическом моделировании с применением функциональных грамматик основой базы знаний ***K*** являются функции f_j. Каждая функция имеет всего один выходной параметр (результат), поэтому элементарные операции декомпозиции оператора ***B*** также должны иметь один выход (рисунок 4).

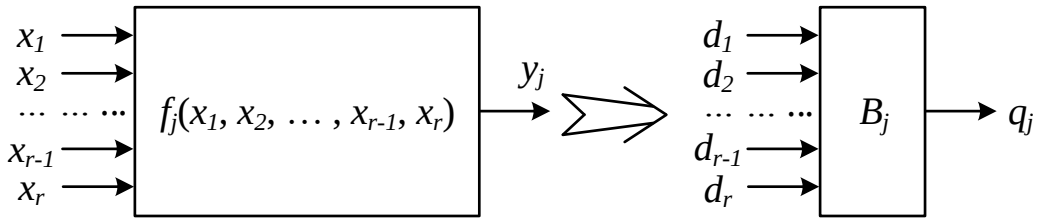

Рисунок 4 – Элементарная операция декомпозиции при использовании функциональных грамматик

Таким образом, последовательный способ декомпозиции оператора ***B*** (16) не применим при использовании функциональных грамматик. Поэтому введем последовательно-параллельный способ [52], при котором каждая операция B_i.

заменяется параллельным соединением нескольких более простых операций B_{ij}:

$$C_{ij} = B_{ij}\left[\{C_{(i-1)1},\ C_{(i-1)2},\ \ldots,\ C_{(i-1)g}\}\right].$$

При этом операция B_t разбивается на s операций $B_{t1}, B_{t2}, \ldots, B_{ts}$ а каждая из последовательных операций $B_{t1}, B_{t2}, \ldots, B_{t-1}$ – на g блоков $B_{i1}, B_{i2}, \ldots, B_{ig}$, где $i = 1,\ 2,\ \ldots,\ t-1$.

В результате, единое уравнение (14) преобразуется к системе из g уравнений:

$$q_j = \boldsymbol{B}_j\left[\{d_1,\ d_2,\ \ldots,\ d_r\}\right];\quad j = 1,\ 2,\ \ldots,\ g.$$

Общее операторное уравнение последовательно-параллельной декомпозиции (рисунок 5) будет иметь вид:

$$\boldsymbol{Q} = \begin{pmatrix} B_{t1} & B_{(t-1)1} & \ldots & B_{21} & B_{11} \\ B_{t2} & B_{(t-1)2} & \ldots & B_{22} & B_{12} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ B_{t(s-1)} & \ldots & \ldots & \ldots & \ldots \\ B_{ts} & B_{(t-1)(g-1)} & \ldots & B_{2(g-1)} & B_{1(g-1)} \\ & B_{(t-1)g} & \ldots & B_{2g} & B_{1g} \end{pmatrix} \boldsymbol{D},$$

где $B_{t1},\ B_{(t-1)1},\ \ldots,\ B_{2g},\ B_{1g}$ – операторы отдельных элементарных операций.

Каждая операция B_{ij} описывается операторным уравнением вида:

$$\boldsymbol{C}_{ij} = B_{ij}\boldsymbol{C}_{i-1} \text{ при } \boldsymbol{C}_0 = \boldsymbol{D},\ \boldsymbol{C}_t = \boldsymbol{Q},\ i = 1,\ 2,\ \ldots,\ t,\ j = 1,\ 2,\ \ldots,\ s.$$

Последовательно-параллельный способ позволяет использовать в качестве операторов B_{ij} функциональные зависимости, связывающие один результатов с несколькими аргументами (рисунок 4). Это дает возможность применять функциональные грамматики для формулировки задачи моделирования и доказательства решения (первый и второй этапы логико-математического моделирования в виде дедуктивного синтеза).

По последовательно-параллельному выводу доказательства автоматически можно построить логико-математическую модель (третий этап дедуктивного синтеза) в виде суперпозиции функций:

$$\sigma = \sigma(f_1,\ f_2,\ \ldots,\ f_P),\ p < t \cdot g,$$

где $f_1,\ f_2,\ \ldots,\ f_P \in \boldsymbol{K}$ – элементарные функции базы знаний $\boldsymbol{K}$, определяющие операции B_{ij}.

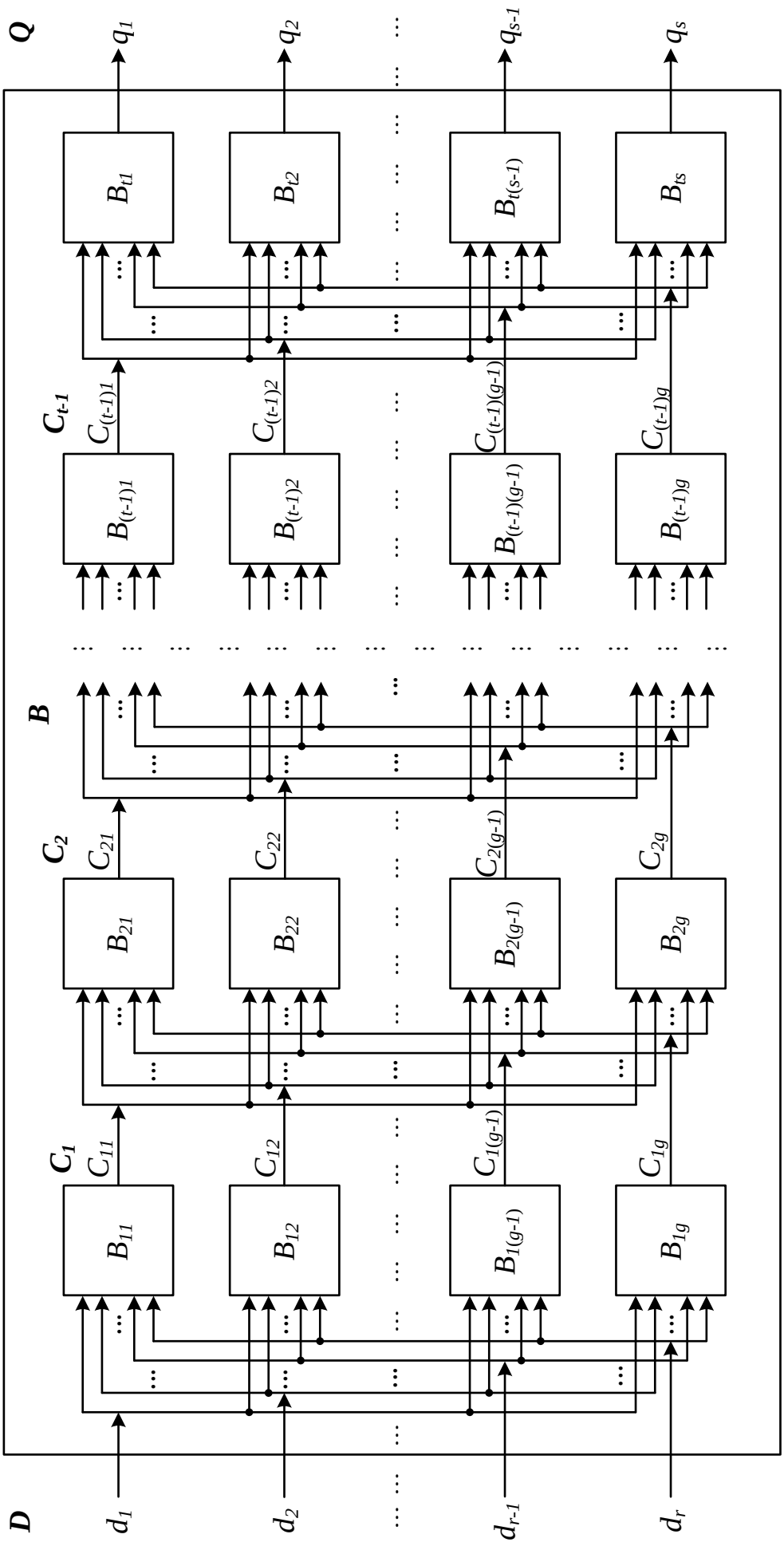

Рисунок 5 – Последовательно-параллельный способ декомпозиции оператора ***B***

Каждая элементарная функция зависит только от своих входных параметров (аргументов) и не зависит от каких-либо глобальных переменных. В итоге результат применения любой функции при одинаковых аргументах не зависит от порядка применения функций. Таким образом, последовательная-параллельная декомпозиция обладает полной модульностью, в отличие от последовательной декомпозиции, где имеет значение порядок следования элементарных операций.

Представление элементарных операций в виде функций позволяет использовать функциональные языки программирования вместо императивных. При этом суперпозиция функций не требует каких-либо сложных действий для перехода к программному коду (этап 4), так как сама является готовой программой на функциональном языке программирования. Использование функционального языка программирования позволяет эффективно использовать параллельные вычисления.

В таблице 3 представлены преимущества последовательно-параллельной декомпозиции над последовательной.

Таблица 3 – Сравнение двух методов декомпозиции системного оператора ***B***

Критерий сравнения	Последовательная декомпозиция	Последовательно-параллельная декомпозиция
Способ представления знаний и описания задачи моделирования	Атрибутные грамматики	Функциональные грамматики
Вид логико-математической модели	Последовательность операций $B_1 \Rightarrow B_2 \Rightarrow \ \ldots \ \Rightarrow B_{t-1} \Rightarrow B_t$	Суперпозиция функций $\sigma = \sigma(f_1,\ f_2,\ \ldots,\ f_P),\ p < t \cdot g$
Математический аппарат доказательства решения задачи и вывода решения	Математическая логика (логика высказываний, логика предикатов и др.)	Лямбда-исчисление
Модульность	Частичная – результат применения операции зависит от порядка операций	Полная – результат каждой функции зависит только от своих аргументов

Продолжение таблицы 3

Критерий сравнения	Последовательная декомпозиция	Последовательно-параллельная декомпозиция
Программная реализация	Программа на императивном языке программирования	Программа на функциональном языке программирования
Возможность параллельных вычислений	Изначально отсутствует и требует применения специальных операций распараллеливания	Существует и реализуется в рамках языка функционального языка программирования

ГЛАВА 2. МЕТОДЫ ЛОГИКО-МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ НА ОСНОВЕ ИСПОЛЬЗОВАНИЯ ФУНКЦИОНАЛЬНЫХ ГРАММАТИК

2.1 Элементы теории формальных грамматик

Выделим и рассмотрим основные элементы теории формальных грамматик, описанной, например, в [6, 14, 21, 61, 98] которые необходимы для ввода понятия функциональных грамматик.

Формальная грамматика – это упорядоченная совокупность четырех элементов:

$$G_K = \{V_{TK},\ V_{NK},\ P_K,\ S_K\},$$

где V_{TK} – конечный алфавит терминальных символов (терминалов);

V_{NK} – конечный алфавит нетерминальных символов (нетерминалов), не пересекающийся с V_T;

P_K – конечное множество правил вывода (продукций);

S_K – начальный нетерминальный символ (аксиома), $S_K \in V_{NK}$.

Тип грамматики зависит от используемых в ней правил вывода. В общем случае, правила вывода имеют вид:

$$\alpha \rightarrow \beta,$$

где $\alpha \in (V_{TK} \cup V_{NK})^+$ – множество всех цепочек конечной длины в объединенном алфавите терминальных и нетерминальных символов, за исключением пустой цепочки ε;

$\beta \in (V_{TK} \cup V_{NK})^*$ – множество всех цепочек конечной длины в том же алфавите, включая пустую цепочку ε.

Контекстно-свободная грамматика (формальная грамматика второго типа по иерархии Хомского) имеет ограничение на вид правил, заключающееся в том, что их левые части должны представлять собой одиночные нетерминальные символы, т.е.:

$$A \to \psi,$$

где $A \in V_{NK}$, $\psi \in (V_{TK} \cup V_{NK})^*$.

Когда ни одно из правил множества P_K грамматики G_K не содержит в правой части пустую цепочку ε, тогда такую грамматику называют контекстно-свободной грамматикой, ее правила имеют вид:

$$A \to \psi,$$

где $A \in V_{NK}$, $\psi \in (V_{TK} \cup V_{NK})^+$.

Далее будем рассматривать именно контекстно-свободные грамматики.

Если несколько продукций имеют одинаковую левую часть:

$$A \to \psi_1, \quad A \to \psi_2, \quad \dots, \quad A \to \psi_n,$$

то может быть использована сокращенная запись:

$$A \to \psi_1 \mid \psi_2 \mid \dots \mid \psi_n,$$

обозначающая набор альтернатив.

Применение одного правила $\alpha \to \beta$ из множества P_K позволяет осуществить непосредственный вывод $\gamma \to \delta$ цепочки символов $\delta \in (V_{TK} \cup V_{NK})^+$ из цепочки $\gamma \in (V_{TK} \cup V_{NK})^+$. Для этого необходимо выполнение условия: $\gamma = \xi_1 \alpha \xi_2$, $\delta = \xi_1 \beta \xi_2$, где ξ_1, ξ_2, $\beta \in (V_{TK} \cup V_{NK})^+$, $\alpha \in V_{NK}$. При этом говорят, что δ непосредственно выводима из γ.

Последовательное применение нескольких правил из множества P позволяет осуществить вывод $\gamma \Rightarrow \varphi$ цепочки символов $\varphi \in (V_{TK} \cup V_{NK})^+$ из цепочки $\gamma \in (V_{TK} \cup V_{NK})^+$. Для этого должны существовать цепочки δ_0, δ_1, δ_2, ... ,δ_n $(n \geq 0)$, для которых выполняется условие: $\delta_0 \to \delta_1 \to \delta_2 \to \dots \to \delta_n$, $\delta_0 = \gamma$, $\delta_n = \varphi$. В этом случае говорят, что φ выводима из γ.

Если каждая цепочка δ_0, δ_1, δ_2, ... ,δ_n в выводе $\gamma \Rightarrow \varphi$ получается из предыдущей заменой самого левого нетерминального символа, то такой вывод называют левосторонним выводом, если же заменой самого правого нетерминала, то – правосторонним. Далее будем рассматривать только левосторонние выводы.

Если цепочка φ состоит только из терминальных символов, т.е. $\varphi \in V_{TK}^{+}$, а цепочка γ состоит только из одного начального нетерминального символа S, т.е. $\gamma = S \in V_{NK}$, то ее называют правильной цепочкой, а вывод $\gamma \to \delta_1 \to \delta_2 \to ... \to \delta_{n-1}, \to \varphi$ – полным выводом. Если же цепочка имеет в своем составе нетерминальные символы, т.е. $\varphi \in (V_{TK} \cup V_{NK})^{+}$, а в множестве продукций P_K нет правил применимых к φ, то такой вывод называют тупиковым.

Множество всех правильных цепочек в грамматике G_K образуют язык $L(G_K)$, т.е. $L(G_K) = \{\varphi \in V_{TK}^{*} \mid S_K \Rightarrow \varphi\}$.

Любому полному выводу в грамматике G_K можно сопоставить дерево вывода, являющееся удобной формой его графического представления. Для контекстно-свободной грамматики дерево должно удовлетворять двум условиям. Во-первых, каждый узел дерева должен быть помечен одним символом из множества $(V_{TK} \cup V_{NK})$, при этом каждая вершина дерева – символом из V_{TK}, а корень – начальным нетерминальным символом $S_K \in V_{NK}$. Во-вторых, если узел дерева помечен символом $A \in V_{NK}$, а его непосредственные потомки – символами a_1, a_2, ... , a_n, где каждый символ $a_i \in (V_{TK} \cup V_{NK})$, то $A \to a_1 a_2 \ ... \ a_n$ – правило вывода в G_K. Схематичное изображение дерева представлено на рисунке 6.

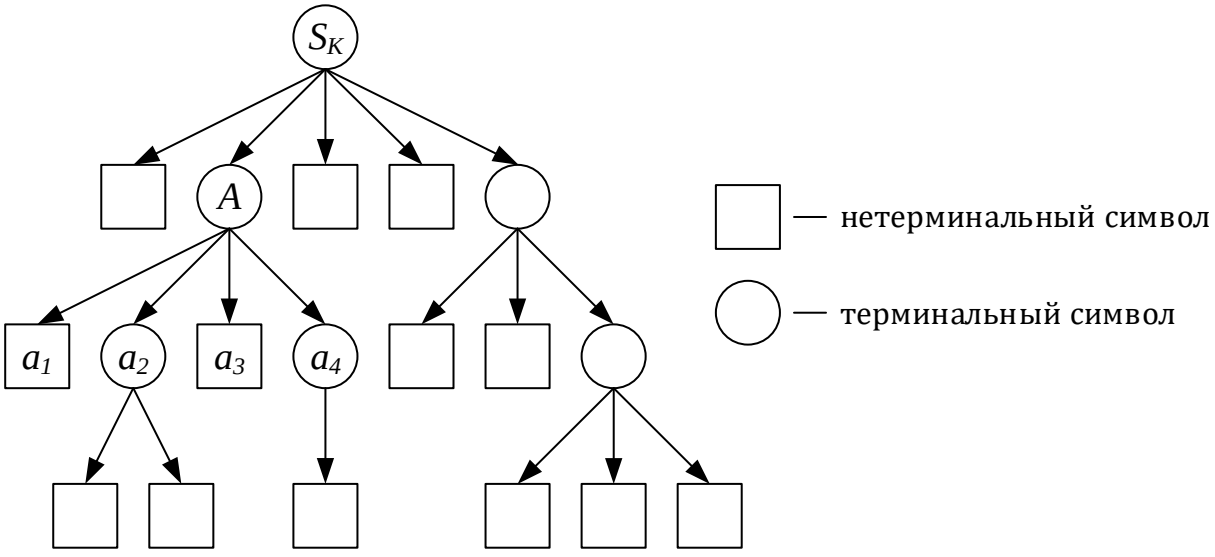

Рисунок 6 – Схематичное изображение дерева вывода

Рассмотрим пример грамматики и принципы построения вывода и его дерева. Пусть задана контекстно-свободная грамматика G_{K1}:

$$G1_K = \{V1_{TK},\ V1_{NK},\ P1_K,\ S1_K\};$$

$$V1_{TK} = \{a,\ b,\ c,\ d,\ e,\ \alpha,\ \beta,\ \gamma,\ \delta\};\quad V1_{NK} = \{S,\ A,\ B,\ C,\ D,\ E\};\quad S1_K = S;$$

$$P1_K = \begin{Bmatrix} S \to A\alpha a\beta B; \\ A \to b\beta C\alpha c \mid \gamma d\beta e\alpha D; \\ B \to \gamma a\beta b\alpha c \mid d\alpha e\beta a\delta; \\ C \to \gamma E\alpha b; \\ D \to c\beta E\delta; \\ E \to \gamma d\alpha e. \end{Bmatrix}.$$

Построим левосторонний полный вывод правильной цепочки:

$$S \to A\alpha a\beta B \to b\beta C\alpha c\alpha a\beta B \to b\beta\gamma E\alpha b\alpha c\alpha a\beta B \to b\beta\gamma\gamma d\alpha e\alpha b\alpha c\alpha a\beta B \to$$

$$\to b\beta\gamma\gamma d\alpha e\alpha b\alpha c\alpha a\beta\gamma a\beta b\alpha c\ .$$

Этот вывод позволяет получить одну правильную цепочку (слово) языка $L(G1_K)$, порожденного грамматикой $G1_K$. Для получения остальных слов необходимо при замене самых левых нетерминалов выбирать альтернативные правые части продукций $A \to b\beta C\alpha c \mid \gamma d\beta e\alpha D$ и $B \to \gamma a\beta b\alpha c \mid d\alpha e\beta a\delta$:

$$S \to A\alpha a\beta B \to \gamma d\beta e\alpha D\alpha a\beta B \to \gamma d\beta e\alpha c\beta E\delta\alpha a\beta B \to \gamma d\beta e\alpha c\beta\gamma d\alpha e\delta\alpha a\beta B \to$$

$$\to \gamma d\beta e\alpha c\beta\gamma d\alpha e\delta\alpha a\beta\gamma a\beta b\alpha c\ ;$$

$$S \to A\alpha a\beta B \to b\beta C\alpha c\alpha a\beta B \to b\beta\gamma E\alpha b\alpha c\alpha a\beta B \to b\beta\gamma\gamma d\alpha e\alpha b\alpha c\alpha a\beta B \to$$

$$\to b\beta\gamma\gamma d\alpha e\alpha b\alpha c\alpha a\beta d\alpha e\beta a\delta\ ;$$

$$S \to A\alpha a\beta B \to \gamma d\beta e\alpha D\alpha a\beta B \to \gamma d\beta e\alpha c\beta E\delta\alpha a\beta B \to \gamma d\beta e\alpha c\beta\gamma d\alpha e\delta\alpha a\beta B \to$$

$$\to \gamma d\beta e\alpha c\beta\gamma d\alpha e\delta\alpha a\beta\gamma a\beta b\alpha c\ .$$

Таким образом, в грамматике $G1_K$ возможно выполнить 4 полных вывода, т.е. язык $L(G1_K)$ содержит четыре слова:

$$L(G1_K) = (b\beta\gamma\gamma d\alpha e\alpha b\alpha c\alpha a\beta\gamma a\beta b\alpha,\ \gamma d\beta e\alpha c\beta\gamma d\alpha e\delta\alpha a\beta\gamma a\beta b\alpha c,$$

$$b\beta\gamma\gamma d\alpha e\alpha b\alpha c\alpha a\beta d\alpha e\beta a\delta,\ \gamma d\beta e\alpha c\beta\gamma d\alpha e\delta\alpha a\beta\gamma a\beta b\alpha c).$$

Каждому из выводов соответствует дерево разбора. На рисунке 7 приведено дерево первого полного вывода, а на рисунке 8 – второго полного вывода.

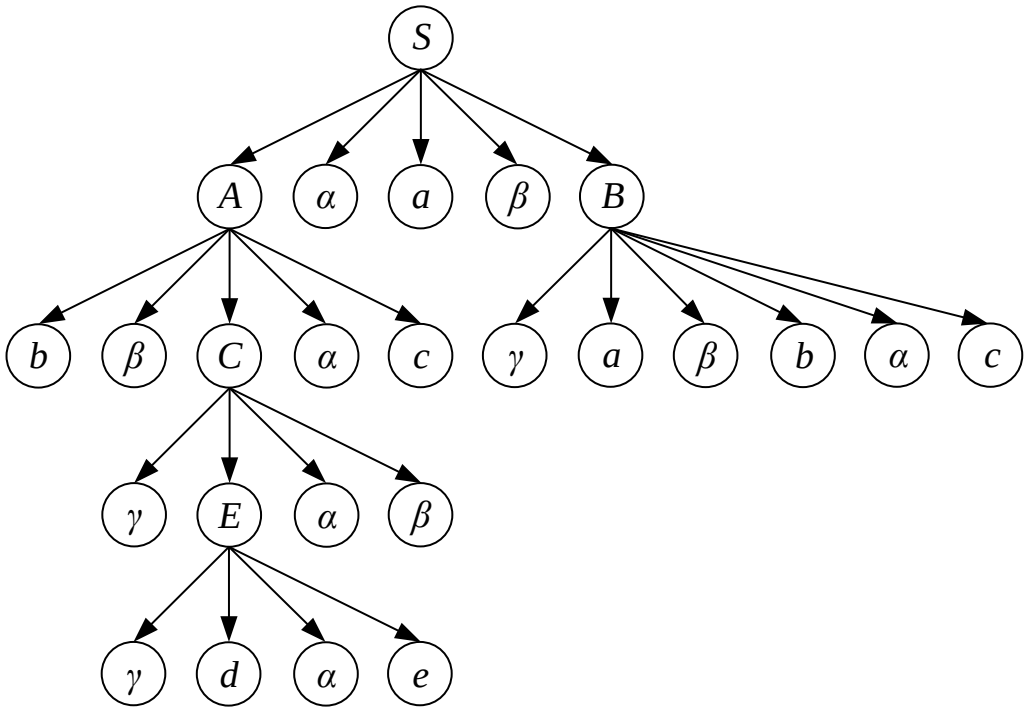

Рисунок 7 – Дерево первого полного вывода в грамматике G_{K1}

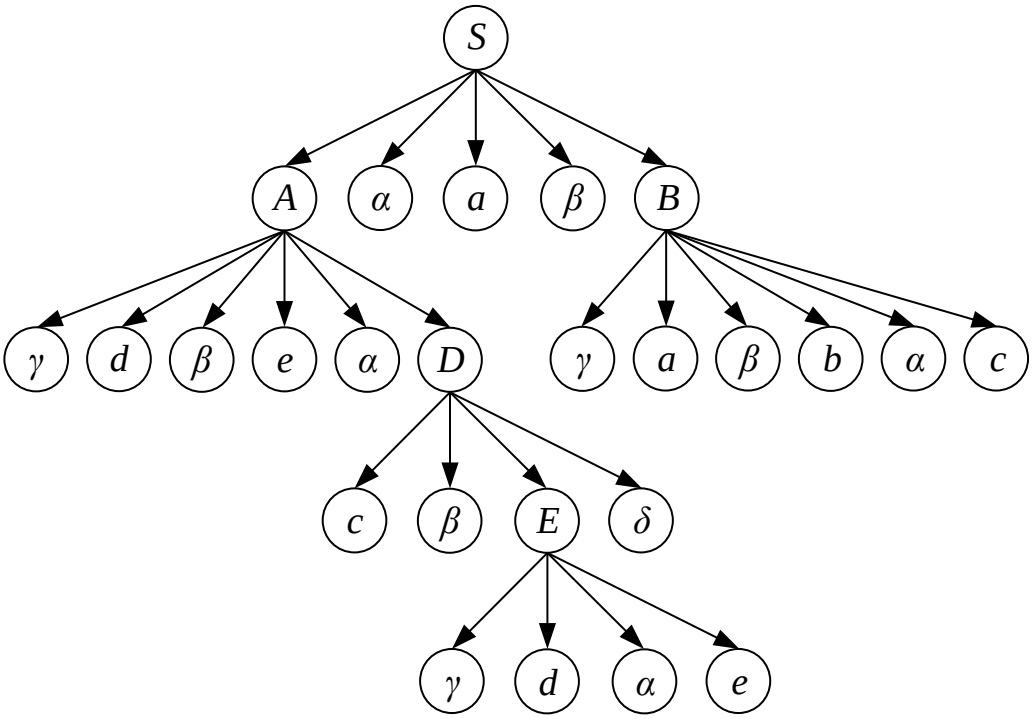

Рисунок 8 – Дерево второго полного вывода в грамматике G_{K1}

Как видно из рисунков, дерево разбора удобно для построения и демонстрации полного вывода правильной цепочки в грамматике. Однако у деревьев вывода можно выделить два недостатка. Во-первых, по дереву вывода неудобно следить за каждым отдельным выводом или, иначе говоря, за цепочкой каждого уровня дерева. Для этого необходимо отыскивать все вершины дерева на данном уровне, расставляя их в нужном порядке. Во-вторых, дерево вывода соответствует лишь одному выводу, поэтому для описания всех слов языка необходимо составлять такое количество деревьев, которое равно количеству слов в языке.

Автор рекомендует, в случае принципиального значения указанных недостатков, вместо конечного множества деревьев вывода использовать одно дерево перебора. Каждый узел такого дерева должна быть помечен цепочкой символов из множества $(V_{TK} \cup V_{NK})^+$, при этом каждая вершина дерева – цепочкой из $V_{TK}{}^+$, а корень – начальным нетерминальным символом $S_K \in V_{NK}$. Если узел дерева перебора помечен цепочкой символов $\xi_1 A \xi_2$, где $A \in V_{NK}$, $\xi_1, \xi_2 \in (V_{TK} \cup V_{NK})^*$, а его непосредственные потомки – цепочками символов $\xi_1\psi_1\xi_2$, $\xi_1\psi_2\xi_2$, ... , $\xi_1\psi_n\xi_2$, где каждая цепочка $\psi_i \in (V_{TK} \cup V_{NK})^+$, то $A \to \psi_1 | \psi_2 | \ ... \ | \psi_n$ – правило вывода в грамматике G_K. Схематичное изображение дерева перебора представлено на рисунке 9.

Для рассмотренной выше грамматики $G1_K$ дерево перебора представлено на рисунке 10. Данное дерево имеет четыре вершины, соответствующие четырем словам языка $L(G1_K)$, порожденного грамматикой $G1_K$.

2.2 Определение функциональных грамматик

В работе [93] В.А. Тузовым было введено понятие функциональной грамматики. Каждой продукции контекстно-свободной грамматики в соответствие поставлена определенная функция.

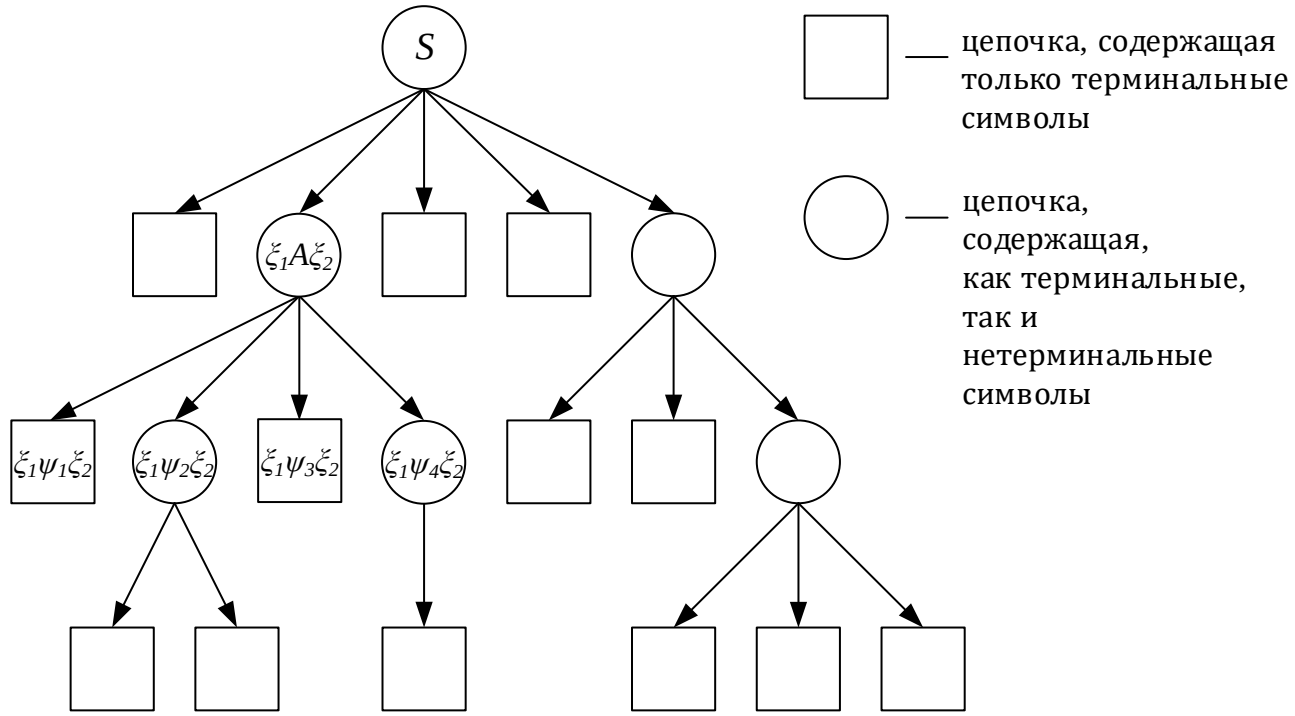

Рисунок 9 – Схематичное изображение дерева перебора

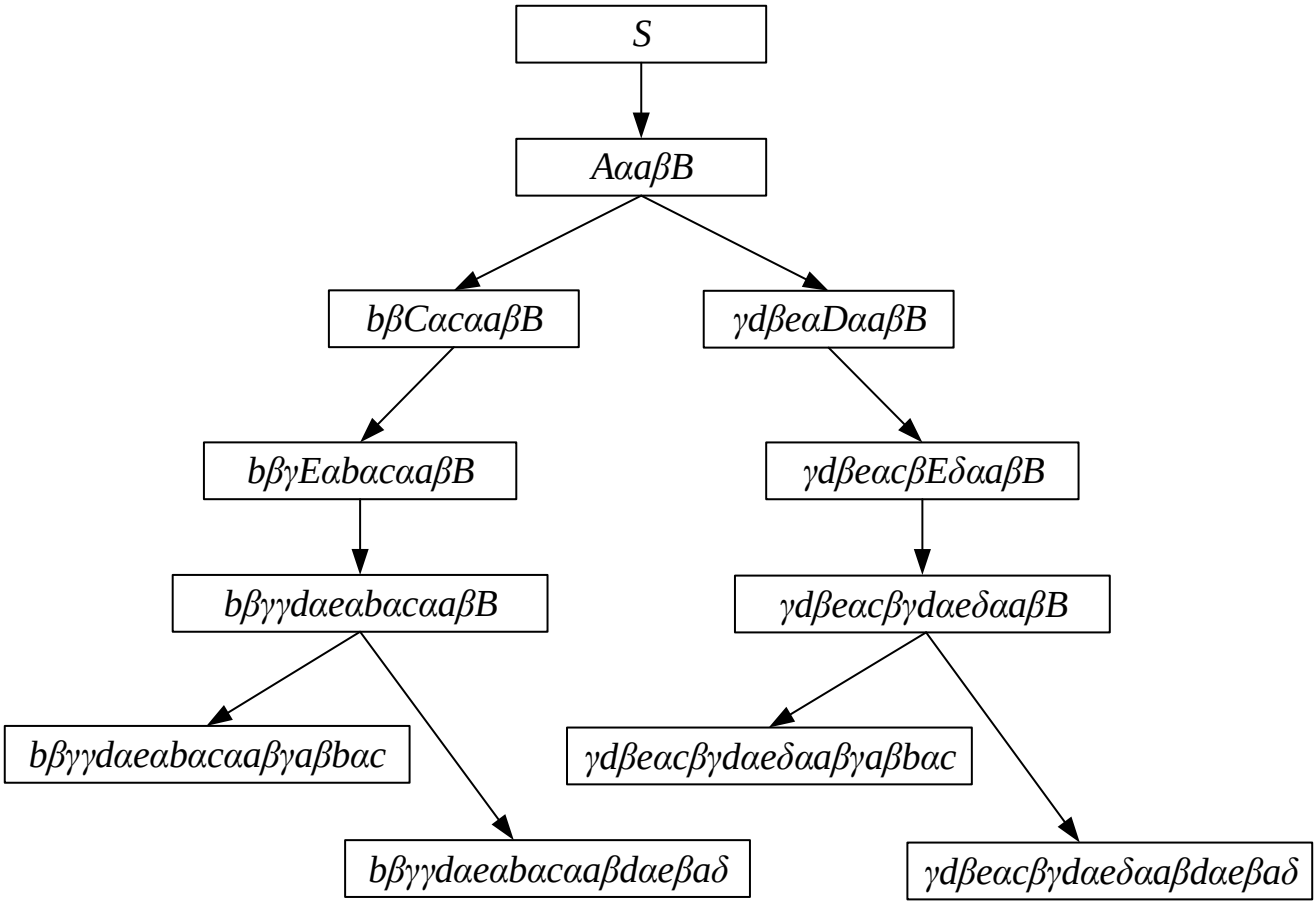

Рисунок 10 – Дерево перебора для грамматики G_{K1}

Применение каждого правила вывода является эквивалентным вызову функций с аргументами, перечисленными в его правой части. В данной диссертационной работе подход В.А. Тузова используется для логико-математического моделирования динамических систем.

Дадим определение функциональных грамматик, использующихся в работе.

Определение 1. Функциональная (полная) контекстно-свободная грамматика – это совокупность:

$$G_F = \{V_T,\ V_N,\ P,\ F,\ F0,\ V_0,\ S\},$$

состоящая из множества (алфавита) терминальных символов:

$$V_T = \{w_j\}, \quad j = 1,\ 2,\ \ldots,\ n_{11};$$

множества (алфавита) нетерминальных символов:

$$V_N = \{e_l\}, \quad l = 1,\ 2,\ \ldots,\ n_{12};$$

множества (алфавита) базисных терминальных символов:

$$V_0 = \{b_i\}, \quad i = 1,\ 2,\ \ldots,\ n_0;$$

множества базисных унарных и бинарных функций:

$$F_0 = \{f_{0i}\}, \quad i = 1,\ 2,\ \ldots,\ n_0 \quad \text{вида:}$$

$$f_{0i} = f(x_1,\ x_2) = x_1 b_i x_2, \quad f_{0i} = f(x_1) = b_i x_1 \ \text{ или } \ f_{0i} = f(x_1) = x_1 b_i;$$

множества функций:

$$F = \{f_r\}, \quad r = 1,\ 2,\ \ldots,\ k \quad \text{вида:}$$

$$f_r = f(x_1,\ x_2,\ \ldots,\ x_p) = \sigma(x_1,\ x_2,\ \ldots,\ x_p,\ f0_1,\ f0_2,\ldots,\ f0_q);$$

множества правил:

$$P = \{P_s\}, \quad s = 1,\ 2,\ \ldots,\ m \quad \text{вида:}$$

$$A_j \rightarrow \varphi_s\ \{f_r\},$$

где $\varphi_s \in V^+$ – произвольная последовательность символов объединенного алфавита $V \equiv V_T \cup V_N$, являющаяся совокупностью фактических параметров аргументов $x_1,\ x_2,\ \ldots,\ x_p$ функции f_r.

и начального нетерминального символа (аксиомы) $S \in V_N$.

Функциональные грамматики являются дальнейшем развитием атрибутных грамматик, впервые введенных Д.Э. Кнутом [114].

Атрибутная грамматика $G_A = \{G_K,\ A_S,\ A_I,\ R\}$ представляет собой контекстно-свободную грамматику $G_K = \{V_{TK},\ V_{NK},\ P_K,\ S_K\}$, в которой с каждым терминальным и нетерминальным символом связано конечное множество наследуемых A_I и синтезируемых A_S атрибутов. Атрибуты связаны между собой конечным множеством семантических правил R.

Формально функциональная грамматика является атрибутной с единственным синтезируемым атрибутом. Но главное принципиальное отличие заключается в том, что функция f_r не является атрибутом ни нетерминала a_j, ни цепочки φ_s. Функция f_r выражает сущность правила $A_j \to \varphi_s\{f_r\}$. Наследуемые атрибуты в функциональной грамматике отсутствуют [93].

Для перехода от обычной контекстно-свободной грамматики $G_K = \{V_{TK},\ V_{NK},\ P_K,\ S_K\}$ к функциональной контекстно-свободной грамматике $G_F = \{V_T,\ V_N,\ P,\ F,\ F_0,\ V_0,\ S\}$ необходимо из всего алфавита терминальных символов V_{TK} выделить базисные терминальные символы $V_0 \subset V_{TK}$, обозначающие базисные функции $f_0 \in F_0$ рассматриваемой системы. Выбор базисных символов зависит от назначения функциональной грамматики, но часто ими являются наиболее распространенные символы. В результате такого базисного приведения алфавит терминальных символов V_{TK} сократиться до множества V_T, причем $V_{TK} \equiv (V_T \cup V_0)$. Также будут упрощены правые части продукций $A_j \to \psi_s$. Для сохранения семантики языка необходимо каждой упрощенной продукции $A_j \to \varphi_s$ поставить в соответствие функцию f_i, аргументами которой будут являться все символы последовательности φ. Тогда правила множества P будут иметь вид: $A_j \to \varphi_s\{f_r\}$. При этом каждая функция $f_r \in F$ должна быть представлена как суперпозиция базисных функций f_{0i}.

Для полного вывода в функциональной грамматике, также, как и для полного вывода в обычной контекстно-свободной грамматике, можно построить дерево

вывода. Для всех выводов в функциональной грамматике, так же, как и в контекстно-свободной грамматике, можно построить предложенное автором дерево перебора. Деревья вывода и перебора будут дополнены функциями $f_r \in F$.

Каждое разветвление (группа ветвей) дерева вывода функциональной грамматики G_F (рисунок 11) обозначается функцией f_r из F, каждый узел, как и обычно, помечается одним символом из множества $(V_T \cup V_N)$, при этом каждая вершина дерева – символом из V_T, а корень – начальным нетерминальным символом $S \in V_N$.Если узел дерева помечен символом $A_j \in V_N$, идущие от него ветви – функцией $f_r \in F$, а следующие узлы этих ветвей – символами $a_1, a_2, \ldots, a_n$, где каждый символ $a_i \in (V_T \cup V_N)$, то $A_j \to a_1 a_2 \ldots a_n \{f_r\}$ – правило вывода в функциональной грамматике G_F.

Каждая ветвь дерева перебора функциональной грамматики G_F (рисунок 12) обозначается функцией f_r из F, каждый узел, помечается цепочкой символом из множества $(V_T \cup V_N)^+$, при этом каждая вершина дерева – цепочкой из V_T^+, а корень – начальным нетерминальным символом $S \in V_N$. Если узел дерева перебора помечен цепочкой символов $\xi_1 A_j \xi_2$, где $A_j \in V_N$, $\xi_1, \xi_2 \in (V_T \cup V_N)^*$, а его непосредственные потомки – цепочками символов $\xi_1 \varphi_{t+1} \xi_2, \xi_1 \varphi_{t+2} \xi_2, \ldots, \xi_1 \varphi_{t+v} \xi_2$, $0 \le t \le m-1$, $1 \le v \le m-t$, где каждая цепочка $\varphi_{s1} \in (V_T \cup V_N)^+$, $s1 = (t+1), \ldots, (t+v)$, при этом ветви, соединяющие узел с его потомками, обозначены функциями $f_{r(1)}, f_{r(2)}, \ldots, f_{r(v)}$, то $A_j \to \varphi_{t+1}\{f_{r(1)}\} \mid \varphi_{t+2}\{f_{r(2)}\} \mid \ldots \mid \varphi_{t+v}\{f_{r(v)}\}$ – правило вывода в грамматике G_F.

Так как полный вывод в любой грамматике – это путь от начального нетерминала до правильной цепочки символов, то в функциональной грамматике этот вывод будет соответствовать последовательно-параллельному вызову функций вывода, т.е. суперпозиции всех функций, расположенных на ветвях дерева перебора от начального нетерминала до правильной цепочки.

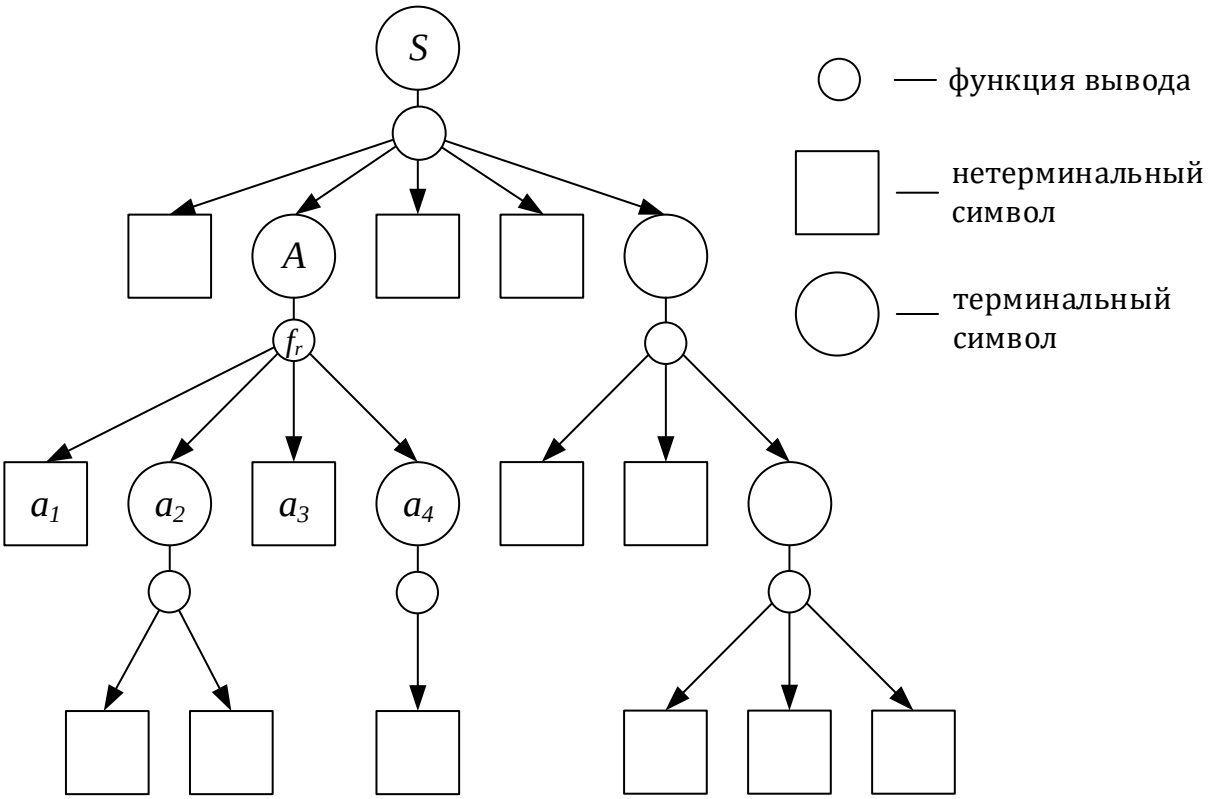

Рисунок 11 – Дерево вывода в функциональной грамматике

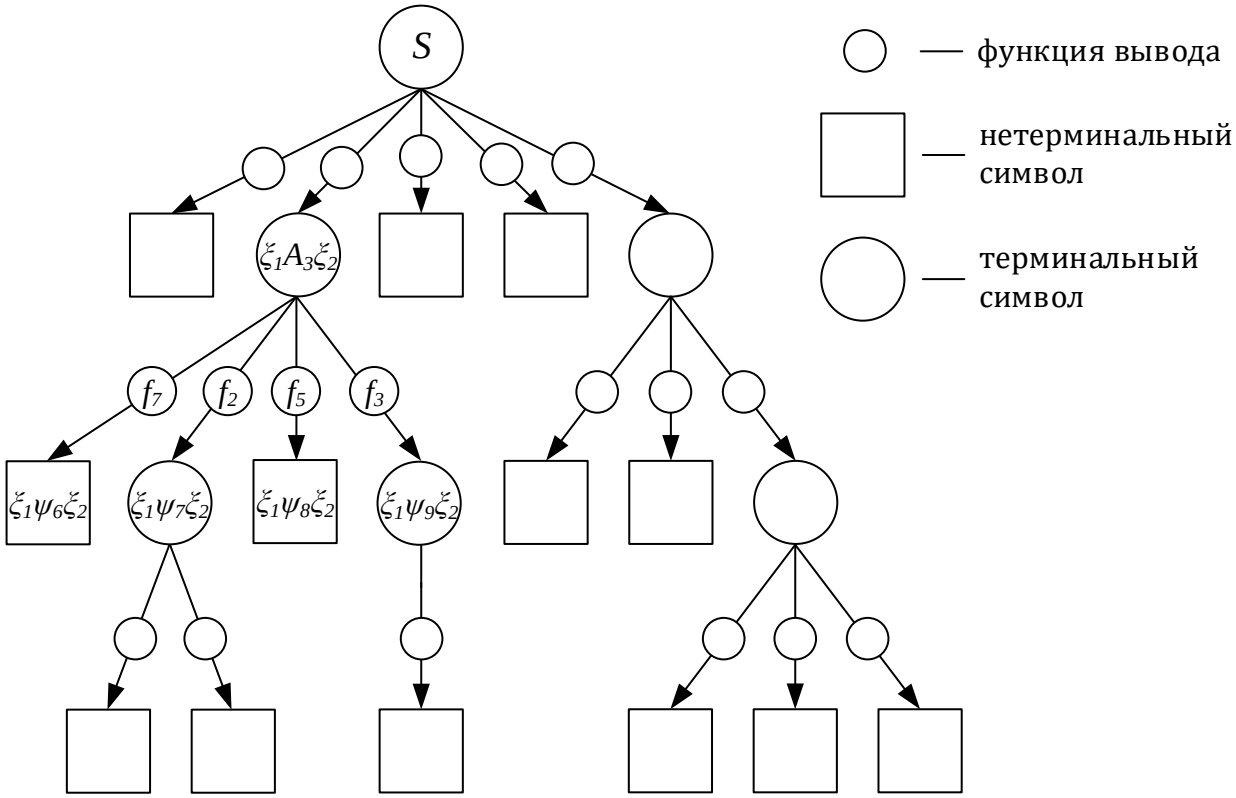

Рисунок 12 – Дерево перебора функциональной грамматики

Покажем преобразование контекстно-свободной грамматики в функциональную грамматику на примере грамматики $G1_K$, рассмотренной в 2.1. В качестве базиса новой грамматики выделим из множества $V1_{TK}$ подмножество $V1_0 = \{\alpha,\ \beta,\ \gamma,\ \delta\}$, множество терминальных символов сократится до множества $V1_T = \{a,\ b,\ c,\ d,\ e\}$. На основе базисных терминалов построим систему базисных функций $F1_0 = \{f_{01},\ f_{02},\ f_{03},\ f_{04}\}$. Далее, опираясь на систему правил $P1_K$, на основе системы базисных функций $F1_0$ построим систему функций вывода $F1 = \{f_1,\ f_2,\ f_3,\ f_4\}$.

Таким образом, схема функциональной грамматики будет иметь вид:

$$G1_F = \{V1_T,\ V1_N,\ V1_0,\ P1,\ F1,\ F1_0,\ S1\};$$

$$V1_T = \{a,\ b,\ c,\ d,\ e\}; \qquad V1_N = \{S,\ A,\ B,\ C,\ D,\ E\};$$

$$V1_0 = \{\alpha,\ \beta,\ \gamma,\ \delta\}; \qquad S1 = S;$$

$$F1_0 = \begin{Bmatrix} f_{01} = (x,y) : x\alpha y; \\ f_{02} = (x,y) : x\beta y; \\ f_{03} = (x) : \gamma x; \\ f_{04} = (x) : x\delta. \end{Bmatrix}; \qquad F1 = \begin{Bmatrix} f_1 = (x,y,z) : x\alpha y\beta z = f0_2(f0_1(x,y),z); \\ f_2 = (x,y,z) : x\beta y\alpha z = f0_1(f0_2(x,y),z); \\ f_3 = (x,y,z) : \gamma x\beta y\alpha z = f0_1\left(f0_2\left(f0_3(x),\ y\right),z\right); \\ f_4 = (x,y,z) : x\alpha y\beta z\delta = f0_4\left(f0_2\left(f0_1(x,y),z\right)\right); \\ f_5 = (x,y) : \gamma x\alpha y = f0_1\left(f0_3(x),y\right); \\ f_6 = (x,y) : x\beta y\delta = f0_4\left(f0_2(x,y)\right). \end{Bmatrix};$$

$$P1 = \begin{Bmatrix} S \to AaB\ \{f_1\}; & C \to Eb\ \{f_5\}; \\ A \to bCc\ \{f_2\} \mid deD\ \{f_3\}; & D \to cE\ \{f_6\}; \\ B \to abc\ \{f_3\} \mid dea\ \{f_4\}; & E \to de\ \{f_5\} \end{Bmatrix}.$$

Обратим внимание, что функции $f_1,\ f_2,\ \ldots,\ f_6$ выражаются через суперпозиции базисных функций $f_{01},\ f_{02},\ f_{03},\ f_{04}$.

Построим дерево перебора для полученной функциональной грамматики $G1_F$ (рисунок 13). Ветви данного дерева содержат функции, соответствующие применяемым продукциям. Так как дерево содержит четыре полных вывода, то в грамматике $G1_F$ будет существовать четыре результата, выражаемых суперпозициями функций.

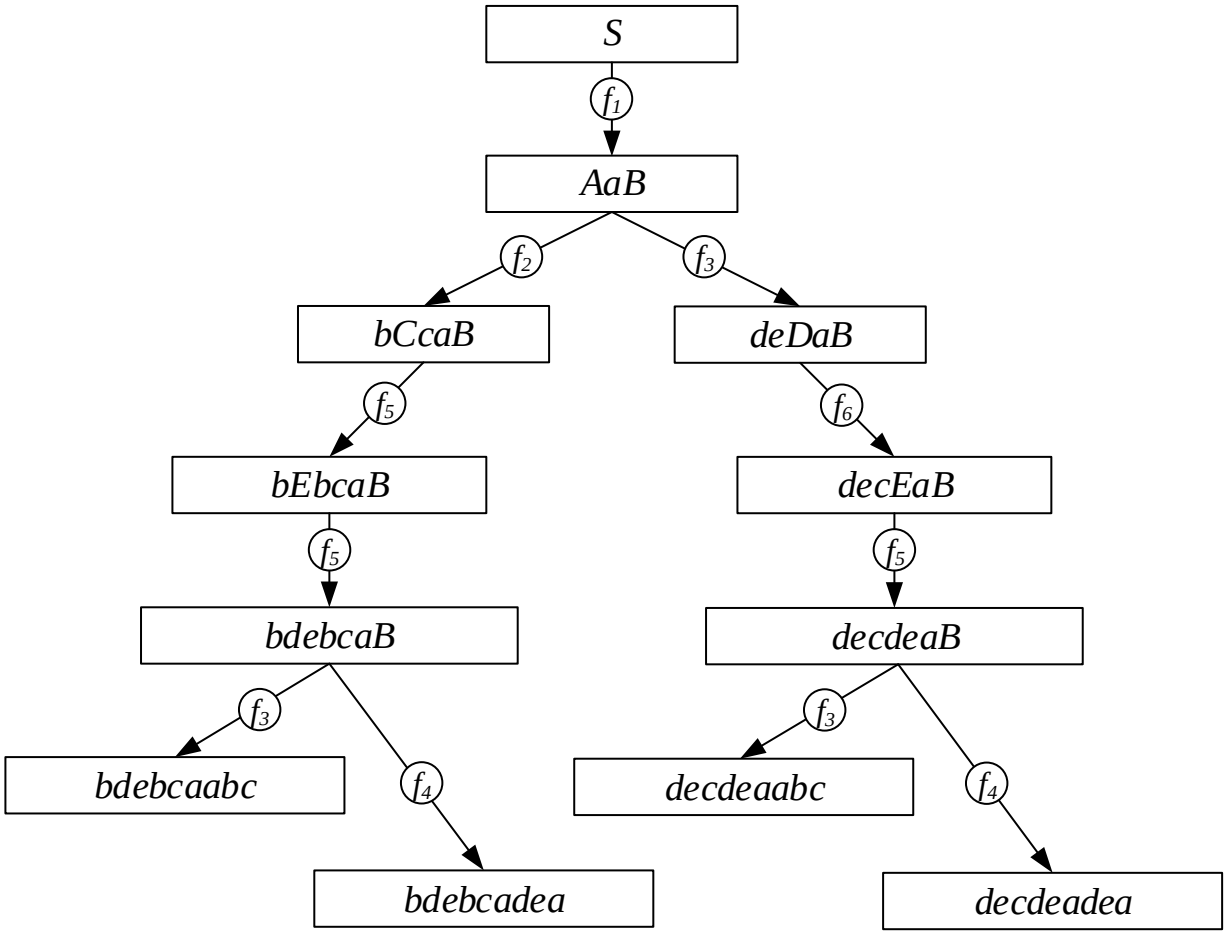

Рисунок 13 – Дерево перебора функциональной грамматики *G2*

Анализ дерева перебора дает следующие суперпозиции:

$$\sigma_1 = f_1(f_2(b, f_5(f_5(d,e),b),c),a,f_3(a,b,c));$$
$$\sigma_2 = f_1(f_2(b, f_5(f_5(d,e),b),c),a,f_4(d,e,a));$$
$$\sigma_3 = f_1(f_3(d,e,f_6(c,f_5(d,e))),a,f_3(a,b,c));$$
$$\sigma_4 = f_1(f_3(d,e,f_6(c,f_5(d,e))),a,f_4(d,e,a)).$$

На основе сравнения схем и деревьев грамматик, можно сделать вывод, что функциональные грамматики имеют ряд особенностей по сравнению с обычными контекстно-свободными грамматиками. Во-первых, схема функциональной грамматики имеет более структурированный вид. Во-вторых, цепочки порожденного языка являются более короткими ввиду сокращенного алфавита терминальных символов. В-третьих, каждому выводу в функциональной грамматике соответствует суперпозиция функций, позволяющая проследить его логику.

2.3 Представление знаний в виде неполной функциональной грамматики

Одной из важных проблем логико-математического моделирования является представление знаний. Классические способы представления знаний и принципы построения интеллектуальных систем описаны в [1, 18, 22, 62, 66-68, 76, 78, 80, 86, 90, 91, 104].

Наиболее распространенными в настоящее время формами представления знаний являются семантические сети и нейронные сети. При этом для логико-математического моделирования, в основном используются семантические сети, так как именно они описываются атрибутными грамматиками.

При использовании обычных контекстно-свободных грамматик способом представления знаний является продукционная система, тогда при использовании функциональных грамматик – модернизированная продукционная система. При этом в обоих случаях имеет место неполные грамматики, в которых не разграничены терминальные и нетерминальные символы и не указана аксиома. Дадим определение неполным контекстно-свободной и функциональной грамматикам.

Определение 2. Неполная контекстно-свободная грамматика – это совокупность:

$$G'_K = \{V_K,\ P_K\},$$

состоящая из множества (алфавита) символов:

$$V_K = \{c_i\}, \quad i = 1,\ 2,\ \ldots,\ n$$

и множества правил:

$$P_K = \{P_{Kj}\}, \quad j = 1,\ 2,\ \ldots,\ m$$

вида:

$$c_i \rightarrow \psi_j,$$

где ψ_j – произвольная последовательность символов алфавита, т.е. $\psi_j \in V_K{}^+$,

$V_K{}^+$ – множество ненулевых последовательностей символов алфавита V_K.

Определение 3. Неполная функциональная контекстно-свободная грамматика – это совокупность:

$$G'_F = \{V,\ P,\ F,\ F_0,\ V_0\},$$

состоящая из множества (алфавита) базисных терминальных символов:

$$V_0 = \{b_i\},\quad i = 1,\ 2,\ \ldots,\ n_0;$$

объединенного множества (алфавита) символов:

$$V = \{a_j\},\quad j = 1,\ 2,\ \ldots,\ n_1;$$

множества базисных унарных и бинарных функций:

$$F_0 = \{f_{0i}\},\quad i = 1,\ 2,\ \ldots,\ n_0$$

вида:

$$f_{0i} = f(x_1,\ x_2) = x_1 b_i x_2,\quad f_{0i} = f(x_1) = b_i x_1 \text{ или } f_{0i} = f(x_1) = x_1 b_i;$$

множества общих функций:

$$F = \{f_r\},\quad r = 1,\ 2,\ \ldots,\ k$$

вида:

$$f_r = f(x_1,\ x_2,\ \ldots,\ x_p) = \sigma(x_1,\ x_2,\ \ldots,\ x_p,\ f_{01},\ f_{02},\ldots,\ f_{0q});$$

множества правил:

$$P = \{P_s\},\quad s = 1,\ 2,\ \ldots,\ m$$

вида:

$$a_j \rightarrow \varphi_s\ \{f_r\},$$

где $\varphi_s \in \boldsymbol{V}^+$ – произвольная последовательность символов объединенного алфавита, являющаяся совокупностью фактических параметров аргументов $x_1,\ x_2,\ \ldots,\ x_p$ функции f_r.

На основании определений 2 и 3 введем два предположения [50].

Предположение 1. Определенная теория динамических систем математически может быть описана в виде неполной контекстно-свободной грамматики $G'_K = \{V_K,\ P_K\}$. При этом символы грамматики c_i обозначают понятия теории, а правила грамматики P_{Kj} – отношения между понятиями в виде формул.

Предположение 2. Математическое описание теории динамических систем в виде неполной контекстно-свободной грамматики $G'_K = \{V_K,\ P_K\}$. может быть преобразовано в неполную функциональную контекстно-свободную грамматику $G'_F = \{V,\ P,\ F,\ F_0,\ V_0\}$ за счет выделения из множества V_K подмножества V_0, такого, что $V_0 \cup V \equiv V_K$, и его элементам соответствуют базисные функции f_{0i}, через которые выражаются функции f_r, описывающие предметную логику правил P_s.

Подтвердим полученные положения на примере создания базы знаний по механике прямолинейного равноускоренного движения тела [92]. При этом сначала создадим базу знаний на основе неполной контекстно-свободной грамматики $G2'_K$ (предположение 1), а потом переведем её в базу знаний на основе неполной функциональной грамматики (предположение 2) $G2'_F$.

Внесём в базу знаний 14 физических величин: расстояние; время; начальная скорость; конечная скорость; ускорение; масса; начальная кинетическая энергия; конечная кинетическая энергия; сила тяги; сила сопротивления; работа силы тяги; работа силы сопротивления; коэффициент сопротивления; ускорение свободного падения. Для описания отношений между понадобятся 5 математических операций: сложение, вычитание, умножение, деление и извлечение квадратного корня. Таким образом, объединённый алфавит грамматики $V2_K$ будет состоять из 20 символов, приведённых в таблице 4.

Таблица 4 – Символы грамматики $G2_K$

Символ грамматики	Смысл в базе знаний	Обозначение в теории
<Dist>	Расстояние	S
<Time>	Время	t
<Spd1>	Начальная скорость	V_0
<Spd2>	Конечная скорость	V
<Accl>	Ускорение	a

Продолжение таблицы 4

Символ грамматики	Смысл в базе знаний	Обозначение в теории
<Mass>	Масса	m
<KinEn1>	Начальная кинетическая энергия	$E_{к0}$
<KinEn2>	Конечная кинетическая энергия	$E_к$
<FrPull>	Сила тяги	$F_т$
<FrFric>	Сила сопротивления	$F_с$
<OpPull>	Работа силы тяги	$A_т$
<OpFric>	Работа силы сопротивления	$A_с$
<CfFric>	Коэффициент сопротивления	μ
<Cfg>	Ускорение свободного падения	g
+	Сложение	+
–	Вычитание	–
*	Умножение	·
/	Деление	$\frac{}{}$
<КОРЕНЬ>	Извлечение квадратного корня	$\sqrt{\ }$
2	Число «два»	2

Между временем, расстоянием, ускорением, начальной и конечной скоростью в кинематике существуют следующие соотношения:

$$V = V_0 + a \cdot t; \qquad S = V_0 \cdot t + \frac{a \cdot t^2}{2}; \qquad S = \frac{V^2 - V_0^2}{2 \cdot a}.$$

Из них можно породить 12 зависимостей:

$$V = V_0 + a \cdot t; \qquad V_0 = \sqrt{V^2 - 2 \cdot a \cdot S}; \quad t = \frac{\sqrt{V_0^2 + 2 \cdot a \cdot S} - V_0}{a};$$

$$V = \sqrt{2 \cdot a \cdot S + V_0^2}; \quad S = V_0 \cdot t + \frac{a \cdot t^2}{2}; \qquad a = \frac{V - V_0}{t};$$

$$V_0 = V - a \cdot t; \qquad S = \frac{V^2 - V_0^2}{2 \cdot a}; \qquad a = \frac{2 \cdot (S - V_0 \cdot t)}{t^2};$$

$$V_0 = \frac{S}{t} - \frac{a \cdot t}{2}; \qquad t = \frac{V - V_0}{a}; \qquad a = \frac{V^2 - V_0^2}{2 \cdot S}.$$

Также существуют 7 соотношений из динамики, связывающие перечисленные выше величины с массой, силой тяги и сопротивления, работами этих сил, коэффициентами сопротивления и ускорения свободного падения, начальной и конечной кинетическими энергиями:

$$m \cdot a = F_m - F_c; \qquad A_c = F_c \cdot S; \qquad E_{к0} = \frac{m \cdot V_0^2}{2};$$

$$F_c = g \cdot \mu \cdot m; \qquad A_m = F_m \cdot S; \qquad E_к = \frac{m \cdot V^2}{2}.$$

$$A_m - A_c = E_к - E_{к0};$$

Из них можно вывести 23 зависимости:

$$V = \sqrt{\frac{2 \cdot E_к}{m}}; \qquad S = \frac{A_m}{F_m}; \qquad a = \frac{F_m - F_c}{m};$$

$$V_0 = \sqrt{\frac{2 \cdot E_{к0}}{m}}; \qquad S = \frac{A_c}{F_c}; \qquad m = \frac{F_m - F_c}{a};$$

$$m = \frac{F_c}{g \cdot \mu}; \qquad F_c = F_m - m \cdot a; \qquad E_{к0} = \frac{m \cdot V_0^2}{2};$$

$$F_c = g \cdot \mu \cdot m;$$

$$m = \sqrt{\frac{2 \cdot E_к}{V^2}}; \qquad F_c = \frac{A_c}{S}; \qquad E_{к0} = E_к - A_m + A_c;$$

$$m = \sqrt{\frac{2 \cdot E_{к0}}{V_0^2}}; \qquad \mu = \frac{F_c}{g \cdot m}; \qquad A_m = F_m \cdot S;$$

$$A_m = E_к - E_{к0} + A_c;$$

$$F_m = F_c + m \cdot a; \qquad E_к = \frac{m \cdot V^2}{2}; \qquad A_c = F_c \cdot S;$$

$$F_m = \frac{A_m}{S}; \qquad E_к = A_m - A_c + E_{к0}; \qquad A_c = A_m - E_к + E_{к0}.$$

На основании полученных 35 зависимостей составим 35 продукций грамматики $G2'_K$, образующих систему правил $P2_K$:

$P2_K$ = { <Spd2> → <Spd1> + <Accl>*<Time> ;

<Spd2> → КОРЕНЬ [2*<Accl>*<Dist> + <Spd1>*<Spd1>] ;

<Spd1> → <Spd2> – <Accl>*<Time> ;

<Spd1> → <Dist>/<Time> – <Accl>*<Time>/2 ;

<Spd1> → КОРЕНЬ [<Spd2>*<Spd2> – 2*<Accl>*<Dist>] ;

<Dist> → <Spd1>*<Time> + <Accl>*<Time>>*<Time> / 2 ;

<Dist> → [<Spd2>*<Spd2> – <Spd1>*<Spd1>] / [2*<Accl>] ;

<Time> → [<Spd2> – <Spd1>] / <Accl> ;

<Time> → [КОРЕНЬ [<Spd1>*<Spd2> + 2*<Accl>*<Dist>] – <Spd1>] / <Accl> ;

<Accl> → [<Spd2> – <Spd1>] / <Time> ;

<Accl> → 2* [<Dist> – <Spd1>*<Time>] / [<Time>^2] ;

<Accl> → [<Spd2>*<Spd2> – <Spd1>*<Spd1>] / [2*<Dist>] ;

<Spd2> → КОРЕНЬ [2*<KinEn2> / <Mass>] ;

<Spd1> → КОРЕНЬ [2*<KinEn1> / <Mass>] ;

<Dist> → <OpPull> / <FrPull> ;

<Dist> → <OpFric> / <FrFric> ;

<Accl> → [<FrPull> – <FrFric>] / <Mass> ;

<Mass> → [<FrPull> – <FrFric>] / <Accl> ;

<Mass> → <FrFric> / <Cfg> / <CfFric> ;

<Mass> → 2*<KinEn2> / [<Spd2>*<Spd2>] ;

<Mass> → 2*<KinEn1> / [<Spd1>*<Spd1>] ;

<FrPull> → <FrFric> + <Mass>*<Accl> ;

<FrPull> → <OpPull> / <Dist> ;

<FrFric> → <FrPull> – <Mass>*<Accl> ;

<FrFric> → <Cfg>*<CfFric>*<Mass> ;

<FrFric> → <OpFric> / <Dist> ;

<CfFric> → <FrFric> / <Cfg> / < Mass> ;

<KinEn2> → <Mass>*<Spd2>*<Spd2> / 2 ;

<KinEn2> → <OpPull> – <OpFric> + <KinEn1> ;

<KinEn1> → <Mass>*<Spd1>*<Spd1> / 2 ;

<KinEn1> → <KinEn2> – <OpPull> + <OpFric> ;

<OpPull> → <FrPull>*<Dist> ;

<OpPull> → <KinEn2> – <KinEn1> + <OpFric> ;

<OpFric> → <FrFric>*<Dist> ;

<OpFric> → <OpPull> – <KinEn2> + <KinEn1>. }

Таким образом, были получена неполная контекстно-свободная грамматика $G2'_K$, состоящая из обобщенного алфавита $V2_K$ и системы правил $P2_K$.

Для перехода к неполной функциональной грамматике $G2'_F$ выбираем в качестве базисных терминальных символов те символы объединённого алфавита, которые выражают элементарные отношения между понятиями: <+>, <–>, <*>, </>, <КОРЕНЬ>, а также <2>.

Тогда объединённый алфавит символов будет иметь вид:

$$V2 = \left\{ \begin{array}{l} \text{<Dist>, <Time>, <Spd1>, <Spd2>, <Accl>,} \\ \text{<Mass>, <KinEn1>, <KinEn2 >, <FrPull>,} \\ \text{<FrFric>, <OpPull>, <OpFric>, <CfFric>, <Cfg>} \end{array} \right\};$$

а алфавит базисных символов:

$$V2_0 = \{\text{<+>, <–>, <*>, </>, <КОРЕНЬ>, <2>}\}.$$

Вместо базисных символов вводим базисные функции:

$F2_0$ = { $f0_1 = (x, y)$: $x + y$;

$f0_2 = (x, y)$: $x – y$;

$f0_3 = (x, y)$: $x * y$;

$f0_4 = (x, y)$: x / y ;

$f0_5 = (x)$: КОРЕНЬ(x) :

$f0_6 = ( )$: 2 }.

После анализа всех продукций системы $P2_K$ можно прийти к выводу, что между описанными понятиями механики существует 18 всевозможных отношений, которые можно выразить через элементарные отношения базисных функций:

$F2$ = { $f_1 = (x, y, z)$: $f0_1(x, f0_3(y, z))$;

$f_2 = (x, y, z)$: $f0_5(f0_1(f0_3(f0_3(x, y), f0_6()), f0_3(z, z)))$;

$f_3 = (x, y, z)$: $f0_2(x, f0_3(y, z))$;

$f_4 = (x, y, z)$: $f0_2(f0_4(x, y), f0_4(f0_3(f0_3(y, y), z), f0_6()))$;

$f_5 = (x, y, z)$: $f0_5(f0_2(f0_3(x, x), f0_3(\mathrm{f0}_3(y, z), f0_6())))$;

$f_6 = (x, y, z)$: $f0_1(f0_3(x, y), f0_4(f0_3(f0_3(y, y), z), f0_6()))$;

$f_7 = (x, y, z)$: $f0_4(f0_2(f0_3(x, x), f0_3(y, y)), f0_3(f0_6(), z))$;

$f_8 = (x, y, z)$: $f0_4(f0_2(x, y), z)$;

$f_9 = (x, y, z)$: $f0_4(f0_2(f0_5(f0_1(f0_3(x, x), f0_3(f0_3(y, z), f0_6()))), x), y)$;

$f_{10} = (x, y, z)$: $f0_4(f0_3(f0_6(), f0_2(x, f0_3(y, z))), f0_3(z, z))$;

$f_{11} = (x, y, z)$: $f0_5(f0_4(f0_3(f0_6(), x), y))$;

$f_{12} = (x, y)$: $f0_4(x, y)$;

$f_{13} = (x, y, z)$: $f0_4(f0_3(x, y), z)$;

$f_{14} = (x, y)$: $f0_4(f0_3\,(f0_6(), x), f0_5(y))$;

$f_{15} = (x, y, z)$: $f0_3(f0_3(x, y), z)$;

$f_{16} = (x, y)$: $f0_4(f0_3(x, f0_3(y, y)), f0_6())$;

$f_{17} = (x, y, z)$: $f0_1(f0_2(x, y), z)$;

$f_{18} = (x, y)$: $f0_3(x, y)$ }.

Дополним систему правил введенными функциями:

P2 = { <Spd2> → <Spd1> <Accl> <Time> {f_1};

<Spd2> → <Accl> <Dist> <Spd1> {f_2};

<Spd2> → <KinEn2> <Mass> {f_{11}};

<Spd1> → <Spd2> <Accl> <Time> {f_3};

<Spd1> → <Dist> <Time> <Accl> {f_4};

<Spd1> → <Spd2> <Accl> <Dist>] {f_5};

<Spd1> → <KinEn1> <Mass> {f_{11}};

<Dist> → <Spd1> <Time> <Accl> <Time> {f_6};

<Dist> → <Spd2> <Spd1> <Accl> {f_7};

<Dist> → <OpPull> <FrPull> {f_{12}};

<Dist> → <OpFric> <FrFric> {f_{12}};

<Time> → <Spd2> <Spd1> <Accl> {f_8};

<Time> → <Spd1> <Accl> <Dist> {f_9};

<Accl> → <Spd2> <Spd1> <Time> {f_8};

<Accl> → <Dist> <Spd1> <Time> {f_{10}};

<Accl> → <Spd2> <Spd1> <Dist> {f_7}.

<Accl> → <FrPull> <FrFric> <Mass> {f_8};

<Mass> → <FrPull> <FrFric> <Accl> {f_8};

<Mass> → <FrFric> <CfFric> <Cfg> {f_{13}};

<Mass> → <KinEn2> <Spd2> {f_{14}};

<Mass> → <KinEn1> <Spd1> {f_{14}};

<FrPull> → <FrFric> <Mass> <Accl> {f_1};

<FrPull> → <OpPull> <Dist> {f_{12}};

<FrFric> → <FrPull> <Mass> <Accl> {f_3};

<FrFric> → <Cfg> <CfFric > <Mass> {f_{15}};

<FrFric> → <OpFric> <Dist> {f_{12}};

<CfFric> → <FrFric> < Mass> <Cfg>{f_{13}};

<KinEn2> → <Mass> <Spd2> {f_{16}};

<KinEn2> → <OpPull> <OpFric> <KinEn1> {f_{17}};

<KinEn1> → <Mass> <Spd1> {f_{16}};

<KinEn1> → <KinEn2> <OpPull> <OpFric> {f_{17}};

<OpPull> → <FrPull> <Dist> {f_{18}};

<OpPull> → <KinEn2> <KinEn1> <OpFric> {f_{17}};

<OpFric> → <FrFric> <Dist> {f_{18}};

<OpFric> → <OpPull> <KinEn2> <KinEn1> {f_{17}} }.

Таким образом, была построена база знаний законов механики в виде неполной функциональной грамматики $G2'_F = \{V2,\ P2,\ F2,\ F2_0,\ V2_0\}$.

Сформулируем основные этапы составления базы знаний предметной области в виде неполной функциональной грамматики [7, 38, 46]. При этом обойдем построение неполной контекстно-свободной грамматики, считая, что предположение 1 выполняется автоматически за счет существования формульного описания рассматриваемой предметной области.

Этапы составления базы знаний:

1) определение границ рассматриваемой теории;

2) выявление всех понятий теории в рамках выбранных границ и составление алфавита символов *V*, а также определение синтаксической структуры каждого символа;

3) выявление всех законов, связывающих выбранные объекты предметной области, и составление набора правил *P*;

4) описание всех функциональных зависимостей между понятиями теории, согласно правилам *P*, т.е. определение множеств функций *F* и базисных функций F_0, а также множества базисных символов V_0.

Опишем логико-математический метод представления знаний в виде неполной функциональной грамматики на языке чистого лямбда исчисления [8, 54, 74].

Для этого введем логические комбинаторы:

$Tr = \lambda x.\lambda y.x$;

$Fl = \lambda x.\lambda y.y$;

$If = \lambda p.\lambda x.\lambda y.p\ x\ y$.

Комбинаторы работы со списками:

$Nu = \lambda t.t(\lambda x.\lambda y.Fl)$;

$Nl = \lambda x.Tr$;

$Cs = \lambda x.\lambda y.\lambda s.s\ x\ y$;

$Cr = \lambda t.t.Tr$;

$Cd = \lambda t.t.Fl$;

$Crr = \lambda t.t.Cr\ (Cr\ t)$;

$Crdr = \lambda t.t.Cr\ (Cd\ (Cr\ t))$;

$Crdr = \lambda t.t.Cr\ (Cd\ (Cr\ t))$;

$Crddr = \lambda t.t.Cr\ (Cd\ (Cd\ (Cr\ t)))$;

$Crdddr = \lambda t.t.Cr\ (Cd\ (Cd\ (Cd\ (Cr\ t))))$;

$Crddddr = \lambda t.t.Cr\ (Cd\ (Cd\ (Cd\ (Cd\ (Cr\ t)))))$;

$Crdd = \lambda t.t.Cr\ (Cd\ (Cd\ t))$;

$Crddd = \lambda t.t.Cr\ (Cd\ (Cd\ (Cd\ t)))$;

$Crdddd = \lambda t.t.Cr\ (Cd\ (Cd\ (Cd\ (Cd\ t))))$;

$Crddddd = \lambda t.t.Cr\ (Cd\ (Cd\ (Cd\ (Cd\ (Cd\ t)))))$.

Комбинатор неподвижной точки:

$Y = \lambda h.(\lambda x.h\ (x\ x))\ (\lambda x.h\ (x\ x))$.

Тогда функцию построения неполной функциональной грамматики можно представить в виде:

$G' = (\lambda x.\lambda y.\lambda z.\lambda p.\lambda q.Cs\ x\ (Cs\ y\ (Cs\ z\ (Cs\ p\ (Cs\ q\ Nl)))))\ V\ P\ F\ F_0\ V_0$.

База знаний представляет собой неполную функциональную грамматику, поэтому, на основе β- и δ-редукций, можно записать:

$K = (\lambda x.x)\ G' \ \rightarrow_{\beta}\ G' \ \rightarrow_{\delta}$

$(\lambda x.\lambda y.\lambda z.\lambda p.\lambda q.Cs\ x\ (Cs\ y\ (Cs\ z\ (Cs\ p\ (Cs\ q\ Nl))))))\ V_0\ V\ F_0\ F\ P \quad \rightarrow_{\beta^*}$

$Cs\ V_0\ (Cs\ V\ (Cs\ F_0\ (Cs\ F\ (Cs\ P\ Nl)))) \ \rightarrow_{\delta\beta^*}$

$\lambda s.s\ V_0\ (\lambda s.s\ V\ (\lambda s.s\ F_0\ (\lambda s.s\ F\ (\lambda s.s\ P\ (\lambda s.s\ Nl))))) .$

$$K = \lambda s.s\ V_0\ (\lambda s.s\ V\ (\lambda s.s\ F_0\ (\lambda s.s\ F\ (\lambda s.s\ P\ (\lambda s.s\ Nl)))))$$

2.4 Решение задачи моделирования на основе полного вывода в функциональной грамматике

Для того чтобы решить задачу моделирования в рамках теории, представленной в виде неполной функциональной грамматики, необходимо довести эту грамматику до полного вида [39, 40, 45]. Это происходит на стадии задания условия задачи, т.е. определения известных и целевых знаний об объекте (рабочие знания). Известные понятия задаются как терминальные символы, искомое понятие – как аксиома, а оставшиеся понятия – как нетерминальные символы. Назначение всех элементов полной и неполной функциональных грамматик приведены в таблице 5.

Каждой новой задаче будет соответствовать новая функциональная грамматика, т.е. теория моделируется неполной функциональной грамматикой G'_F, а задачи в рамках данной теории – набором полных функциональных грамматик $G1'_F, G2'_F, \ldots, Gs'_F$:

$$G'_F = (V,\ P,\ F,\ F_0,\ V_0) \ \Rightarrow$$

$$\begin{cases} G1'_F = (V1_T,\ V1_N,\ P1,\ F1,\ F1_0,\ V1_0,\ S1),\ V \equiv (V1_T \cup V1_N),\ S1 \in V1_N; \\ G2'_F = (V2_T,\ V2_N,\ P2,\ F2,\ F2_0,\ V2_0,\ S2),\ V \equiv (V2_T \cup V2_N),\ S2 \in V2_N; \\ - \\ Gs'_F = (Vs_T,\ Vs_N,\ Ps,\ Fs,\ Fs_0,\ Vs_0,\ Ss),\ V \equiv (Vn_T \cup Vn_N),\ Ss \in Vn_N. \end{cases}$$

Таблица 5 – Назначение элементов функциональных грамматик

<table>
<tr><td>Элемент грамматики</td><td>Неполная функциональная грамматика</td><td>Полная функциональная грамматика</td></tr>
<tr><td>Объединенный алфавит V</td><td colspan="2">Все понятия об объекте (входят в K)</td></tr>
<tr><td>Алфавит терминальных символов V_T</td><td>–</td><td>Исходные понятия об объекте D</td></tr>
<tr><td>Алфавит нетерминальных символов V_N</td><td>–</td><td>Неизвестные понятия об объекте</td></tr>
<tr><td>Алфавит базисных символов V_{T0}</td><td colspan="2" rowspan="2">Элементарные отношения между понятиями (входят в K)</td></tr>
<tr><td>Система базисных функций F0</td></tr>
<tr><td>Система функций вывода F</td><td colspan="2">Все возможные функциональные зависимости в теории (входят в K)</td></tr>
<tr><td>Система правил вывода P</td><td colspan="2">Отношения между конкретными понятиями (входят в K)</td></tr>
<tr><td>Аксиома S</td><td>–</td><td>Целевые понятия об объекте Q</td></tr>
</table>

Таким образом, для решения каждой задачи индивидуально задаются аксиома, терминальные и нетерминальные символы. Продукции и общий алфавит постоянны в рамках предметной области. Кроме того, следует заметить, что если неизвестных величин в задаче несколько, то решение разбивается на несколько этапов, в каждом из которых заново задается аксиома. Аксиомы предыдущих этапов можно в качестве известных величин переводить в класс терминальных символов.

В рамках полной функциональной грамматики появляется возможность построить дерево перебора, рассмотренное в 2.2. Оно необходимо для поиска полного вывода, соединяющего корневой узел – аксиому – с узлом, представленным только терминальными символами. Осуществление такого вывода есть задача механизма вывода, который должен последовательно применять продукции грамматики, позволяющие заменять нетерминалы на цепочки правых частей продукций. Однако при построении дерева перебора

необходимо учитывать «правила роста дерева».

Правила роста дерева – это специальные правила, предложенные автором для ограничения разрастания дерева, что необходимо для исключения комбинаторного взрыва и выбора самого оптимального решения.

Правила роста дерева [39]:

1) Правило роста в ширину – поиск необходимо производить «в ширину», что подразумевает изучение всех узлов каждого уровня наличие правильной цепочки, и только после этого – переход на более глубокий уровень. Если узел, содержащий правильную цепочку, найден, то необходимо прекратить дальнейшее построение дерева. Данное правило позволяет выбрать из нескольких возможных правильных решений более простое. Действительно, переход на следующий уровень означает вызов новой функции и усложнение результата.

2) Правило тупиковых ветвей – необходимо отсекать узлы дерева, содержащие аксиомы или ранее заменённые в её ветви нетерминалы. Данное правило позволяет избежать зацикливания дерева и комбинаторного взрыва. Действительно, узлы вызывающие ранее заменённые символы заставляют процесс поиска возвращаться назад и не могут привести к правильной цепочке.

В результате применения каждой продукции в дереве перебора образуется новая ветвь, которой сопоставляется функция данной продукции. Поэтому найденному полному выводу будет соответствовать суперпозиция функций, расположенных в узлах, через которые проходит этот вывод. Данная суперпозиция и отражает решение задачи.

Таким образом, для решения задачи в рамках теории, представленной базой знаний, моделированной неполной функциональной грамматикой, необходимо:

1) выделить из объединённого алфавита V алфавит терминальных символов V_T, которые соответствуют понятиям, заданным в условии задачи; оставшийся алфавит считать алфавитом нетерминальных символов V_N;

2) выделить из полученного нетерминального алфавита аксиому – начальный нетерминальный символ S, соответствующий целевому понятию задачи;

3) построить дерево разбора, руководствуясь «правилами роста дерева»;

4) выдать в качестве результата первую полученную суперпозицию, содержащую в качестве аргументов только терминальные символы (правильную цепочку).

Если в задаче $s > 1$ целевых понятий, то её решение необходимо разбить на s задач, аксиомой в каждой из которых будет являться одно из целевых понятий. При этом, в алфавит терминальных символов i-ой задачи ($i = 1..s$) должны входить аксиомы с 1-ой по $i-1$-ую решённых задач.

Введем предположение о представлении задачи математического моделирования динамической системы в виде совокупности полных функциональных грамматик.

Предположение 3. Задача математического моделирования динамической системы $W = \{\boldsymbol{K},\ \boldsymbol{D},\ \boldsymbol{Q}\}$ может быть представлена в виде совокупности полных функциональных контекстно-свободных грамматик $G_j = \{Vj_T,\ Vj_N,\ P,\ F,\ F_0,\ V_0,\ Sj\},\quad j = 1,\ 2,\ \ldots,\ s;$ где s – количество выходных переменных в множестве $\boldsymbol{Q}$; за счет выделения из обобщенного алфавита V неполной функциональной контекстно-свободной грамматики $\boldsymbol{K} = G' = \{V,\ P,\ F,\ F_0,\ V_0\}$ алфавита терминальных символов Vj_T, обозначающих понятия входных переменных $\boldsymbol{D}$, и последовательного выбора из оставшегося множества нетерминальных символов Vj_N ($Vj_T \cup Vj_N \equiv V$) аксиом Sj, обозначающих понятия выходных переменных q_j.

Следствие. Если задача математического моделирования динамической системы $W = \{\boldsymbol{K},\ \boldsymbol{D},\ \boldsymbol{Q}\}$ представлена в виде совокупности полных функциональных контекстно-свободных грамматик $Gj = \{Vj_T,\ Vj_N,\ P,\ F,\ F_0,\ V_0,\ Sj\},$ и для указанной задачи существует решение, то логико-математическая модель может быть построена в виде совокупности суперпозиций функций $\sigma_j = \sigma_j(x_1,\ x_2,\ \ldots,\ x_p,\ f_{(1j)},\ f_{(2j)},\ \ldots,\ f_{(tg)}),$ где $x_1,\ x_2,\ \ldots,\ x_p \in V_T;\quad f_{(1j)},\ f_{(2j)},\ \ldots, f_{(tj)} \in F,\ j = 1,\ 2,\ \ldots,\ s;$ s – количество выходных переменных в множестве $\boldsymbol{Q}$; t – количество последовательных блоков

декомпозиции оператора вывода.

При этом функции $f_{(1j)}$, $f_{(2j)}$, ... , $f_{(tj)}$ выступают в качестве элементарных операторов последовательно-параллельной декомпозиции B_{ij}, а суперпозиция функций σ_j имеет смысл оператора вывода решения $\boldsymbol{B}$.

Доказательство. Задача математического моделирования $W = \{\boldsymbol{K},\ \boldsymbol{D},\ \boldsymbol{Q}\}$ имеет решение лишь в том случае, если цель моделирования в виде выходных переменных q_j может быть выражена через входные переменные $\boldsymbol{D}$ в виде оператора $\boldsymbol{B}$. Т.е., согласно предположению 3, задача математического моделирования имеет решение, если с помощью применения правил P_j может быть построен полный вывод:

$$S \to \varphi_{V1} \to \varphi_{V2} \to ... \to \varphi_{V(n-1)} \to \varphi_{Vn} \to \varphi_T,$$

где φ_{V1}, φ_{V2}, ... , $\varphi_{V(n-1)}$, $\varphi_{Vn} \in (V_T \cup V_N)^+$ – последовательности, содержащие как терминальные, так и нетерминальные символы;

$\varphi_T \in V_T^{\ +}$ – последовательность терминальных символов.

Каждому применению правила

$$a_j \to \varphi_{Vj}\ \{f_i\},\ j = 1,\ 2,\ ...\ ,\ n$$

соответствует функция:

$$f_i(x_1,\ x_2,\ ...\ ,\ x_{pi}),$$

где i – номер произвольной функции грамматики, $1 \le i \le k$;

p_i – количество аргументов функции f_i.

Первому непосредственному выводу $S \to \varphi_{V1}$ соответствует функция:

$$f_{i(1)}(x_1,\ x_2,\ ...\ ,\ x_{pi(1)}).$$

Второму непосредственному выводу $\varphi_{V1} \to \varphi_{V2}$ соответствует функция:

$$f_{i(2)}(x_{pi(1)+1},\ x_{pi(1)+2},\ ...\ ,\ x_{pi(1)+pi(2)}).$$

Тогда выводу $S \to \varphi_{V2}$ соответствует суперпозиция:

$$f_{i(2)}(x_{pi(1)+1},\ x_{pi(1)+2},\ ...\ , x_{pi(1)+pi(2)-1},\ f_{i(1)}(x_1,\ x_2,\ ...\ ,\ x_{pi(1)})) =$$
$$= f_{i(2)}(x_1,\ x_2,\ ...\ ,\ x_{pi(1)+pi(2)-1},\ f_{i(1)}).$$

В общем случае, выводу $S \to \varphi_{Vj}$ соответствует суперпозиция функций:

$$f_{i(j)}(x_1,\ x_2,\ \dots,\ x_{pi(1)+pi(2)+\dots+pi(j)-j+1},\ f_{i(1)},\ f_{i(2)},\ \dots,\ f_{i(j-1)}).$$

Тогда полный вывод описывается суперпозицией функций:

$$f_{i(n+1)}(x_1,\ x_2,\ \dots,\ x_{pi(1)+pi(2)+\dots+pi(n+1)-n},\ f_{i(1)},\ f_{i(2)},\ \dots,\ f_{i(n)}) =$$
$$= \sigma(x_1,\ x_2,\ \dots,\ x_{pi(1)+pi(2)+\dots+pi(n+1)-n},\ f_{i(1)},\ f_{i(2)},\ \dots,\ f_{i(n+1)}) =$$
$$= \sigma(x_1,\ x_2,\ \dots,\ x_p,\ f_{i(1)},\ f_{i(2)},\ \dots,\ f_{i(q)}).$$

где $p = p_{i(1)} + p_{i(2)} + \dots + p_{i(n+1)} - n$ – количество аргументов суперпозиции,

$q = n + 1$ – количество функций в суперпозиции.

Таким образом, если задача математического моделирования имеет решение, то в представляющей её функциональной контекстно-свободной грамматике может быть построен полный вывод:

$$S \to \varphi_{V1} \to \varphi_{V2} \to \dots \to \varphi_{V(n-1)} \to \varphi_{Vn} \to \varphi_T,$$

которому соответствует математическая модель в виде суперпозиции функций:

$$\sigma(x_1,\ x_2,\ \dots,\ x_p,\ f_{i(1)},\ f_{i(2)},\ \dots,\ f_{i(q)})$$

Следствие доказано.

Рассмотрим решение задачи в рамках на базы знаний, составленной в 2.3. Пусть в задаче известны: сила тяги, действующая на тело, пройденный им путь, его начальная и конечная кинетические энергии, а требуется найти действующую силу трения. В этом случае, роль терминальных символов будут играть: <FrPull>, <Dist>, <KinEn1>, <KinEn2>; а аксиомы – <FrFric>, т.е.:

$V2_T =$ {<Dist>, <KinEn1>, <KinEn2>, <FrPull>};

$$V2_N = \left\{ \begin{array}{l} \text{<Time>, <Spd1>, <Spd2>, <Accl>, <Mass>,} \\ \\ \text{<FrFric>, <OpPull>, <OpFric>, <CfFric>, <Cfg>} \end{array} \right\};$$

$S2 =$ <FrFric>.

В соответствии с правилами роста, дерево перебора будет иметь вид, представленный на рисунке 14. Из рисунка видно, что подъём по дереву (спуск по рисунку), согласно первому правилу роста, осуществляется только после анализа

всех узлов каждого нового уровня.

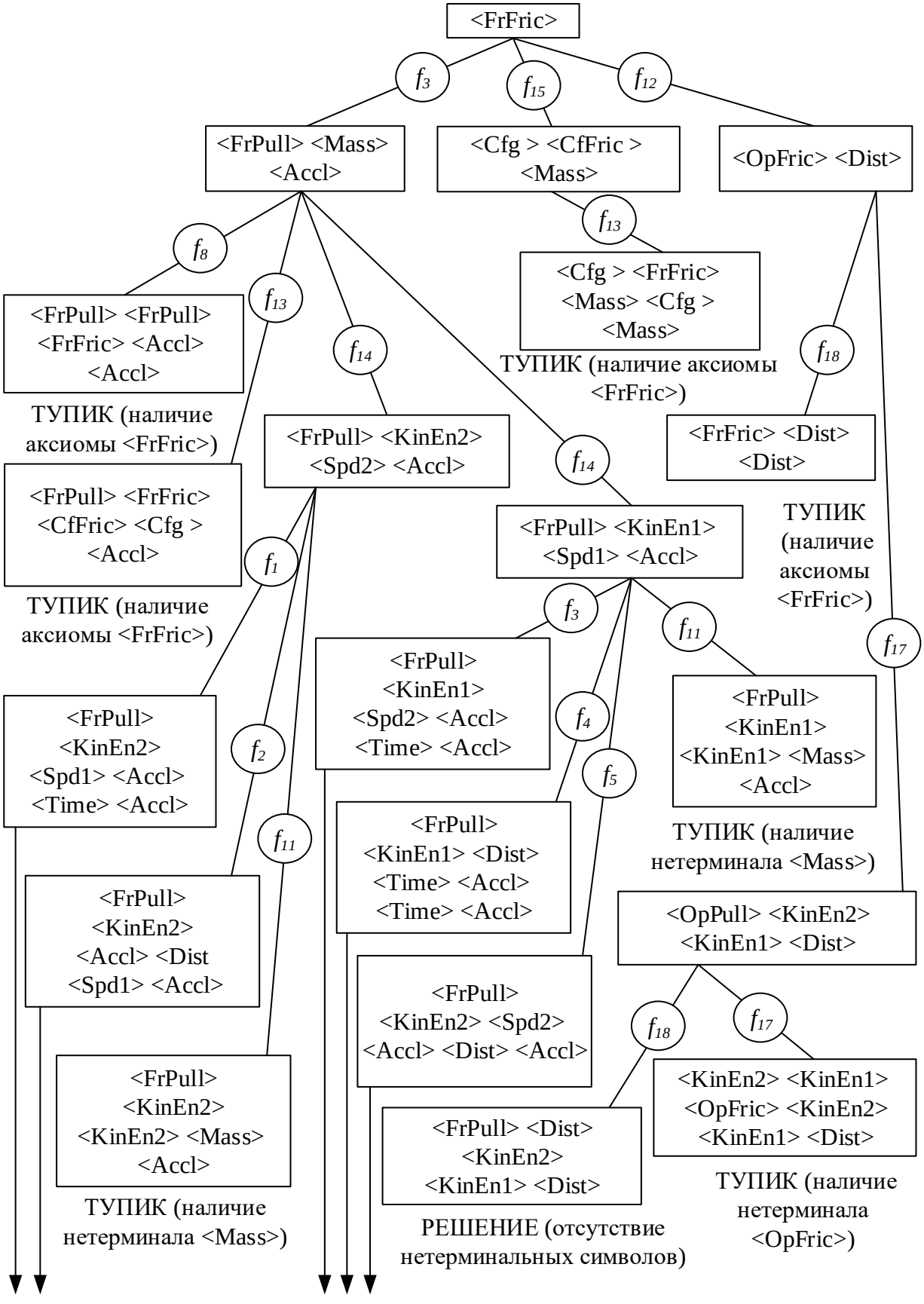

Рисунок 14 – Дерево перебора для решения задачи в рамках базы знаний $G2'_F$

Так на первом уровне среди трёх узлов не было найдено решение (правильная цепочка), поэтому был осуществлён переход на следующий уровень. На втором уровне количество узлов стало равным 7, но 1-ый, 2-ой, 5-ый и 6-ой (слева направо) узлы, согласно второму правилу роста были отброшены из-за наличия в них аксиомы <FrFric>. Так как среди оставшихся трёх узлов не нашлось правильной цепочки, то был осуществлён переход на следующий, третий уровень.

На третьем уровне было получено 9 новых узлов, из которых 3 удалены: 3-ий и 7-ой – из-за наличия ранее применённого нетерминала <Mass>, а 9-ый – из-за нетерминала <OpFric>. После анализа оставшихся 6 узлов было выяснено, что последний (8-ой на уровне) узел содержит только терминальные символы, т.е. является правильной цепочкой.

Таким образом, узел третьего уровня, состоящий из терминалов:

<FrPull> <Dist> <KinEn2> <KinEn1> <Dist>

Путь, который привёл к этому узлу, представляет собой последовательный вызов функций: f_{12}, f_{17}, f_{18}. С учётом порядка использования этих функций, конечный результат будет представлять собой суперпозицию функций:

<FrFric> = $f_{12}(f_{17}(f_{18}$(<FrPull>, <Dist>), <KinEn2>, <KinEn1>), <Dist>).

Таким образом, пример подтвердил предположение 3 и следствие из него.

Рассмотрим описание логико-математического метода решения задачи моделирования на языке чистого лямбда исчисления. Данный метод основывается на теории, представленной предположениями 1, 2, 3 и следствием и может быть представлен тремя сложными функциями на языке чистого лямбда исчисления:

1) функция *K* создания базы знаний в рамках определенной предметной области (с учетом предположений 1 и 2);

2) функция *G* формулирования задачи моделирования динамической системы (реализация предположения 3);

3) функция *M* вывода логико-математической модели динамической системы (реализация следствия).

Функция *K* была рассмотрена в 2.3. Рассмотрим вывод функции *G*.

Функция построения списка задачи моделирования имеет вид:

$W = (\lambda x.\lambda y.\lambda z.Cs\ x\ (Cs\ y\ (Cs\ z\ Nl)))\ K\ D\ Q.$

Функции выделения алфавита нетерминальных символов:

$Vn = Y\ (\lambda f.\lambda p.\lambda q.If\ (Nu\ p)\ Nl\ (If\ (Vn1\ q\ (Cr\ p))\ (f\ (Cd\ p)\ q)\ (Cn\ (Cr\ p)\ (f\ (Cd\ p)\ q))))$, где:

$Vn1 = Y\ (\lambda f.\lambda r.\lambda t.If\ (Nu\ r)\ Fl\ (If\ (=\ (Cr\ r)\ t)\ Tr\ (f\ (Cd\ r)\ t)))$.

Функция *Vn* и подчиненная ей функция *Vn1* содержат две рекурсии, при этом одна рекурсия вложена во вторую.

Функции вывода полной функциональной грамматики на основе задачи моделирования:

$G = (\lambda x.G2\ (G1\ x)\ (Crdd\ x))\ W$, где:

$G1 = \lambda x.Cn\ (Crr\ x)\ (Cn\ (Crd\ x)\ (Cn\ (Vn\ (Crdr\ x)\ (Crd\ x))\ (Cn\ (Crddr\ x)\ (Cn\ (Crdddr\ x)\ (Cn\ (Crddddr\ x)\ Nl)))))$,

$G2 = Y\ (\lambda f.\lambda x.\lambda y.If\ (Nu\ y)\ Nl\ (Cn\ (Cr\ x)\ (Cn\ (Crd\ x)\ (Cn\ (Crdd\ x)\ (Cn\ (Crddd\ x)\ (Cn\ (Crdddd\ x)\ (Cn\ (Crddddd\ x)\ (Cn\ (Cr\ y)\ (f\ x\ (Cd\ y)))))))))$.

$G2 = Y\ (\lambda f.\lambda x.\lambda y.If\ (Nu\ y)\ Nl\ (Cn\ \ (Cn\ (Cr\ x)\ (Cn\ (Crd\ x)\ (Cn\ (Crdd\ x)\ (Cn\ (Crddd\ x)\ (Cn\ (Crdddd\ x)\ (Cn\ (Crddddd\ x)\ (Cn\ (Cr\ y)\ Nl)))))))\ (f\ x\ (Cd\ y))))$.

Подчиненная функция *G2* имеет рекурсию.

Упростим функцию построения списка задачи моделирования с помощью δ- и β-редукций:

$W = (\lambda x.\lambda y.\lambda z.Cs\ x\ (Cs\ y\ (Cs\ z\ Nl)))\ K\ D\ Q \ \rightarrow_{\beta}\ \ Cs\ K\ (Cs\ D\ (Cs\ Q\ Nl)) \ \rightarrow_{\delta *}$

$(\lambda x.\lambda y.\lambda s.s\ x\ y)\ K\ ((\lambda x.\lambda y.\lambda s.s\ x\ y)\ D\ ((\lambda x.\lambda y.\lambda s.s\ x\ y)\ Q\ Nl)) \ \rightarrow_{\beta *}$

$\lambda s.s\ K\ (\lambda s.s\ D\ (\lambda s.s\ Q\ (\lambda s.s\ Nl)))$.

Упростим функцию *G1*:

$G1 = \lambda x.Cn\ (Crr\ x)\ (Cn\ (Crd\ x)\ (Cn\ (Vn\ (Crdr\ x)\ (Crd\ x))\ (Cn\ (Crddr\ x)\ (Cn\ (Crdddr\ x)\ (Cn\ (Crddddr\ x)\ Nl))))) \ \rightarrow_{\delta}$

$\lambda x.Cn\ (Crr\ x)\ (Cn\ (Crd\ x)\ (Cn\ ((Y\ (\lambda f.\lambda p.\lambda q.If\ (Nu\ p)\ Nl\ (If\ (Vn1\ q\ (Cr\ p))\ (f\ (Cd\ p)\ q)\ (Cn\ (Cr\ p)\ (f\ (Cd\ p)\ q)))))\ (Crdr\ x)\ (Crd\ x))\ (Cn\ (Crddr\ x)\ (Cn\ (Crdddr\ x)\ (Cn\ (Crddddr\ x)\ Nl))))) \ \rightarrow_{\delta}$

$\lambda x.Cn\ (Crr\ x)\ (Cn\ (Crd\ x)\ (Cn\ ((Y\ (\lambda f.\lambda p.\lambda q.If\ (Nu\ p)\ Nl\ (If\ ((Y\ (\lambda f.\lambda r.\lambda t.If\ (Nu\ r)\ Fl$

(*If* (= (*Cr r*) *t*) *Tr* (*f* (*Cd r*) *t*)))) *q* (*Cr p*)) (*f* (*Cd p*) *q*)) (*Cn* (*Cr p*) (*f* (*Cd p*) *q*))))) (*Crdr x*) (*Crd x*)) (*Cn* (*Crddr x*) (*Cn* (*Crdddr x*) (*Cn* (*Crddddr x*) *Nl*)))))

Подставим *W*, *G2*, *G1* и *Vn* в *G* (*δ*-редукции) и упростим (*β*-редукции):

G = (*λx.G2* (*G1 x*) (*Crdd x*)) *W* $\rightarrow_{\beta}$ *G2* (*G1 W*) (*Crdd W*) $\rightarrow_{\delta}$

G2 (*G1* (*λs.s K* (*λs.s D* (*λs.s Q* (*λs.s Nl*))))) (*Crdd* (*λs.s K* (*λs.s D* (*λs.s Q* (*λs.s Nl*))))) $\rightarrow_{\beta}$ *G2* (*G1* (*λs.s K* (*λs.s D* (*λs.s Q* (*λs.s Nl*))))) *Q* $\rightarrow_{\delta}$

(*Y* (*λf.λx.λy.If* (*Nu y*) *Nl* (*Cn* (*Cn* (*Cr x*) (*Cn* (*Crd x*) (*Cn* (*Crdd x*) (*Cn* (*Crddd x*) (*Cn* (*Crdddd x*) (*Cn* (*Crddddd x*) (*Cn* (Cr *y*) *Nl*))))))) (*f x* (*Cd y*))))) (*G1* (*λs.s K* (*λs.s D* (*λs.s Q* (*λs.s Nl*))))) *Q* $\rightarrow_{\delta}$

(*Y* (*λf.λx.λy.If* (*Nu y*) *Nl* (*Cn* (*Cn* (*Cr x*) (*Cn* (*Crd x*) (*Cn* (*Crdd x*) (*Cn* (*Crddd x*) (*Cn* (*Crdddd x*) (*Cn* (*Crddddd x*) (*Cn* (Cr *y*) *Nl*))))))) (*f x* (*Cd y*))))((*λx.Cn* (*Crr x*) (*Cn* (*Crd x*) (*Cn* ((*Y* (*λf.λp.λq.If* (*Nu p*) *Nl* (*If* ((*Y* (*λf.λr.λt.If* (*Nu r*) *Fl* (*If* (= (*Cr r*) *t*) *Tr* (*f* (*Cd r*) *t*)))) *q* (*Cr p*)) (*f* (*Cd p*) *q*)) (*Cn* (*Cr p*) (*f* (*Cd p*) *q*))))) (*Crdr x*) (*Crd x*)) (*Cn* (*Crddr x*) (*Cn* (*Crdddr x*) (*Cn* (*Crddddr x*) *Nl*))))))(*λs.s K* (*λs.s D* (*λs.s Q* (*λs.s Nl*))))) *Q* $\rightarrow_{\beta}$

(*Y* (*λf.λx.λy.If* (*Nu y*) *Nl* (*Cn* (*Cn* (*Cr x*) (*Cn* (*Crd x*) (*Cn* (*Crdd x*) (*Cn* (*Crddd x*) (*Cn* (*Crdddd x*) (*Cn* (*Crddddd x*) (*Cn* (Cr *y*) *Nl*))))))) (*f x* (*Cd y*))))((*λx.Cn* (*Crr x*) (*Cn* (*Crd x*) (*Cn* ((*Y* (*λf.λp.λq.If* (*Nu p*) *Nl* (*If* ((*Y* (*λf.λr.λt.If* (*Nu r*) *Fl* (*If* (= (*Cr r*) *t*) *Tr* (*f* (*Cd r*) *t*)))) *q* (*Cr p*)) (*f* (*Cd p*) *q*)) (*Cn* (*Cr p*) (*f* (*Cd p*) *q*))))) (*Crdr x*) (*Crd x*)) (*Cn* (*Crddr x*) (*Cn* (*Crdddr x*) (*Cn* (*Crddddr x*) *Nl*))))))(*λs.s K* (*λs.s D* (*λs.s Q* (*λs.s Nl*))))) $\rightarrow_{\beta}$

После еще одной β-редукции функция *G* имеет вид:

G = *Y* (*λf.λx.λy.If* (*Nu y*) *Nl* (*Cn* (*Cr x*) (*Cn* (*Crd x*) (*Cn* (*Crdd x*) (*Cn* (*Crddd x*) (*Cn* (*Crdddd x*) (*Cn* (*Crddddd x*) (*Cn* (Cr *y*) (*f x* (*Cd y*)))))))) (*Cn* (*Cr K*) (*Cn D* (*Cn* ((*Y* (*λf.λp.λq.If* (*Nu p*) *Nl* (*If* ((*Y* (*λf.λr.λt.If* (*Nu r*) *Fl* (*If* (= (*Cr r*) *t*) *Tr* (*f* (*Cd r*) *t*)))) *q* (*Cr p*)) (*f* (*Cd p q*)) (*Cn* (*Cr p*) (*f* (*Cd p*) *q*))))) (*Crd K*) (*Crd D*)) (*Cn* (*Crdd K*) (*Cn* (*Crddd K*) (*Cn* (*Crdddd K*) *Nl*))))) *Q*)

Эту же функцию *G* можно выразить через аргументы функции *K*. Для этого подставим *K* в *G* (*δ*-редукции) и упростим (*β*-редукции):

G = (*Y* (*λf.λx.λy.If* (*Nu y*) *Nl* (*Cn* (*Cn* (*Cr x*) (*Cn* (*Crd x*) (*Cn* (*Crdd x*) (*Cn* (*Crddd x*) (*Cn* (*Crdddd x*) (*Cn* (*Crddddd x*) (*Cn* (Cr *y*) *Nl*))))))) (*f x* (*Cd y*))))(*Cn* (*Cr K*) (*Cn D*

(*Cn* ((*Y* (*λf.λp.λq.If* (*Nu p*) *Nl* (*If* ((*Y* (*λf.λr.λt.If* (*Nu r*) *Fl* (*If* (= (*Cr r*) *t*) *Tr* (*f* (*Cd r*) *t*)))) *q* (*Cr p*)) (*f* (*Cd p*) *q*)) (*Cn* (*Cr p*) (*f* (*Cd p*) *q*))))) (*Crd K*) *D*) (*Cn* (*Crdd K*) (*Cn* (*Crddd K*) (*Cn* (*Crdddd K*) *Nl*)))))) *Q* $\rightarrow_{\delta\beta^*}$

(*Y* (*λf.λx.λy.If* (*Nu y*) *Nl* (*Cn* (*Cn* (*Cr x*) (*Cn* (*Crd x*) (*Cn* (*Crdd x*) (*Cn* (*Crddd x*) (*Cn* (*Crdddd x*) (*Cn* (*Crddddd x*) (*Cn* (Cr *y*) *Nl*)))))) (*f x* (*Cd y*))))) (*λs.s* V_{T0} (*λs.s D* (*λs.s* ((*Y* (*λf.λp.λq.If* (*Nu p*) *Nl* (*If* ((*Y* (*λf.λr.λt.If* (*Nu r*) *Fl* (*If* (= (*Cr r*) *t*) *Tr* (*f* (*Cd r*) *t*)))) *q* (*Cr p*)) (*f* (*Cd p*) *q*)) (*Cn* (*Cr p*) (*f* (*Cd p*) *q*))))) *V D*) (*λs.s* F_0 (*λs.s F* (*λs.s P* (*λs.s Nl*)))))) *Q*

Аналогичным образом выводится более громоздкая функция поиска решения задачи моделирования *M*.

На рисунке 15 приведена общая схема логико-математического метода моделирования динамических систем посредством функциональных грамматик.

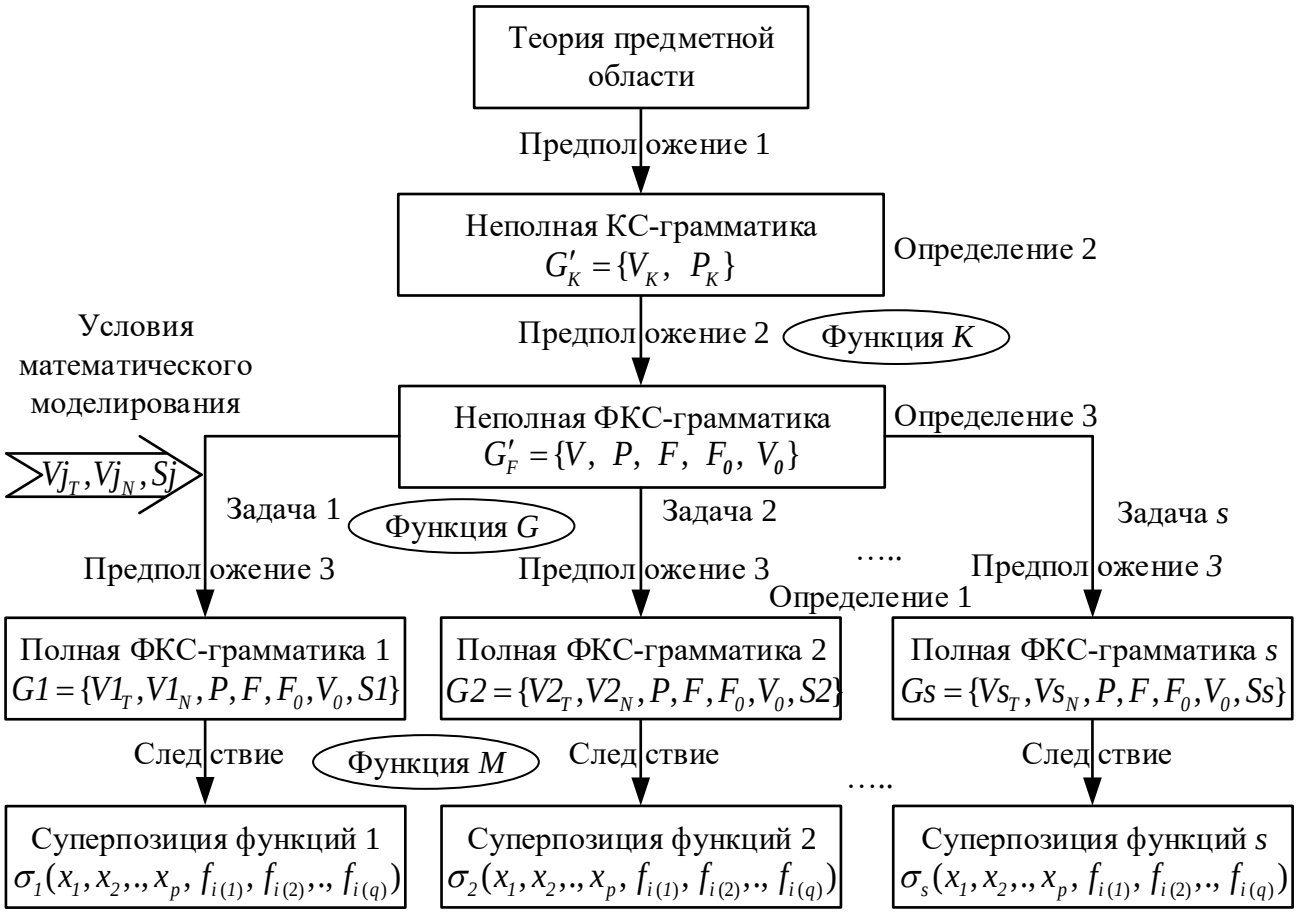

Рисунок 15 – Схема логико-математического моделирования динамических систем с использованием аппарата функциональных грамматик

ГЛАВА 3. ЛОГИКО-МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ РАДИОТЕХНИЧЕСКИХ СИСТЕМ

3.1 Ограничение области знаний

Для проверки достоверности методики логико-математического моделирования, описанной в главе 2, составим базу знаний по радиотехнике [41, 45] и проверим её адекватность на прямых и обратных задачах моделирования.

Принципы моделирования радиотехнических систем описаны в [12, 32, 65].

Первым этапом является ограничение теории, по которой производится составление базы знаний. Согласно [9] наиболее распространённым видом радиотехнических динамических систем являются линейные стационарные системы. Ограничимся именно такими системами.

Линейность определяется двумя свойствами:

$$\boldsymbol{T}(\beta)\left(\boldsymbol{X}_1(\alpha_1,\ t)+\boldsymbol{X}_2(\alpha_2,\ t)\right)=\boldsymbol{T}(\beta)\boldsymbol{X}_1(\alpha_1,\ t)+\boldsymbol{T}(\beta)\boldsymbol{X}_2(\alpha_2,\ t);$$

$$\boldsymbol{T}(\beta)\left(\delta\boldsymbol{X}(\alpha,\ t)\right)=\delta\boldsymbol{T}(\beta)\boldsymbol{X}(\alpha,\ t);$$

где $\boldsymbol{X}_1(\alpha_1,\ t)$, $\boldsymbol{X}_2(\alpha_2,\ t)$ – два векторных входных сигнала; δ – произвольное число.

Стационарность описывается системой уравнений:

$$\begin{cases}\boldsymbol{Y}(t)=\boldsymbol{T}(\beta)\boldsymbol{X}(\alpha,\ t),\\ \boldsymbol{Y}(t\pm t_0)=\boldsymbol{T}(\beta)\boldsymbol{X}(\alpha,\ t\pm t_0),\end{cases}$$

где t_0 – произвольный временной сдвиг.

Определимся с внутренней структурной схемой, определяемой системным оператором $\boldsymbol{T}(\beta)$. Ограничимся одним входом и одним выходом. В этом случае, динамическая система будет представлять собой четырехполюсник. Из [77] известно, что четырехполюсники могут быть трёх видов: *Г*-образные, *Т*-образные и *П*-образные. Но при подключении нагрузки к выходу *Т*-образные и *П*-образные четырехполюсники можно представить в виде двойного *Г*-образного четырехполюсника (рисунок 16). Поэтому в качестве внутренней структуры выбираем *Г*-образный четырехполюсник, содержащий две ветви: Z_1 и Z_2.

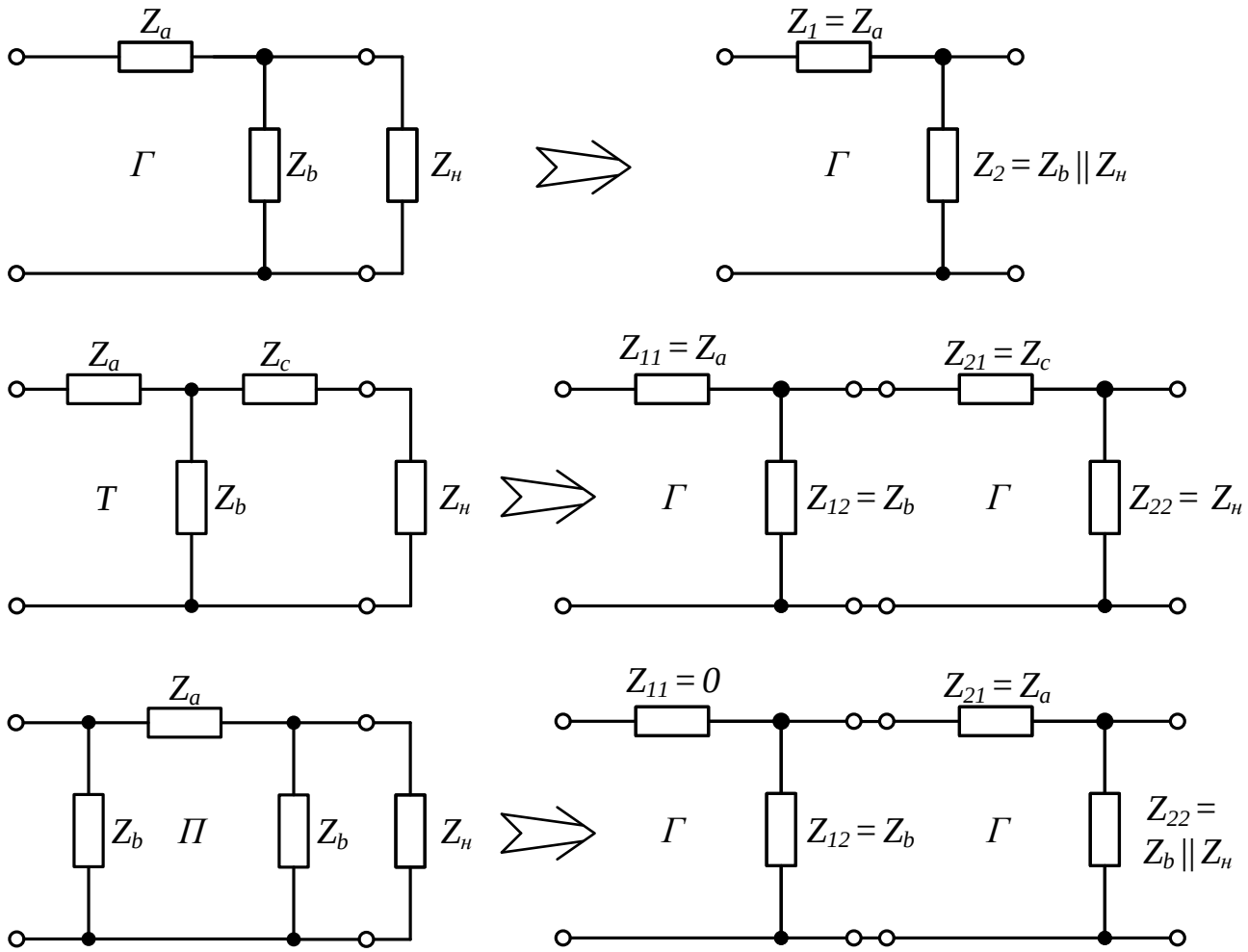

Рисунок 16 – Виды четырехполюсников

Определим внутреннюю структуру ветвей четырехполюсника, т.е. $\boldsymbol{T}(\beta)$. Любой элемент электрической цепи, в том числе полупроводниковые приборы можно представить эквивалентной схемой, содержащей до 5 возможных элементов (рисунок 17): источник ЭДС E, источник тока J, активное сопротивление R, индуктивность L и электрическая ёмкость C.

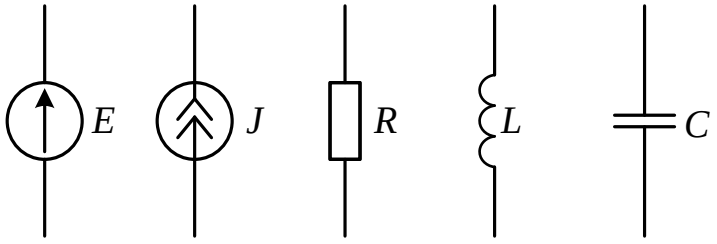

Рисунок 17 – Элементы электрических цепей

Ограничимся только пассивными радиотехническими цепями. Это означает, что схема должна содержать элементы *R*, *L* и *C*.

Теперь определимся с требованиями, предъявляемыми к входным и выходным сигналам:

$$U_{ВХ}(t) = \boldsymbol{X}(\alpha,\ t);\ U_{ВЫХ}(t) = \boldsymbol{Y}(t).$$

Эти сигналы должны быть одномерными, т.к. выбрана схема в виде четырехполюсника. Для решения реальных задач сигналы должны быть недетерминированными, поэтому их аналитическое задание не представляется возможным. Значит, сигналы должны быть заданы в числовой форме, предназначенной для применения численных методов. Это можно добиться условной дискретизацией входного и выходного сигнала по теореме Котельникова [85]. Условность дискретизации означает, что в реальности сигналы могут оставаться аналоговыми, а для работы модели они будут представлены в дискретном виде.

Таким образом, моделируемая радиотехническая динамическая система должна представлять собой линейную стационарную пассивную электрическую цепь активных, индуктивных и емкостных элементов в виде Г-образного четырехполюсника, преобразующую входной дискретный радиотехнический сигнал в выходной дискретный сигнал, подчиняющиеся теореме Котельникова. (рисунок 18). Такому описанию соответствует, например, класс пассивных электрических фильтров, работающих с аналоговыми (условно дискретными) сигналами.

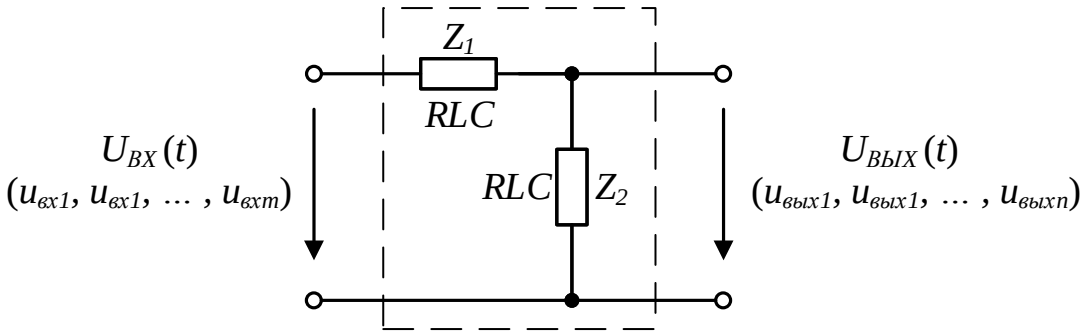

Рисунок 18 – Моделируемая радиотехническая динамическая система

3.2 Описание алфавита символов

Вторым этапом составления базы знаний является выявление всех понятий теории и составление обобщенного алфавита символов, а также описание синтаксиса каждого символа.

Составим алфавит символов функциональной грамматики $\boldsymbol{V_{PT}}$. Для этого выберем набор понятий, которыми можно охарактеризовать систему. Понятия делятся на три группы: характеристики входного сигнала $\boldsymbol{X}(\alpha, t)$, параметры цепи $\boldsymbol{T}(\beta)$ и характеристики выходного сигнала $\boldsymbol{Y}(t)$ (таблица 6).

Так как анализ радиотехнических систем производится во временной и частотной областях, то среди характеристик сигналов выбираем: временные зависимости $u_{ВХ}(t)$, $u_{ВЫХ}(t)$ и комплексные спектральные представления (спектральные плотности) $U_{ВХ}(j\omega)$, $U_{ВЫХ}(\omega)$. Кроме того, отдельное значение имеют составляющие спектральных плотностей: амплитудные спектры $U_{ВХ}(\omega)$, $U_{ВЫХ}(\omega)$ и фазовые спектры $\varphi_{UВХ}(\omega)$, $\varphi_{UВЫХ}(\omega)$. Таким образом, к первой и третьей группе будут относится по 4 понятия.

Для цепи выделим три подгруппы понятий: параметры принципиальной схемы, временные характеристики и спектральные характеристики. К параметрам принципиальной схемы отнесем входное сопротивление в операторной форме $Z_{ВХ}(p)$ и выходное сопротивление $Z_{ВХ}(p)$. Операторная форма удобна для описания цепей. Кроме того, на основе операторного представления ветвей Γ-образного четырехполюсника можно ввести операторное представление принципиальной схемы $\Gamma(p)$.

В качестве временных параметров выбираем переходную $g(t)$ и импульсную характеристики $h(t)$; а в качестве спектральных характеристик – комплексный частотный коэффициент передачи $K(j\omega)$, операторную передаточную характеристику $K(p)$, а также их производные – амплитудно-частотную $K(\omega)$ и фазочастотную характеристики $\varphi_K(\omega)$. В итоге, ко второй группе понятий будут относится 9 параметров и характеристик.

Таблица 6 – Характеристики радиотехнической системы

Первая группа Характеристики входного сигнала $\boldsymbol{X}(\alpha, t)$	**Вторая группа** Характеристики цепи $\boldsymbol{T}(\beta)$	**Третья группа** Характеристики выходного сигнала $\boldsymbol{Y}(t)$
Зависимость во временной области $u_{BX}(t)$	Принципиальная схема $\Gamma(p)$	Зависимость во временной области $u_{BЫX}(t)$
	Входное операторное сопротивление $Z_{BX}(p)$	
	Выходное операторное сопротивление $Z_{BЫX}(p)$	
Спектральная плотность $U_{BX}(j\omega)$	Переходная характеристика $g(t)$	Спектральная плотность $U_{BЫX}(j\omega)$
	Импульсная характеристика $h(t)$	
Амплитудный спектр $U_{BX}(\omega)$	Передаточная характеристика $K(p)$	Амплитудный спектр $U_{BЫX}(\omega)$
	Частотный коэффициент передачи $K(j\omega)$	
Фазовый спектр $\varphi_{UBX}(\omega)$	Амплитудно-частотная характеристика $K(\omega)$	Фазовый спектр $\varphi_{UBЫX}(\omega)$
	Фазочастотная характеристика $\varphi_K(\omega)$	

Каждому из понятий сопоставим один символ (таблица 7). В итоге получим 17 символов.

Таблица 7– Алфавит символов контекстно-свободной грамматики

Характеристика линейной стационарной радиотехнической системы	Обозначение	Символ алфавита
1. Принципиальная схема	Γ	<Схема>
2. Входное операторное сопротивление	$Z_{BX}(p)$	<Сопр1>
3. Выходное операторное сопротивление	$Z_{BЫX}(p)$	<Сопр2>
4. Переходная характеристика	g	<Перех>
Импульсная характеристика	h	<Импул>
6. Передаточная характеристика	K	<Перед>
7. Частотный коэффициент передачи	K	<Коэфф>

Продолжение таблицы 7

Характеристика линейной стационарной радиотехнической системы	Обозначение	Символ алфавита
Амплитудно-частотная характеристика	K	<АмпЧХ>
9. Фазочастотная характеристика	$\varphi_K(\omega)$	<ФазЧХ>
10. Входной сигнал	$u_{BX}(t)$	<Сигн1>
11. Спектральная плотность входного сигнала	$U_{BX}(j\omega)$	<Спек1>
12. Амплитудный спектр входного сигнала	$U_{BX}(\omega)$	<АмпС1>
13. Фазовый спектр входного сигнала	$\varphi_{UBX}(\omega)$	<ФазС1>
14. Выходной сигнал	$u_{ВЫХ}(t)$	<Сигн2>
15. Спектральная плотность выходного сигнала	$U_{ВЫХ}(j\omega)$	<Спек2>
16. Амплитудный спектр выходного сигнала	$U_{ВЫХ}(\omega)$	<АмпС2>
Фазовый спектр выходного сигнала	$\varphi_{UВЫХ}(\omega)$	<ФазС2>

В итоге, алфавит V_{PT} принимает вид:

$$V_{PT} = \left\{ \begin{array}{l} < \text{Схема} >,\ < \text{Сопр1} >,\ < \text{Сопр2} >,\ < \text{Перех} >,\ < \text{Импул} >,\ < \text{Перед} >, \\ < \text{Коэфф} >,\ < \text{АмпЧХ} >,\ < \text{ФазЧХ} >,\ < \text{Сигн1} >,\ < \text{Спек1} >,\ < \text{АмпС1} >, \\ < \text{ФазС1} >,\ < \text{Сигн2} >,\ < \text{Спек2} >,\ < \text{АмпС2} >,\ < \text{ФазС2} > \end{array} \right\}.$$

Запишем синтаксис значения каждого из символов функциональной грамматики в виде обычных контекстно-свободных грамматик.

1. Синтаксис символа <Схема>

$$G_{C1} = (V_{T\,C1},\ V_{N\,C1},\ P_{C1},\ S_{C1});$$

$$V_{T\,C1} = \{0,\ 1,\ 2,\ \ldots,\ 9,\ (,\ ),\ \_,\ \text{Схема}\};$$

$$V_{N\,C1} = \left\{ \begin{array}{l} < \text{Схема} >,\ < \text{ДРФ} >,\ < \text{числитель} >,\ < \text{знаменатель} >, \\ < \text{многочлен} >,\ < \text{числа} >,\ < \text{число} >,\ < \text{цифра} > \end{array} \right\};$$

$$\begin{array}{rl} P_{C1} = \{ & < \text{Схема} > \rightarrow (\ \text{Схема}\ \_\ < \text{ДРФ} >\ \_\ < \text{ДРФ} >\ ), \\ & < \text{ДРФ} > \rightarrow\ < \text{числитель} >\ \_\ < \text{знаменатель} >, \\ & < \text{числитель} > \rightarrow\ < \text{многочлен} >, \\ & < \text{знаменатель} > \rightarrow\ < \text{многочлен} >, \\ & < \text{многочлен} > \rightarrow\ (\ < \text{числа} >\ ), \\ & < \text{числа} > \rightarrow < \text{число} > \ |\ < \text{числа} >\ \_\ < \text{число} >, \end{array}$$

$$< \text{число} > \rightarrow < \text{цифра} > \mid < \text{число} > < \text{цифра} >,$$

$$< \text{цифра} > \rightarrow 0 \mid 1 \mid 2 \mid \ldots \mid 9 \quad \};$$

$S_{C1} = < \text{Схема} >.$

2. Синтаксис символов <Сопр1>, <Сопр2>, <Перед>

$G_{C2} = (V_{T\,C2},\ V_{N\,C2},\ P_{C2},\ S_{C2});$

$V_{T\,C2} = \{0,\ 1,\ 2,\ \ldots,\ 9,\ (,\ ),\ \_,\ \text{ДРФ}\};$

$$V_{N\,C2} = \left\{ \begin{array}{l} < \text{Сопр1} >,\ < \text{ДРФ} >,\ < \text{числитель} >,\ < \text{знаменатель} >, \\ < \text{многочлен} >,\ < \text{числа} >,\ < \text{число} >,\ < \text{цифра} > \end{array} \right\};$$

$P_{C2} = \{$

$$< \text{Сопр1} > \rightarrow (\ \text{ДРФ}\ \_\ < \text{ДРФ} >),$$

$$< \text{ДРФ} > \rightarrow\ < \text{числитель} >\ \_\ < \text{знаменатель} >,$$

$$< \text{числитель} > \rightarrow < \text{многочлен} >,$$

$$< \text{знаменатель} > \rightarrow < \text{многочлен} >,$$

$$< \text{многочлен} > \rightarrow (\ < \text{числа} >),$$

$$< \text{числа} > \rightarrow < \text{число} > \mid < \text{числа} > \_ < \text{число} >,$$

$$< \text{число} > \rightarrow < \text{цифра} > \mid < \text{число} > < \text{цифра} >,$$

$$< \text{цифра} > \rightarrow 0 \mid 1 \mid 2 \mid \ldots \mid 9 \quad \};$$

$S_{C2} = < \text{Сопр1} >.$

3. Синтаксис символов <Перех>, <Импул>

$G_{C3} = (V_{T\,C3},\ V_{N\,C3},\ P_{C3},\ S_{C3});$

$V_{T\,C3} = \{0,\ 1,\ 2,\ \ldots,\ 9,\ (,\ ),\ \_,\ \text{Функция}\};$

$$V_{N\,C3} = \left\{ \begin{array}{l} < \text{Перех} >,\ < \text{слагаемые} >,\ < \text{слагаемое} >,\ < \text{экспонента} >, \\ < \text{экспосинус} >,\ < \text{модуль} >,\ < \text{степень} >,\ < \text{частота} >, \\ < \text{аргумент} >,\ < \text{числа} >,\ < \text{число} >,\ < \text{цифра} > \end{array} \right\};$$

$P_{C3} = \{$

$$< \text{Перех} > \rightarrow (\ \text{Функция}\ \_\ < \text{слагаемые} >),$$

$$< \text{слагаемые} > \rightarrow < \text{слагаемое} > \mid < \text{слагаемые} > \_$$

$$< \text{слагаемое} >,$$

$$< \text{слагаемое} > \rightarrow\ < \text{экспонента} > \mid < \text{экспосинус} >,$$

$$< \text{экспонента} > \rightarrow\ (< \text{модуль} >\ \ < \text{степень} >),$$

< экспосинус > → (< модуль > < степень > < частота >
< аргумент >),
< модуль > → (< число >),
< степень > → (< число >),
< частота > → (< число >),
< аргумент > → (< число >),
< число > → < цифра > | < число > < цифра >,
< цифра > → 0 | 1 | 2 | … | 9 };

$S_{C3} = <\text{Перех}>$.

4. Синтаксис символа <Коэфф>

$G_{C4} = (V_{T\,C4},\ V_{N\,C4},\ P_{C4},\ S_{C4})$;

$V_{T\,C4} = \{0,\ 1,\ 2,\ \ldots,\ 9,\ (,\ ),\ \_,\ \text{ДКФ}\}$;

$$V_{N\,C4} = \left\{ \begin{array}{c} <\text{Коэфф}>,\ <\text{ДКФ}>,\ <\text{числитель}>,\ <\text{знаменатель}>, \\ <\text{комплфунк}>,<\text{действчасть}>,\ <\text{мнимчасть}>, \\ <\text{многочлен}>,\ <\text{числа}>,\ <\text{число}>,\ <\text{цифра}> \end{array} \right\};$$

P_{C4} = { <Коэфф> → (ДКФ _ < ДКФ >),
< ДКФ > → < числитель > _ < знаменатель >,
< числитель > → < комплфунк >,
< знаменатель > → < комплфунк >,
< комплфунк > → (< действчасть > _ < мнимчасть >),
< действчасть > → < многочлен >,
< мнимчасть > → < многочлен >,
< многочлен > → (< числа >),
< числа > → < число > | < числа > _ < число >,
< число > → < цифра > | < число > < цифра >,
< цифра > → 0 | 1 | 2 | … | 9 };

$S_{C4} = <\text{Коэфф}>$.

5. Синтаксис символа <АмпЧХ>

$G_{C5} = (V_{T\,C5},\ V_{N\,C5},\ P_{C5},\ S_{C5})$;

$V_{T\,C5} = \{0,\ 1,\ 2,\ \ldots,\ 9,\ (,\ ),\ \_,\ \text{МДКФ}\}$;

$$V_{N\,C5} = \left\{ \begin{array}{l} < \text{АмпЧХ} >,\ < \text{МДКФ} >,\ < \text{числитель} >,\ < \text{знаменатель} >, \\ < \text{квадрат} >,\ < \text{многочлен} >,\ < \text{числа} >,\ < \text{число} >,\ < \text{цифра} > \end{array} \right\};$$

$P_{C5} =$ {
<АмпЧХ> → (МДКФ _ < МДКФ >),
< МДКФ > → < числитель > _ < знаменатель >,
< числитель > → < квадрат > _ < квадрат >,
< знаменатель > → < квадрат > _ < квадрат >,
< квадрат > → < многочлен >,
< многочлен > → (< числа >),
< числа > → < число > | < числа > _ < число >,
< число > → < цифра > | < число > < цифра >,
< цифра > → 0 | 1 | 2 | … | 9 };

$S_{C5} = < \text{АмпЧХ} >$.

6. Синтаксис символа <ФазЧХ>

$G_{C6} = (V_{T\,C6},\ V_{N\,C6},\ P_{C6},\ S_{C6})$;

$V_{T\,C6} = \{0,\ 1,\ 2,\ \ldots,\ 9,\ (,\ ),\ \_,\ \text{АДКФ}\}$;

$$V_{N\,C6} = \left\{ \begin{array}{l} < \text{ФазЧХ} >,\ < \text{АДКФ} >,\ < \text{арктангенс} >, \\ < \text{многочлен} >,\ < \text{числа} >,\ < \text{число} >,\ < \text{цифра} > \end{array} \right\};$$

$P_{C6} =$ {
<ФазЧХ> → (АДКФ _ < АДКФ >),
< АДКФ > → < арктангенс > _ < арктангенс >,
< арктангенс > → < многочлен > _ < многочлен >,
< многочлен > → (< числа >),
< числа > → < число > | < числа > _ < число >,
< число > → < цифра > | < число > < цифра >,
< цифра > → 0 | 1 | 2 | … | 9 };

$S_{C6} = < \text{ФазЧХ} >$.

7. Синтаксис символов <Сигн1>, <Сигн2>, <АмпС1>, <АмпС2>, <ФазС1> и <ФазС2>

$$G_{C7} = (V_{T\,C7},\ V_{N\,C7},\ P_{C7},\ S_{C7});$$

$$V_{T\ C7} = \{0,\ 1,\ 2,\ \ldots,\ 9,\ (,\ ),\ \_,\ \text{Ряд}\};$$

$$V_{N\ C7} = \left\{ \begin{array}{l} < \text{Сигн1} >,\ < \text{интервал} >,\ < \text{коэффициент} >, \\ < \text{ряд} >,\ < \text{числа} >,\ < \text{число} >,\ < \text{цифра} > \end{array} \right\};$$

$P_{C7} = \{$ <Сигн1> → (Ряд _ < интервал > _ < коэффициент > _ <ряд>),

< интервал > → < число >,

< коэффициент > → < число >,

<ряд> → (< числа >),

< числа > → < число > | < числа > _ < число >,

< число > → < цифра > | < число > < цифра >,

< цифра > → 0 | 1 | 2 | … | 9 };

$S_{C7} = < \text{Сигн1} >.$

8. Синтаксис символов <Спек1> и <Спек2>

$$G_{C8} = (V_{T\,C8},\ V_{N\,C8},\ P_{C8},\ S_{C8});$$

$$V_{T\,C8} = \{0,\ 1,\ 2,\ \ldots,\ 9,\ (,\ ),\ \_,\ \text{РядК}\};$$

$$V_{N\ C8} = \left\{ \begin{array}{l} < \text{Спек1} >,\ < \text{интервал} >,\ < \text{коэффициент} >,\ < \text{рядК} >, \\ < \text{мод} >,\ < \text{арг} >,\ < \text{комплчисло} >,\ < \text{комплчисла} >, \\ < \text{число} >,\ < \text{цифра} > \end{array} \right\};$$

$P_{C8} = \{$ <Спек1> → (РядК _ <интервал> _ <коэффициент> _ <рядК>),

< интервал > → < число >,

< коэффициент > → < число >,

<рядК> → (< комплчисла >),

< комплчисла > → <комплчисло> | <комплчисла > _ <комплчисло>,

$$< \text{комплчисло} > \rightarrow (< \text{мод} > \_ < \text{арг} >),$$

$$< \text{мод} > \rightarrow < \text{число} >,$$

$$< \text{арг} > \rightarrow < \text{число} >,$$

$$< \text{число} > \rightarrow < \text{цифра} > \mid < \text{число} > < \text{цифра} >,$$

$$< \text{цифра} > \rightarrow 0 \mid 1 \mid 2 \mid \ldots \mid 9 \quad \};$$

$S_{C8} = < \text{Спек1} >$.

Таким образом, каждый из 17 символов полученной функциональной грамматики имеет один из 8 синтаксисов: G_{C1}, G_{C2}, ... , G_{C8}. Пояснения к ним представлены в таблице 8.

Таблица 8 – Синтаксис терминальных символов

Синтаксис	Выражение синтаксиса	Обозначение	Символы алфавита
G_{C1}	Принципиальная схема, выраженная операторными сопротивлениями	Схема	<Схема>
G_{C2}	Дробно-рациональная функция вида $\frac{a_n p^n + a_{n-1}p^{n-1} + \ldots + a_2 p^2 + a_1 p + a_0}{b_m p^m + b_{m-1}p^{m-1} + \ldots + b_2 p^2 + b_1 p + b_0}$	ДРФ	<Сопр1> <Сопр2> <Перед>
G_{C3}	Функция	Функция	<Перех> <Импул>
G_{C4}	Комплексная дробно-рациональная функция вида $\frac{\left[(a_{2n}\omega^{2n} - a_{2(n-1)}\omega^{2(n-1)} + \ldots + a_4\omega^4 - a_2\omega^2 + a_0) + \right.}{\left. + j(a_{2n-1}\omega^{2n-1} - a_{2(n-1)-1}\omega^{2(n-1)-1} + \ldots + a_5\omega^5 - a_3\omega^3 + a_1\omega)\right]}$ $\frac{\left[(b_{2m}\omega^{2m} - b_{2(m-1)}\omega^{2(m-1)} + \ldots + b_4\omega^4 - b_2\omega^2 + b_0) + \right.}{\left. + j(b_{2m-1}\omega^{2m-1} - b_{2(m-1)-1}\omega^{2(m-1)-1} + \ldots + b_5\omega^5 - b_3\omega^3 + b_1\omega)\right]}$	ДКФ	<Коэфф>
G_{C5}	Модуль комплексной дробно-рациональной функции $\sqrt{\frac{\left[(a_{2n}\omega^{2n} - a_{2(n-1)}\omega^{2(n-1)} + \ldots + a_4\omega^4 - a_2\omega^2 + a_0)^2 + \right.}{\left. + (a_{2n-1}\omega^{2n-1} - a_{2(n-1)-1}\omega^{2(n-1)-1} + \ldots + a_5\omega^5 - a_3\omega^3 + a_1\omega)^2\right]}}$ $\sqrt{\frac{\left[(b_{2m}\omega^{2m} - b_{2(m-1)}\omega^{2(m-1)} + \ldots + b_4\omega^4 - b_2\omega^2 + b_0)^2 + \right.}{\left. + (b_{2m-1}\omega^{2m-1} - b_{2(m-1)-1}\omega^{2(m-1)-1} + \ldots + b_5\omega^5 - b_3\omega^3 + b_1\omega)^2\right]}}$	МДКФ	<АмпЧХ>

Продолжение таблицы 8

Синтаксис	Выражение синтаксиса	Обозначение	Символы алфавита
G_{C6}	Аргумент комплексной дробно-рациональной функции $\text{arctg}\left(\frac{a_{2n-1}\omega^{2n-1} - a_{2(n-1)-1}\omega^{2(n-1)-1} + \ldots + a_5\omega^5 - a_3\omega^3 + a_1\omega}{a_{2n}\omega^{2n} - a_{2(n-1)}\omega^{2(n-1)} + \ldots + a_4\omega^4 - a_2\omega^2 + a_0}\right) -$ $-\text{arctg}\left(\frac{b_{2m-1}\omega^{2m-1} - b_{2(m-1)-1}\omega^{2(m-1)-1} + \ldots + b_5\omega^5 - b_3\omega^3 + b_1\omega}{b_{2m}\omega^{2m} - b_{2(m-1)}\omega^{2(m-1)} + \ldots + b_4\omega^4 - b_2\omega^2 + b_0}\right)$	АДКФ	<ФазЧХ>
G_{C7}	Ряд вещественных значений	Ряд	<Сигн1> <Сигн2> <АмпС1> <АмпС2> <ФазС1> <ФазС2>
G_{C8}	Ряд комплексных значений	РядК	<Спек1> <Спек2>

3.3 Определение набора правил

Третьим этапом составления базы знаний является выявление отношений между выбранными характеристиками системы. Это удобно сделать в виде таблицы 9.

Исходя из таблицы 9, можно записать систему правил:

P_{PT} = { <Схема> → <Сопр1> <Сопр2> {*f1*};

<Сопр1> → <Схема> {*f2*};

<Сопр1> → <Сопр2> <Перех> {*f3*};

<Сопр1> → <Сопр2> <Перед> {*f4*};

<Сопр2> → <Схема> {*f5*};

<Сопр2> → <Сопр1> <Перех> {*f6*};

<Сопр2> → <Сопр1> <Перед> {*f7*};

<Перех> → <Сопр1> <Сопр2> {*f8*};

Таблица 9 – Отношения между понятиями радиотехники

	<Схема>	<Сопр1>	<Сопр2>	<Перех>	<Импул>	<Перед>	<Коэфф>	<АмпЧХ>	<ФазЧХ>	<Сигн1>	<Спек1>	<АмпС1>	<ФазС1>	<Сигн2>	<Спек2>	<АмпС2>	<ФазС2>
<Схема>		*f1*	*f1*														
<Сопр1>	*f2*		*f3/f4*	*f3*		*f4*											
<Сопр2>	*f5*	*f6/f7*		*f6*		*f7*											
<Перех>		*f8*	*f8*		*f9*												
<Импул>				*f10*			*f11*										
<Перед>		*f4*	*f4*				*f12*										
<Коэфф>					*f13*	*f14*		*f15*	*f15*		*f16*				*f16*		
<АмпЧХ>							*f17*					*f18*				*f18*	
<ФазЧХ>							*f19*						*f20*				*f20*
<Сигн1>											*f21*						
<Спек1>							*f22*			*f23*		*f24*	*f24*		*f22*		
<АмпС1>								*f25*			*f26*					*f25*	
<ФазС1>									*f27*		*f28*						*f27*
<Сигн2>				*f29*	*f30*					*f29/ f30*					*f21*		
<Спек2>							*f31*				*f31*			*f23*		*f24*	*f24*
<АмпС2>								*f32*				*f32*			*f26*		
<ФазС2>									*f33*				*f33*		*f28*		

<Перех> → <Импул> {*f9*};

<Импул> → <Перех> {*f10*};

<Импул> → <Коэфф> {*f11*};

<Перед> → <Сопр1> <Сопр2> {*f4*};

<Перед> → <Коэфф> {*f12*};

<Коэфф> → <Импул> {*f13*};

<Коэфф> → <Перед> {f*14*};

<Коэфф> → <АмпЧХ> <ФазЧХ> {*f15*};

<Коэфф> → <Спек1> <Спек2> {*f16*};

<АмпЧХ> → <Коэфф> {*f17*};

<АмпЧХ> → <АмпС1> <АмпС2> {*f18*};

<ФазЧХ> → <Коэфф> {*f19*};

<ФазЧХ> → <ФазС1> <ФазС2> {*f20*};

<Сигн1> → <Спек1> {*f21*};

<Спек1> → <Коэфф> <Спек2> {*f22*};

<Спек1> → <Сигн1> {*f23*};

<Спек1> → <АмпС1> <ФазС1> {*f24*};

<АмпС1> → <АмпЧХ> <АмпС2> {*f25*};

<АмпС1> → <Спек1> {*f26*};

<ФазС1> → <ФазЧХ> <ФазС2> {*f27*};

<ФазС1> → <Спек1> {*f28*};

<Сигн2> → <Перех> <Сигн1> {*f29*};

<Сигн2> → <Импул> <Сигн1> {*f30*};

<Сигн2> → <Спек2> {*f21*};

<Спек2> → <Коэфф> <Спек1> {*f31*};

<Спек2> → <Сигн2> {*f23*};

<Спек2> → <АмпС2> <ФазС2> {*f24*};

<АмпС2> → <АмпЧХ> <АмпС1> {*f32*};

<АмпС2> → <Спек2> {*f26*};

<ФазС2> → <ФазЧХ> <ФазС1> {*f33*};

<ФазС2> → <Спек2> {*f28*} }.

3.4 Определение набора функций

Завершающим этапом составления базы знаний является реализация набора функций F_{PT}, используемых в системе правил P_{PT}. В таблице 10 перечислены все полученные функции с указанием их назначения.

Таблица 10 – Набор функций для реализации законов радиотехники

Функ-ция	Суть преобразования	Описываемые взаимосвязи
f1	Построение схемы цепи по операторным входному и выходному сопротивлениям	$\begin{matrix} Z_{ВХ}(p) \\ Z_{ВЫХ}(p) \end{matrix} \Rightarrow$ Схема
f2	Расчет операторного входного сопротивления цепи по схеме	$\text{Схема} \Rightarrow Z_{ВХ}(p)$
f3	Расчет операторного входного сопротивления цепи по операторному выходному сопротивлению и переходной характеристике	$\begin{matrix} Z_{ВЫХ}(p) \\ g(t) \end{matrix} \Rightarrow Z_{ВХ}(p)$
f4	Деление операторных дробно-рациональных функций (ДРФ)	$Z_{ВХ}(p) = \frac{Z_{ВЫХ}(p)}{K(p)}$ $K(p) = \frac{Z_{ВЫХ}(p)}{Z_{ВХ}(p)}$
f5	Расчет операторного выходного сопротивления цепи по схеме	$\text{Схема} \Rightarrow Z_{ВЫХ}(p)$
f6	Расчет операторного выходного сопротивления цепи по операторному входному сопротивлению и переходной характеристике	$\begin{matrix} Z_{ВХ}(p) \\ g(t) \end{matrix} \Rightarrow Z_{ВЫХ}(p)$
f7	Умножение двух ДРФ	$Z_{ВЫХ}(p) = Z_{ВХ}(p) \cdot K(p)$
f8	Расчет переходного процесса	$\begin{matrix} Z_{ВХ}(p) \\ Z_{ВЫХ}(p) \end{matrix} \Rightarrow g(t)$
f9	Интегрирование функции	$g(t) = \int h(t)dt$

Продолжение таблицы 10

Функ-ция	Суть преобразования	Описываемые взаимосвязи
f10	Дифференцирование функции	$h(t)=\frac{dg}{dt}$
f11	Обратное преобразование Фурье	$h(t)=\frac{1}{2\pi}\int_{-\infty}^{+\infty}K(j\omega)e^{j\omega\cdot t}dt$
f12	Преобразование комплексной ДРФ в операторную ДРФ	$K(p)=K(j\omega)$
f13	Прямое преобразование Фурье	$K(j\omega)=\int_{-\infty}^{+\infty}h(t)e^{-j\omega\cdot t}dt$
f14	Преобразование операторной ДРФ в комплексную ДРФ	$K(j\omega)=K(p)$
f15	Расчет комплексной ДРФ по функциям модуля и аргумента	$K(j\omega)=K(\omega)\cdot e^{j\cdot\varphi_K(\omega)}$
f16	Деление двух рядов комплексных дискретных значений с выводом комплексной ДРФ	$K(j\omega)=\frac{U_{ВЫХ}(j\omega)}{U_{ВХ}(j\omega)}$
f17	Расчет функции модуля по комплексной ДРФ	$K(\omega)=\lvert K(j\omega)\rvert$
f18	Деление двух рядов дискретных значений	$K(\omega)=\frac{U_{ВЫХ}(\omega)}{U_{ВХ}(\omega)}$
f19	Расчет функции аргумента по комплексной ДРФ	$\varphi_K(\omega)=\arg[K(j\omega)]$
f20	Вычитание двух рядов дискретных значений	$\varphi_K(\omega)=\varphi_{U_{ВЫХ}}(\omega)-\varphi_{U_{ВХ}}(\omega)$
f21	Обратное дискретное преобразование Фурье	$u_{ВХ}(t)=\frac{1}{2\pi}\int_{-\infty}^{+\infty}U_{ВХ}(j\omega)e^{j\omega\cdot t}dt$ $u_{ВЫХ}(t)=\frac{1}{2\pi}\int_{-\infty}^{+\infty}U_{ВЫХ}(j\omega)e^{j\omega\cdot t}dt$
f22	Деление ряда комплексных дискретных значений на комплексную ДРФ	$U_{ВХ}(j\omega)=\frac{U_{ВЫХ}(j\omega)}{K(j\omega)}$
f23	Прямое дискретное преобразование Фурье	$U_{ВХ}(j\omega)=\int_{-\infty}^{+\infty}u_{ВХ}(t)e^{-j\omega\cdot t}dt$ $U_{ВЫХ}(j\omega)=\int_{-\infty}^{+\infty}u_{ВЫХ}(t)e^{-j\omega\cdot t}dt$

Продолжение таблицы 10

Функ-ция	Суть преобразования	Описываемые взаимосвязи
f24	Расчет ряда комплексных дискретных значений по дискретным рядам модулей и аргументов	$U_{ВХ}(j\omega) = U_{ВХ}(\omega)\cdot e^{j\cdot\varphi_{UВХ}(\omega)}$ $U_{ВЫХ}(j\omega) = U_{ВЫХ}(\omega)\cdot e^{j\cdot\varphi_{UВЫХ}(\omega)}$
f25	Деление ряда дискретных значений на ДРФ	$U_{ВХ}(\omega) = \frac{U_{ВЫХ}(\omega)}{K(\omega)}$
f26	Расчет ряда модулей по ряду комплексных дискретных значений	$U_{ВХ}(\omega) = \lvert U_{ВХ}(j\omega)\rvert$ $U_{ВЫХ}(\omega) = \lvert U_{ВЫХ}(j\omega)\rvert$
f27	Вычитание функции из ряда дискретных значений	$\varphi_{U_{ВХ}}(\omega) = \varphi_{U_{ВЫХ}}(\omega) - \varphi_K(\omega)$
f28	Расчет ряда аргументов по ряду комплексных дискретных значений	$\varphi_{U_{ВХ}}(\omega) = \arg[U_{ВХ}(j\omega)]$ $\varphi_{U_{ВЫХ}}(\omega) = \arg[U_{ВЫХ}(j\omega)]$
f29	Интеграл Дюамеля	$u_{ВЫХ}(t) = u_{ВХ}(0)g(t) + \int_0^t \frac{du_{ВХ}}{d\tau} g(t-\tau)d\tau$
f30	Интеграл наложения	$u_{ВЫХ}(t) = \int_{-\infty}^{t} u_{ВХ}(\tau)h(t-\tau)d\tau$
f31	Умножение ряда комплексных дискретных значений на комплексную ДРФ	$U_{ВЫХ}(j\omega) = U_{ВХ}(j\omega)\cdot K(j\omega)$
f32	Умножение ряда дискретных значений на ДРФ	$U_{ВЫХ}(\omega) = U_{ВХ}(\omega)\cdot K(\omega)$
f33	Сложение двух рядов дискретных значений	$\varphi_{U_{ВЫХ}}(\omega) = \varphi_{U_{ВХ}}(\omega) + \varphi_K(\omega)$

В качестве базовых функций набора F_0 выступают элементарные функции, входящие в состав выражений из таблицы 10: интегрирование, дифференцирование, суммирование, произведение, модуль, аргумент и т.д. Символы обозначающие элементарные функции составляют алфавит базисных символов V_0.

3.5 Примеры решения прямой и обратной задач логико-математического моделирования радиотехнической системы

Приведем примеры решения задач моделирования на основе построенной базы знаний по радиотехнике линейных стационарных систем.

Пример прямой задачи моделирования

Требуется определить выходной сигнал системы по известному входному сигналу и передаточной функции.

Тогда исходные данные ***D***: $u_{ВХ}(t)$, $K(p)$; цель моделирования ***Q***: $u_{ВЫХ}(t)$,

По данным ***D*** и ***Q*** производится пополнение базы знаний, в результате чего алфавиты символов и аксиома будут иметь вид:

$$V_T = \{< \text{Сигн1} >,\ < \text{Перед} >\};$$

$$V_N = \begin{Bmatrix} < \text{Сигн2} >,\ < \text{Импул} >,\ < \text{Коэфф} >, \\ < \text{Спек1} >,\ < \text{Спек2} > \end{Bmatrix};$$

$$S = < \text{Сигн2} >.$$

В полученной полной функциональной грамматике производится полный вывод, который можно показать на дереве перебора (рисунок 19).

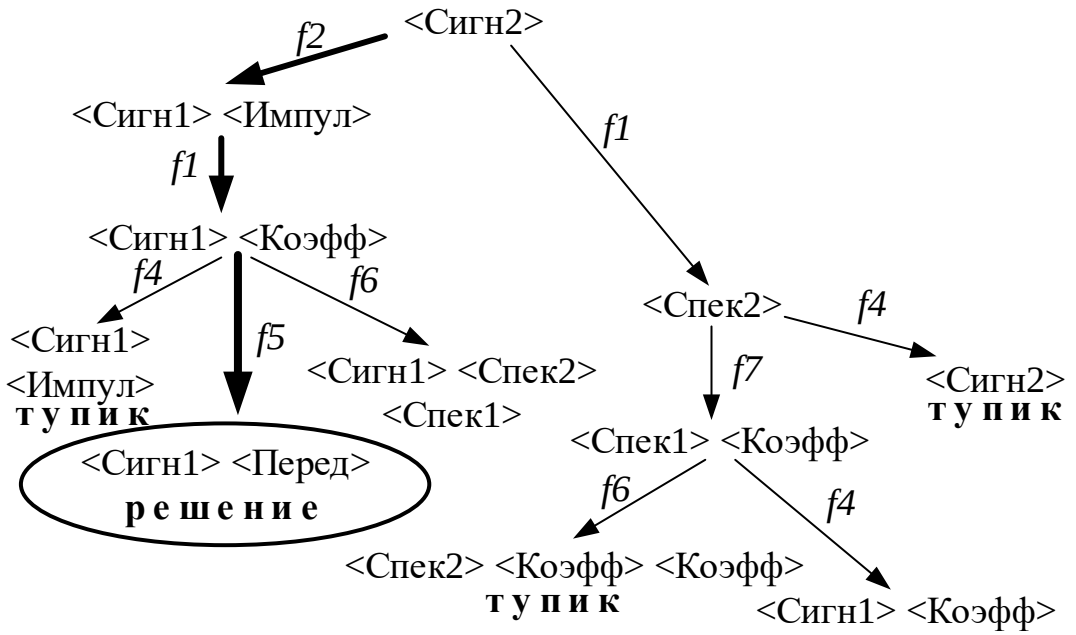

Рисунок 19 – Дерево перебора при решении прямой задачи моделирования

Результатом вывода является суперпозиция функций:

<Сигн2> = *f2*(<Сигн1>, *f1*(*f5*(<Перед>)))

Ему соответствует математическая модель:

$$u_{вых}(t) = \int_{-\infty}^{t} u_{вх}(t-\tau) \cdot \left(\frac{1}{2\pi} \int_{-\infty}^{+\infty} [K(p)]_{p \to j\omega} e^{j\omega \cdot t} dt \right) d\tau$$

Пример обратной задачи моделирования

Требуется определить передаточную функцию по известным входному и выходному сигналам.

Тогда исходные данные ***D***: $u_{ВХ}(t)$, $u_{ВЫХ}(t)$; цель моделирования ***Q***: $K(p)$.

По данным ***D*** и ***Q*** производится пополнение базы знаний, в результате чего алфавиты символов и аксиома будут иметь вид:

$$V_T = \{ < \text{Сигн1} >, \ < \text{Сигн2} > \};$$

$$V_N = \begin{Bmatrix} < \text{Перед} >, \ < \text{Импул} >, \ < \text{Коэфф} >, \\ < \text{Спек1} >, \ < \text{Спек2} > \end{Bmatrix};$$

$$S = < \text{Перед} >.$$

Дерево перебора показано на рисунке 20.

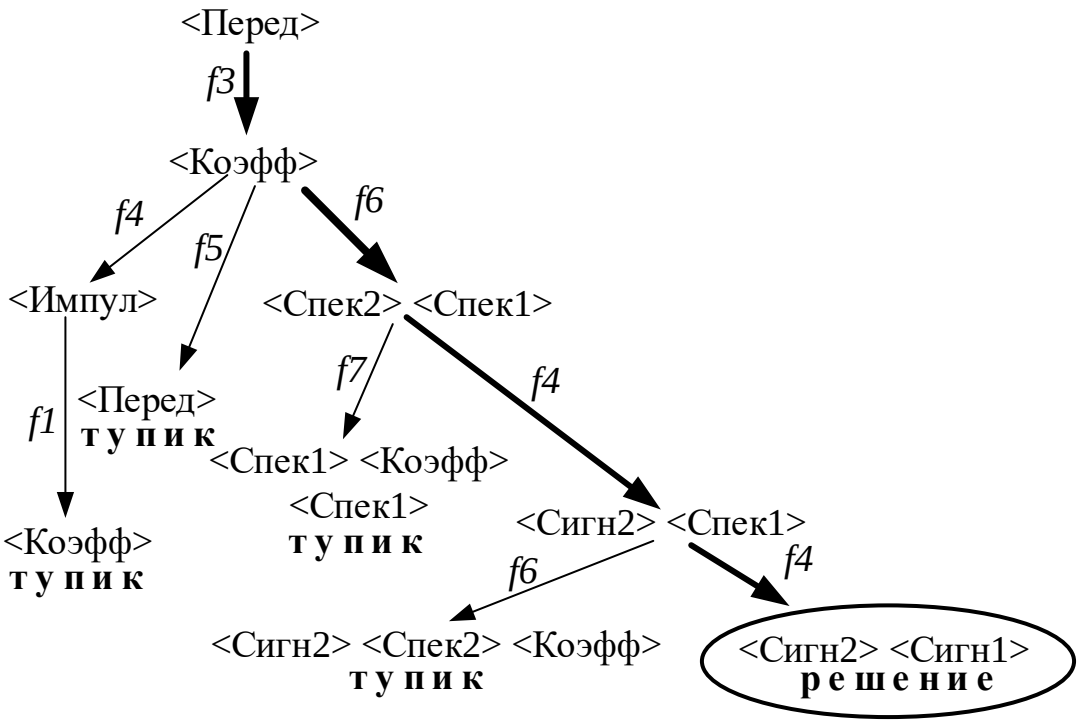

Рисунок 20– Дерево перебора при решении обратной задачи моделирования

Результатом вывода является суперпозиция функций:

<Перед> = *f3*(*f6*(*f4*(<Сигн1>, *f4*(<Сигн2>))))

Ему соответствует математическая модель:

$$K(j\omega) = \left[\frac{\int_{-\infty}^{+\infty} u_{ВЫХ}(t) e^{-j\omega \cdot t} dt;}{\int_{-\infty}^{+\infty} u_{ВХ}(t) e^{-j\omega \cdot t} dt;} \right]_{j\omega \to p}$$

Приведенные примеры показывают возможность использования полученной базы знаний для решения задач анализа и синтеза пассивных радиотехнических фильтров.

База знаний может быть расширена за счет добавления новых понятий и законов. Например, можно получить базу знаний, решающую задачи моделирования активных систем, построенных на полупроводниковых приборах.

Также базу знаний можно модифицировать для решения задач моделирования цифровых устройств. Подходы применения функциональных грамматик и λ-исчисления для моделирования цифровых схем рассматриваются в работах [110, 111].

ГЛАВА 4. АЛГОРИТМЫ ПРОГРАММНОГО КОМПЛЕКСА ЛОГИКО-МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ

4.1 Общие вопросы программной реализации

Для разработки программного алгоритма для логико-математического моделирования динамических систем необходимо определить тип программного комплекса. Он зависит от категории применяемых знаний.

Согласно С.С. Лаврову, знания подразделяются на три категории (рисунок 21): фактографические, алгоритмические и концептуальные [68].

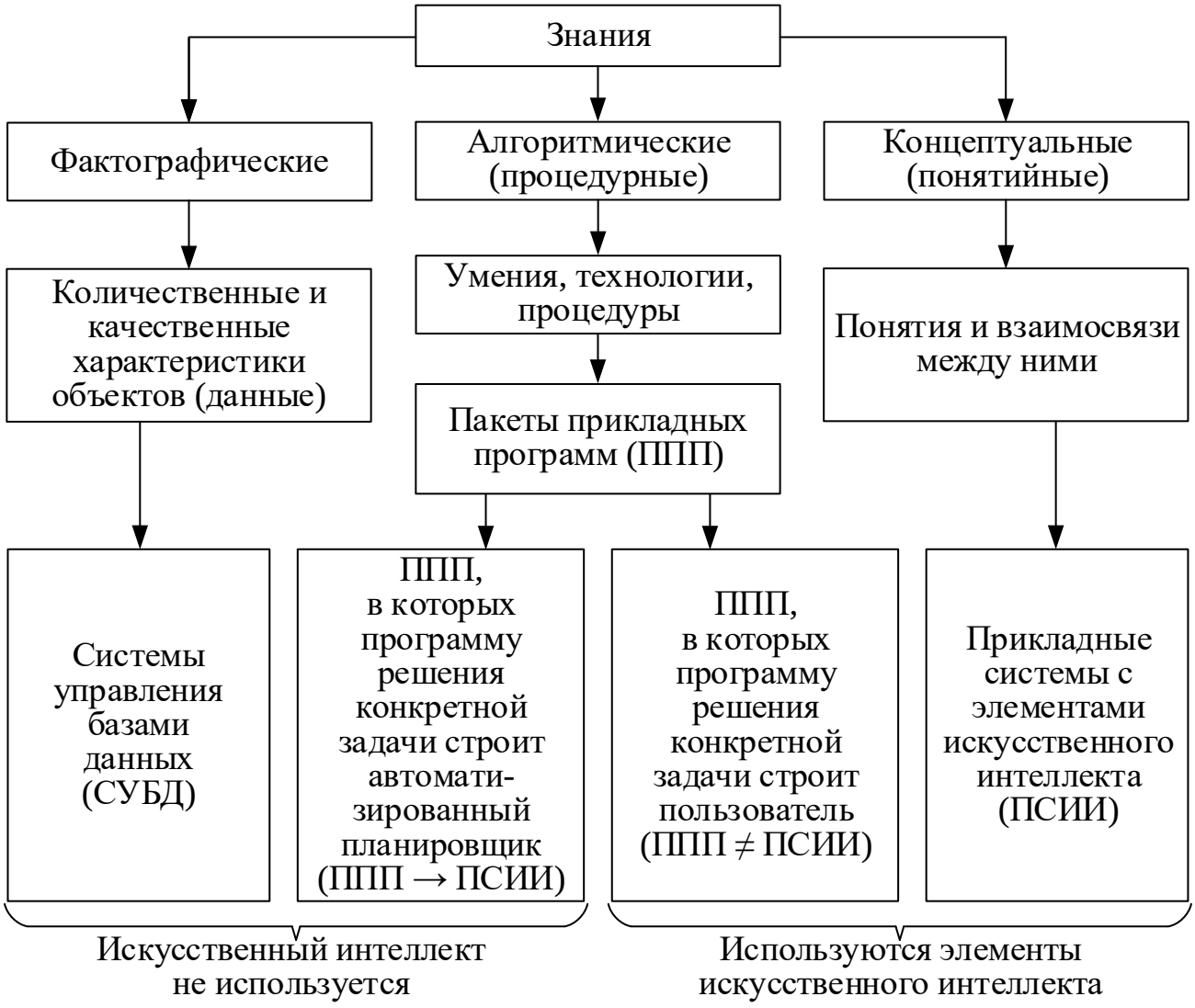

Рисунок 21 – Категории знаний и программных комплексов

Задачи, решаемые на основе фактографических знаний, вообще не требует программирования, для них достаточно иметь уже готовую систему управления базами данных (СУБД). Задачи на основе алгоритмических знаний требуют программирования для создания пакета прикладных программ (ППП) решения. Задачи на основе концептуальных знаний требуют создание прикладных систем с элементами искусственного интеллекта (ПСИИ).

Так как при концептуальном программировании решение задач строится на использовании терминов (понятий) рассматриваемой предметной области, т.е. концептуальных знаний (отсюда и название), то результат такого программирования всегда является системой с элементами искусственного интеллекта (ПСИИ). В отличие от этого, системы обычного (не концептуального) программирования могут иметь отношение к искусственному интеллекту только при определенных условиях (рисунок 21).

В логико-математическом моделировании динамических систем в качестве знаний используются понятия и взаимосвязи между ними, поэтому алгоритмы для создания программного комплекса относятся к концептуальному программированию, а сам программный комплекс – к системам с элементами искусственного интеллекта [49].

Концептуальное программирование – это способ решения задач на ЭВМ, заключающийся в описании понятий для выражения смысла задачи. В терминах этих понятий описывается задача, строится логико-математическая модель и выполняются вычисления [97].

Свойства концептуального программирования:

1) программирование в терминах предметной области решаемых задач (моделирования);

2) использование ЭВМ уже на этапе постановки задачи;

3) автоматический синтез программы решения каждой задачи (логико-математической модели);

4) использование семантической памяти (базы знаний) для накопления знаний о решаемых задачах.

Таким образом, программный комплекс логико-математического моделирования динамических систем обязательно должен иметь два модуля: базу знаний и механизм вывода решения (рисунок 22) [46].

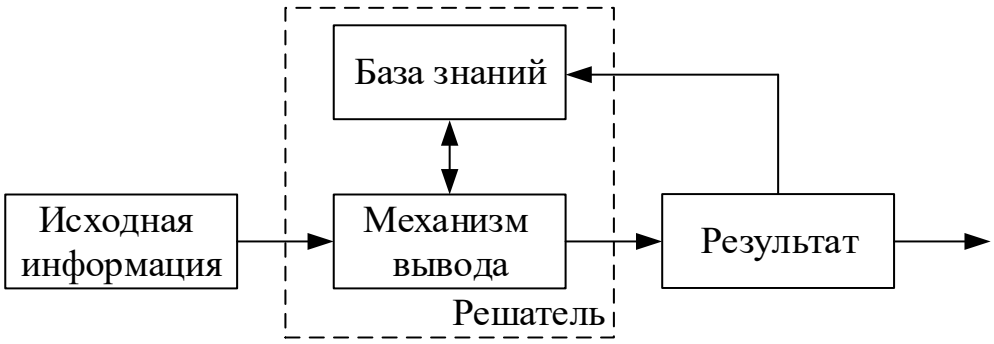

Рисунок 22 – Структурная схема программного комплекса

При использовании функциональных грамматик для составления базы знаний и реализации механизма вывода оптимальным является применение функционального языка программирования.

Так как в процессе логико-математического моделирования с использованием аппарата функциональных грамматик применение каждой продукции представляет собой вызов функции, то применение функциональное языка программирования для реализации поиска является целесообразным. Кроме того, результатом моделирования при рассмотренном методе является суперпозиция функций, что по своей сути является готовой функциональной программой.

Из всех видов функциональных языков программирования для реализации алгоритмов логико-математического моделирования динамических систем был выбран Лисп, так как он является хорошо изученным, имеет строгую теоретическую основу в виде лямбда-исчисления и оперирует списками, что в случае символьной обработки и применения численных методов является весьма удобным.

Принципы функционального программирования описаны в [11, 16, 54, 56, 74, 81, 103, 106, 107].

4.2 Программный модуль ввода базы знаний и задачи моделирования динамических систем

Алгоритм ввода базы знаний динамических систем основывается на функции *K*, записанной в 2.3 на языке чистого лямбда-исчисления, а алгоритм установки задачи моделирования – на функции *G*, записанной в 2.4.

Для ввода базы знаний и задачи моделирования на языке Лисп был разработан набор списков [42, 43, 44]:

1) список СИМВОЛЫ – одноуровневый список, первым элементом которого является начальный нетерминальный символ, остальными – все терминальные символы решаемой задачи;

2) список ПРОДУКЦИИ – четырёхуровневый список, состоящих из двух списков: ПРАВИЛА и ФУНКЦИИ;

3) список ПРАВИЛА – трёхуровневый список, состоящих из конечного множества списков ПРАВИЛО;

4) список ПРАВИЛО – двухуровневый список, первый элемент которого – атом, представляющий собой нетерминал левой части продукции, второй и последующие элементы – списки АЛЬТЕРНАТИВЫ;

5) список АЛЬТЕРАНТИВА – одноуровневый список, первый элемент которого обозначает функцию вывода правила, второй и последующие – нетерминалы правой части продукции;

6) список ФУНКЦИИ – двухуровневый список, все элементы которого являются списками ФУНКЦИЯ;

7) список ФУНКЦИЯ – одноуровневый список, состоящий из двух атомов: первый представляет собой функцию вывода, второй – количество аргументов этой функции;

8) список ДЕРЕВО – четырёхуровневый список, состоящий из конечного множества списков УЗЕЛ;

9) список УЗЕЛ – трёхуровневый список, состоящий из трёх списков: ЦЕПОЧКА, НЕТЕРМИНАЛЫ, ПУТЬ;

10) список ЦЕПОЧКА – одноуровневый список, состоящий из атомов, составляющих цепочку символов определённого узла дерева;

11) список НЕТЕРМИНАЛЫ – одноуровневый список, содержащий все нетерминальные символы, которые были подвергнуты замене в данной ветви;

12) список ПУТЬ – двухуровневый список, состоящий из конечного множества списков ВЫЗОВ;

13) список ВЫЗОВ – одноуровневый двухатомный список, первый атом которого определяет вызванную функцию вывода, а второй – номер терминала, с которого происходит вызов следующей функции.

Список ПРОДУКЦИИ состоит из двух подсписков: ФУНКЦИИ и ПРАВИЛА, которые, в свою очередь, содержат конечное число списков ФУНКЦИЯ и ПРАВИЛО соответственно.

Списки ФУНКЦИЯ являются двухатомными и содержат наименование функции, через которую выражено конкретное правило, а также количество аргументов этой функции. Через списки ФУНКЦИЯ устанавливается связь между множествами правил *P* и функций *F*.

Каждый список ПРАВИЛО начинается с символа левой части продукции и содержит в своем составе один или несколько списков АЛЬТЕРНАТИВА. Первый элемент списка АЛЬТЕРНАТИВА обозначает функцию, через которую выражается правило грамматики; остальные символы представляют собой цепочку левой части этого правила.

Для создания описанного списка ПРОДУКЦИИ и ввода функций грамматики используется трехаргументная функция «база». Первый аргумент – атом, представляющий название базы знаний и соответственно наименование глобальной переменной в виде списка ПРОДУКЦИИ, второй аргумент – список функций грамматики, третий – список правил грамматики.

Рассмотрим наполнение некоторых из приведённых списков при реализации базы знаний и задачи по физике, представленных в 2.4.

Список СИМВОЛЫ:

(<FrFric> <KinEn1> <KinEn2> <FrPull> <Dist>)

Список ПРОДУКЦИИ после ввода базы знаний:

((<Spd2> (f1 <Spd1> <Accl> <Time>) (f2 <Accl> <Dist> <Spd1>) (f11 <KinEn2> <Mass>)) (<Spd1> (f3 <Spd2> <Accl> <Time>) (f4 <Dist> <Time> <Accl>) (f5 <Spd2> <Accl> <Dist>) (f11 <KinEn1> <Mass>)) (<Dist> (f6 <Spd1> <Time> <Accl>) (f7 <Spd2> <Spd1> <Accl>) (f12 <OpPull> <FrPull>) (f12 <OpFric> <FrFric>)) (<Time> (f8 <Spd2> <Spd1> <Accl>) (f9 <Spd1> <Accl> <Dist>)) (<Accl> (f8 <Spd2> <Spd1> <Time>) (f10 <Dist> <Spd1> <Time>) (f7 <Spd2> <Spd1> <Dist>) (f8 <FrPull> <FrFric> <Mass>)) (<Mass> (f8 <FrPull> <FrFric> <Accl>) (f13 <FrFric> <CfFric> <Cfg>) (f14 <KinEn2> <Spd2>) (f14 <KinEn1> <Spd1>)) (<FrPull> (f1 <FrFric> <Mass> <Accl>) (f12 <OpPull> <Dist>)) (<FrFric> (f3 <FrPull> <Mass> <Accl) (f15 <Cfg> <CfFric> <Mass>) (f12 <OpFric> <Dist>)) (<CfFric> (f13 <FrFric> <Mass> <Cfg>)) (<KinEn2> (f16 <Mass> <Spd2>) (f17 <OpPull> <OpFric> <KinEn1>)) (<KinEn1> (f16 <Mass> <Spd1>) (f17 <KinEn2> <OpPull> <OpFric>)) (<OpPull> (f18 <FrPull> <Dist>) (f17 <KinEn2> <KinEn1> <OpFric>)) (<OpFric> (f18 <FrFric> <Dist>) (f17 <OpPull> <KinEn2> <KinEn1>)))

Список ПРОДУКЦИИ после ввода исходных данных:

((<Spd2> (f1 <Spd1> <Accl> <Time>) (f2 <Accl> <Dist> <Spd1>) (f11 <KinEn2> <Mass>)) (<Spd1> (f3 <Spd2> <Accl> <Time>) (f4 <Dist> <Time> <Accl>) (f5 <Spd2> <Accl> <Dist>) (f11 <KinEn1> <Mass>)) (<Time> (f8 <Spd2> <Spd1> <Accl>) (f9 <Spd1> <Accl> <Dist>)) (<Accl> (f8 <Spd2> <Spd1> <Time>) (f10 <Dist> <Spd1> <Time>) (f7 <Spd2> <Spd1> <Dist>) (f8 <FrPull> <FrFric> <Mass>)) (<Mass> (f8 <FrPull> <FrFric> <Accl>) (f13 <FrFric> <CfFric> <Cfg>) (f14 <KinEn2> <Spd2>) (f14 <KinEn1> <Spd1>)) (<FrFric> (f3 <FrPull> <Mass> <Accl) (f15 <Cfg> <CfFric> <Mass>) (f12 <OpFric> <Dist>)) (<CfFric> (f13 <FrFric> <Mass> <Cfg>)) (<OpPull> (f18 <FrPull> <Dist>) (f17 <KinEn2> <KinEn1> <OpFric>)) (<OpFric> (f18 <FrFric> <Dist>) (f17 <OpPull> <KinEn2> <KinEn1>)))

Для управления списками построения базы знаний и установки задачи моделирования были разработаны функции на языке Лисп. Листинг функций

представлен в Приложении А. Рассмотрим назначение каждой из этих функций:

1) Функция выполнить – выполняет все функции, являющиеся подсписками списка x.

2) Функция ввести – осуществляет ввод переменных, перечисленных в списке x.

3) Функция соед – соединяет два списка x и y в один.

4) Функция добав – добавляет к списку x элемент y.

5) Функция добав1 – добавляет элемент y к последнему элементу (обязательно подсписку) списка x.

6) Функция послед – выделяет последний элемент списка x.

7) Функция дубл – добавляет к списку x копию его последнего атома.

8) Функция обрез – удаляет из списка x последний элемент.

9) Функция начало – вырезает из списка x n-1 первых элементов.

10) Функция конец – вырезает из списка x последние элементы, начиная с элемента n.

11) Функция элемент – выбирает из списка x элемент, находящийся на позиции n.

12) Функция замена – вставляет атомы списка y вместо n-го элемента списка x.

13) Функция вставка – вставляет в список x элемент y на позицию n.

14) Функция вставка1 – вставляет в список x элемент y на последнее место n-го внутреннего списка.

15) Функция содерж – выдает значение T если в списке x содержится элемент y.

16) Функция пересеч – выдает значение T если в списке x содержится хотя бы один из элементов списка y.

17) Функция включ – выдает значение T если все элементы списка x содержится в списке y.

18) Функция располож1 – выдает номер позиции внутреннего списка в списке x, первым элементом которого является y, 0 – если элемент отсутствует.

19) Функция колич – выдает число элементов в списке x.

20) Функция печать – печатает на экран все атомы списка x, разделяя их пробелами.

21) Функция перв – выдает список из первых элементов всех подсписков списка x.

22) Функция функции – выполняет все функции, содержащиеся в списке x, и строит список всех введенных функций и количества их аргументов.

23) Функция правило – добавляет в список продукций y правило грамматики, представленное в виде списка x, с учетом числа n, указывающего на расположение нетерминала продукции x в списке y, первый элемент которого задает левую часть продукции, второй элемент – сопоставляет продукции определенную функцию, остальные элементы – задают правую часть продукции; в списке продукций происходит группировка правил с одинаковыми левыми частями.

24) Функция продукции – добавляет в список продукций набор правил грамматики, представленных в виде подсписков списка x, аналогично списку x функции правило.

25) Функция проверка1 – проверяет продукцию x на корректность использования функций, перечисленных в y; выдает nil при отсутствии ошибок, список из 1 и названия функции для её обозначения отсутствия в списке y; список из 2 и названия функции для обозначения неправильного количества аргументов.

26) Функция проверка – проверяет набор продукций x на корректность использования функций, перечисленных в y; выдает без изменений список x при отсутствии ошибок, список из 1, названия функции и номера неправильной продукции для обозначения отсутствия функции в списке y; список из 2, названия функции и номера неправильной продукции для обозначения неточного количества аргументов.

27) Функция ошибка – выдает текст ошибки в соответствии с кодом ошибки; первый элемент списка x – код ошибки; второй элемент – название ошибочной функции; третий и четвертый – координаты ошибки.

28) Функция база – строит базу знаний в виде списка из двух подсписков – функций и правил вывода; x – название базы; y – множество функций вывода; z – множество правил вывода.

4.3 Программный модуль механизма вывода логико-математической модели

Алгоритм функционирования механизма вывода основывается на лямбда-функции *M*, указанных в 2.4. Разработку алгоритма удобно осуществлять на основе построении дерева перебора [44, 48].

Представим дерево перебора в виде специального списка ДЕРЕВО. Данный список является динамическим, т.е. изменяется в ходе построения дерева за счет применения продукций и отсекания тупиковых ветвей. Списки УЗЛЫ содержат информацию о каждом узле дерева в виде подсписков ЦЕПОЧКА, НЕТЕРМИНАЛЫ и ПУТЬ. Список ЦЕПОЧКА содержит цепочку символов конкретного узла. Список НЕТЕРМИНАЛЫ необходим для реализации правила тупиковых ветвей, он содержит перечисление всех нетерминалов, которые были подвержены замене в ветви, соединяющей начальный нетерминальный символ с текущим узлом. Список ПУТЬ несет информацию о суперпозиции функций для рассматриваемого узла. Данный список состоит из двухатомных подсписков ВЫЗОВ. Первый атом списка ВЫЗОВ содержит наименование вызываемой функции, а второй – позицию вызова для следующей функции.

Построение дерева осуществляется изменением списка ДЕРЕВО. Его грамматика имеет вид:

$$G_{ДР} = \{V_{T\,ДР},\ V_{N\,ДР},\ P_{ДР},\ S_{ДР}\};$$

$$V_{T\,ДР} = \begin{Bmatrix} 0,\ 1,\ 2,\ \ldots,\ 9, \\ a,\ b,\ \ldots,\ z,\ A,\ B,\ \ldots,\ Z, \\ а,\ б,\ \ldots,\ я,\ А,\ Б,\ \ldots,\ Я, \\ (,\ ),\ <,\ >,\ \_ \end{Bmatrix};$$

$$V_{N\ ДР} = \left\{ \begin{array}{c} < \text{ДЕРЕВО} >,\ < \text{УЗЛЫ} >,\ < \text{УЗЕЛ} >,\ < \text{ЦЕПОЧКА} >, \\ < \text{ПУТЬ} >,\ < \text{ВЫЗОВ} >,\ < \text{знак\_функции} >, \\ < \text{число} >,\ < \text{символ} >, < \text{терминал} >,\ < \text{нетерминал} >, \\ < \text{слово} >,\ < \text{знак} >,\ < \text{цифра} >,\ < \text{буква} > \end{array} \right\};$$

$P_{ДР}$ = { < ДЕРЕВО > → (< УЗЛЫ >),

< УЗЛЫ > → < УЗЕЛ > | < УЗЛЫ > _ < УЗЕЛ >,

< УЗЕЛ > → ((< ЦЕПОЧКА >) _ (< НЕТЕРМИНАЛЫ >) _ (< ПУТЬ >)),

< ЦЕПОЧКА > → <символ> | < ЦЕПОЧКА > <символ>,

< НЕТЕРМИНАЛЫ > → <нетерминал> | < НЕТЕРМИНАЛЫ > <нетерминал>,

< ПУТЬ > → (< ВЫЗОВ >) | < ПУТЬ >_ (< ВЫЗОВ >),

< ВЫЗОВ > → <знак_функции> _ <число>,

<знак_функции> → f <число>,

<число> → <цифра> | <число> <цифра>,

<символ> → <терминал> | <нетерминал>,

<терминал> → < <слово> >,

<нетерминал> → < <слово> >,

<слово> → <знак> | <слово> <знак>,

<знак> → <цифра> | <буква>,

<цифра> → 0 | 1 | 2 | … | 9,

<буква> → a | b | … | z | A | B | … | Z | а | б | … | я | А | Б | … | Я

};

$S_{ДР}$ = < ДЕРЕВО > .

Список ДЕРЕВО является динамическим, т.е. изменяется в ходе построения дерева за счет применения продукций и отсекания тупиковых ветвей. Списки УЗЛЫ содержат информацию о каждом узле дерева в виде подсписков ЦЕПОЧКА, НЕТЕРМИНАЛЫ и ПУТЬ. Список ЦЕПОЧКА содержит цепочку символов конкретного узла. Список НЕТЕРМИНАЛЫ необходим для реализации правила

тупиковых ветвей, он содержит перечисление всех нетерминалов, которые были подвержены замене в ветви, соединяющей начальный нетерминальный символ с текущим узлом. Список ПУТЬ несет информацию о суперпозиции функций для рассматриваемого узла. Данный список состоит из двухатомных подсписков ВЫЗОВ. Первый атом списка ВЫЗОВ содержит наименование вызываемой функции, а второй – позицию вызова для следующей функции.

После определения цепочки, не содержащей нетерминальных символов, происходит формирование итоговой суперпозиции в префиксной лисповской нотации.

Для запуска механизма вывода используется двухаргументная функция «решение». Первый аргумент представляет собой список СИМВОЛЫ, а второй – атом, обозначающий название базы знаний, в рамках которой необходимо произвести поиск решения. Список СИМВОЛЫ состоит из начального нетерминального символа (первый атом) и терминальных символов (последующие атомы).

В 2.4 приводится пример решения задачи в рамках базы знаний «Механика» в виде дерева перебора. Рассмотрим этот же пример с точки зрения программного алгоритма. Для запуска механизма поиска решения этой задачи необходимо выполнить функцию «решение»:

(Решение '(<FrFric> <KinEn1> <KinEn2> <FrPull> <Dist>) Механика)

Механизм вывода строит дерево перебора с помощью изменения списка ДЕРЕВО. Ниже приведены этапы изменения этого списка. При этом зачеркиванием обозначено отсекание тупиковых ветвей, а выделением – узел решения.

1. (((<FrPull> <Mass> <Accl>) (<FrFric>) ((f3 2)))

((<Cfg> <CfFric> <Mass>) (<FrFric>) ((f15 2))) ((<OpFric> <Dist>) (<FrFric>) ((f12 1))))

2. (~~((<FrPull> <FrPull> <FrFric> <Accl> <Accl>) (<FrFric> <Mass>) ((f3 2) (f8 3)))~~

~~((<FrPull> <FrFric> <CfFric> <Cfg> <Accl>) (<FrFric> <Mass>) ((f3 2) (f13 2)))~~

((<FrPull> <KinEn2> <Spd2> <Accl>) (<FrFric> <Mass>) ((f3 2) (f14 3)))

((<FrPull> <KinEn1> <Spd1> <Accl>) (<FrFric> <Mass>) ((f3 2) (f14 3)))

~~((<Cfg> <FrFric> <Mass> <Cfg> <Mass>) (<FrFric> <CfFric>) ((f15 2) (f13 2)))~~

~~((<FrFric> <Dist> <Dist>) (<FrFric> <OpFric>) ((f12 1) (f18 1)))~~

((<OpPull> <KinEn2> <KinEn1> <Dist>) (<FrFric> <OpFric>) ((f12 1) (f17 1))))

3. (((<FrPull> <KinEn2> <Spd1> <Accl> <Time> <Accl>) (<FrFric> <Mass> <Spd2>)

((f3 2) (f14 3) (f1 3)))

((<FrPull> <KinEn2> <Accl> <Dist> <Spd1> <Accl>) (<FrFric> <Mass> <Spd2>)

((f3 2) (f14 3) (f2 3)))

~~((<FrPull> <KinEn2> <KinEn2> <Mass> <Accl>) (<FrFric> <Mass> <Spd2>)~~

~~((f3 2) (f14 3) (f11 4)))~~

((<FrPull> <KinEn1> <Spd2> <Accl> <Time> <Accl>) (<FrFric> <Mass> <Spd1>)

((f3 2) (f14 3) (f3 3)))

((<FrPull> <KinEn1> <Dist> <Time> <Accl> <Accl>) (<FrFric> <Mass> <Spd1>)

((f3 2) (f14 3) (f4 4)))

((<FrPull> <KinEn1> <Spd2> <Accl> <Dist> <Accl>) (<FrFric> <Mass> <Spd1>)

((f3 2) (f14 3) (f5 3)))

~~((<FrPull> <KinEn1> <KinEn1> <Mass> <Accl>) (<FrFric> <Mass> <Spd1>)~~

~~((f3 2) (f14 3) (f11 4)))~~

((<FrPull> <Dist> <KinEn2> <KinEn1> <Dist>) (<FrFric> <OpFric> <OpPull>)

((f12 1) (f17 1) (f18 0)))

~~((<KinEn2> <KinEn1> <OpFric> <KinEn2> <KinEn1> <Dist>)~~

~~(<FrFric> <OpFric> <OpPull>) ((f12 1) (f17 3))))~~

Далее на основе выделенного узла решения происходит формирование итоговой суперпозиции функций:

4. ((<FrPull> <Dist> <KinEn2> <KinEn1> <Dist>) ((f12 1) (f17 1) (f18 0)))

5. (f12 (f17 (f18 <FrPull> <Dist>) <KinEn2> <KinEn1>) <Dist>)

Так как функции *f12, f17* и *f18* введены в Лисп, то полученное выражение является программой на Лиспе. Задавая различные фактические переменные <FrPull> <Dist>, <KinEn2>, <KinEn1>, <Dist>, можно получать соответствующие числовые решения.

Таким образом, после определения цепочки, не содержащей нетерминальных символов, происходит формирование итоговой суперпозиции в префиксной лисповской нотации.

Для запуска механизма вывода используется двухаргументная функция «решение». Первый аргумент представляет собой список СИМВОЛЫ, а второй – атом, обозначающий название базы знаний, в рамках которой необходимо произвести поиск решения. Список СИМВОЛЫ состоит из начального нетерминального символа (первый атом) и терминальных символов (последующие атомы).

Программный комплекс, осуществляющий ввод базы знаний и вывод логико-математической модели, приведен в Приложении А. На него было получено Свидетельство о регистрации программы для ЭВМ [83].

Рассмотрим назначение функций, реализующие механизм вывода решения.

1) Функция сокращ – удаляет из списка x подсписки, первые элементы которых встречаются в списке y.

2) Функция добстр – добавляет к концу списка x элемент y с учетом вложенности k – количества закрывающихся скобок в конце.

3) Функция уменьш – уменьшает первый ненулевой атом списка x на единицу, предварительно удалив возможные нули из начала списка.

4) Функция формир – формирует суперпозицию функции из списка полного вывода решения x, списка применяемых функций y и списка количества аргументов функций z.

5) Функция коорд – находит самый левый нетерминал в списке x, подлежащий замене, и выдает два числа – номер найденного нетерминала в списке

x и номер его замены в списке y.

6) Функция состав – формирует из трех списков список дерева; x – сложный список выводов решения; y – сложный список использованных нетерминалов; z – сложный список использованных функций.

7) Функция стрелка – производит замену левых нетерминалов; x – список одного узла дерева, состоящий из трех подсписков: подсписка вывода решения, подсписка использованных нетерминалов; подсписка использованных функций; y – список продукций.

8) Функция стрелки – производит замену левых нетерминалов; x – список дерева, состоящий из списков узлов; y – список продукций.

9) Функция тупики – удаляет из списка дерева x тупиковые узлы.

10) Функция ствол – создает первоначальный список дерева x – аксиома; y – список продукций.

11) Функция поиск – осуществляет поиск решения, в виде узла, состоящего только из терминальных символов; x – список дерева; y – список терминальных символов.

12) Функция аргум – строит список количества аргументов всех функций, входящих в суперпозицию; x – список задействованных функций; y – список всех функций.

13) Функция уровень – осуществляет построение следующего уровня дерева перебора; x – список ствола дерева; y – список продукций; z – список использованных нетерминалов; u – список функций.

14) Функция решение – стартовая функция решателя; выдает невыполненную суперпозицию; x – список символов; аксиомы и терминалов; y – название базы знаний.

14) Функция решение1 – второй вариант стартовой функции решателя; выдает числовой результат; x – список символов; аксиомы и терминалы в виде подсписков с числовыми значениями; y – название базы знаний.

4.4 Применение суперпозиции функций в системах программирования

Результат работы любой системы знаний относится к одному из четырёх видов ответов [68]:

1) логический ответ – система подтверждает или опровергает какую-либо информацию об объекте;

2) фактографический ответ – система выдает решение задачи в виде определенного факта об объекте;

3) процедурный ответ – система синтезирует программу для ЭВМ, способную решать поставленную задачу;

4) понятийный ответ – система выдает определенный закон, по которому можно решить задачу.

При использовании аппарата функциональных грамматик результат решения задачи представляет собой суперпозицию функций $\sigma = \sigma\{f_i\}$. Из неё можно получить любой из перечисленных ответов. Согласно [94], суперпозиция функций может иметь один из трёх типов:

1) тип «знач» – выполненная суперпозиция функций, представляющая собой определенное значение;

2) тип «функ» – невыполненная, но активная суперпозиция функций, являющая функциональной программой;

3) тип «текст» – невыполненная суперпозиция функций, представляющая собой обычный текст.

Изначально, построенная суперпозиция имеет тип «текст», например:

f1(f2(a, f3(b, c), d), e, f4(g, h))

Такой результат работы системы соответствует понятийному ответу. Его суть – закон решения рассматриваемой задачи.

Если вместо аргументов a, b, c, d, e, g, h подать исходные данные задачи, то за счет выполнения функций f1, f2, f3, f4, описанных в базе знаний, будет получено значение функции. Тогда суперпозиция будет иметь тип «знач». Это соответствует

фактографическому и логическому ответам системы знаний. Конкретный вид ответа зависит от типа использованных функций.

Если же функции базы знаний f1, f2, f3, f4 транслировать в функции на интересующем языке программирования, то суперпозиция получит тип «функ». Система знаний в этом случае будет выдавать процедурный ответ, т.е. реализовывать автоматический синтез программ.

Таким образом, результатом работы механизма вывода программного комплекса логико-математического моделирования является суперпозиция функций, которая представляет универсальный результат, т.к. содержит в себе полную информацию о ходе решения задачи и может быть использована в нескольких формах.

Во-первых, суперпозиция функций может быть выполнена, т.е. вместо её формальных параметров могут быть поданы фактические параметры и получен конкретный результат заданной задачи моделирования. Во-вторых, полученная суперпозиция функций может быть выдана в текстовом виде для обозначения алгоритма (хода) решения задачи моделирования. И, наконец, суперпозиция функций может быть преобразована в программу на любом языке программирования для моделирования динамических систем с одинаковыми типами исходных и целевых знаний. В этом случае, за счет добавления еще одного модуля программный комплекс может быть преобразован в генератор программ математического моделирования на необходимом языке программирования.

Рассмотрим отличие результата работы системы логико-математического моделирования на основе функциональных грамматик от классической системы, использующей атрибутные грамматики [49].

В качестве примера, была выбрана элементарная задача, описанная в [97].

Предметная область задана соотношениями:

$$X + Y = 2{,}5; \qquad Z = X - Y;$$
$$W = Z + X \cdot Y; \qquad X = 0{,}5.$$

Необходимо построить программу, вычисляющую W.

В [97] на языке УТОПИСТ условие задачи имеет вид:

программа ВТОРАЯ;

пусть

X, Y, Z, W: вещ;

урав X+Y=2.5;

урав Z=X-Y;

урав W=Z+X*Y;

урав X=0.5;

действия

на ВТОРАЯ вычислить W;

конец

Результат, выдаваемой системой, представляет синтезированный код на языке УТОПИСТ:

X:=0.5;

Y:=2.5-X;

Z:=X-Y;

W:=Z+X*Y.

Теперь рассмотрим, как выглядит поиск решения и результат в системе, основанной на функциональных грамматиках.

База знаний «ВТОРАЯ» в виде функциональной грамматики:

$G_F = \{V_T, V_N, P, F, F_0, V_0\ S\};$ $V_T = \{X\};$ $V_N = \{Y, Z, W\};$

$P = \{$ $X \to Y \ \{f_1\};$
$X \to ZY \ \{f_2\};$
$X \to WZY \ \{f_6\};$
$Y \to X \ \{f_1\};$
$Y \to ZX \ \{f_3\};$
$Y \to WZX \ \{f_6\};$
$Z \to XY \ \{f_3\};$
$Z \to WXY \ \{f_5\};$
$W \to ZXY \ \{f_4\};$ };

$F = \{$ $f_1(x,y) = f_{03}(f_{05}(),y);;$
$f_2(x,y) = f_{01}(x,y);$
$f_3(x,y) = f_{03}(x,y);$
$f_4(x,y,z) = f_{01}(x, f_{02}(y,z));$
$f_5(x,y,z) = f_{03}(x, f_{02}(y,z));$
$f_6(x,y,z) = f_{04}(f_{03}(x,y),z)$ };

$F_0 = \{$ $f_{01}(x,y) = x + y;$
$f_{02}(x,y) = x \cdot y;$
$f_{03}(x,y) = x - y;$
$f_{04}(x,y,z) = x / y;$
$f_{05}() = 2{,}5;$ };

$V_0 = \{ < + >, < \cdot >, < - >, < / >, < 2{,}5 > \};$

$S = \{W\}.$

Поиск решения осуществляется в разработанном программном комплексе на языке Лисп [83].

Команда (Решение ‘(W X) ВТОРАЯ) запускает процесс решения задачи. На рисунке 23 представлен алгоритм поиска решения в виде дерева перебора.

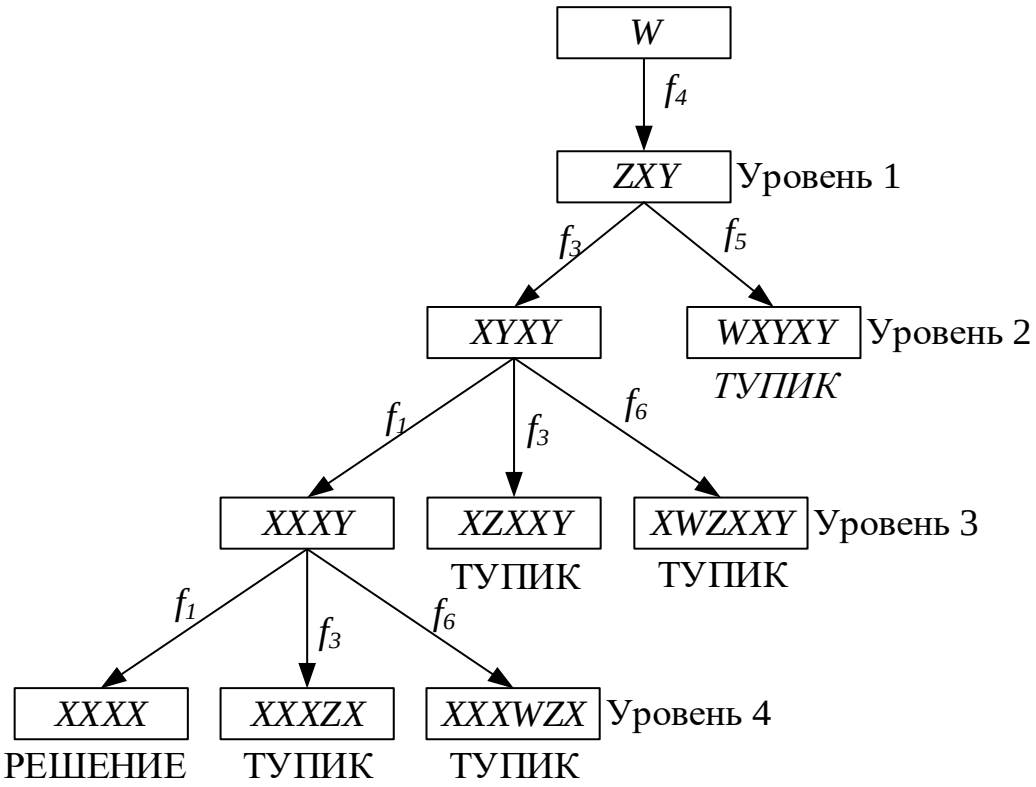

Рисунок 23 – Дерево перебора при решении задачи

Таким образом, решение представляет ветвь:

$$W \xrightarrow{f_4} ZXY \xrightarrow{f_3} XYXY \xrightarrow{f_1} XXXY \xrightarrow{f_1} XXXX$$

Данному результату соответствует суперпозиция функций:

$$W = f_4(\ (f_3(X,\ f_1(X)),\ X,\ f_1(X)\ ).$$

Такое решение является более предпочтительным, чем код, выдаваемый на языке УТОПИСТ. Суперпозиция функций может принять форму кода на любом языке программирования, в котором существует аппарат функций.

Так, например, если необходима программа на языке Си, то нужно закодировать функции множества *F* следующим образом:

```
float f1(float x) {
return (2.5-x); };
float f3(float x, float y) {
return (x-y); };
float f4(float x, float y, float z) {
return (x+y*z); };
```

Тогда программа на языке Си будет иметь вид:

```
int main { float W;
W = f4(  (f3 (0.5,  f1(0.5) ),  0.5,  f1(0.5)  )
return(0);  }
```

Аналогичным образом можно закодировать функции на другом языке программирования и записать программу в виде суперпозиции этих функций.

Таким образом, использование функциональных грамматик в концептуальном программировании является более универсальным, чем использование атрибутных грамматик.

Во-первых, функциональные грамматики более адекватно описывают базу знаний и модель вывода решения, что при правильном использовании может увеличить эффективность поиска решения задачи.

Во-вторых, результат концептуального программирования в виде суперпозиции функций позволяет находить решение задачи как непосредственно (логический или фактографический ответы, тип суперпозиции «знач»), так и в виде синтезированной программы для ЭВМ на любом выбранном языке программирования (процедурный ответ, тип «функ»). Кроме того, при добавлении лингвистического модуля решение задачи может быть представлено в виде текстовых рекомендаций для пользователя (понятийный ответ, тип суперпозиции «текст»).

4.5 Применение символьных и численных методов при программировании базы знаний по радиотехнике

Программирование базы знаний по радиотехнике линйных стационарных систем, разработанной в Главе 3, помимо арифметических операций основывается на применении численных методов и методов символьной обработки информации. В таблице 11 приведены перечень использованных методов.

Таблица 11 – Методы программирования функций базы знаний V_{PT}

Группа методов	Метод	Функции применения
Символьные методы	Упрощение полиномов	*f1*, *f2*, *f3*, *f4*, *f6*, *f14*
	Умножение полиномов	*f7*
	Деление полиномов	*f4*, *f8*
	Преобразование полиномов	*f1*, *f2*, *f3*, *f6*, *f7*, *f8*
	Символьная замена переменных	*f3*, *f6*, *f8*, *f12*, *f14*
	Аналитическое интегрирование	*f9*, *f3*
	Аналитическое дифференцирование	*f10*
	Поэлементное операции с числовыми рядами	*f15*, *f16*, *f17*, *f4*, *f18*, *f19*, *f20*, *f22*, *f24*, *f25*, *f27*, *f31*, *f33*
Численные методы	Численное дифференцирование	*f29*
	Численное интегрирование методом Симпсона	*f29*, *f30*
	Решение нелинейных операторных уравнений методом Лобачевского	*f8*
	Быстрое преобразование Фурье	*f23*
	Обратное быстрое преобразование Фурье	*f11*

Использованные при программировании численные методы описаны в [10, 19, 29, 30, 82, 109]. Фрагмент кода базы знаний приведен в Приложении Б.

ЗАКЛЮЧЕНИЕ

В диссертационной работе получены следующие **основные научные результаты**:

1. Разработан последовательно-параллельный способ декомпозиции оператора логического вывода для решения задачи моделирования динамической системы, позволяющий использовать функциональную парадигму и параллельные вычисления в логико-математическом моделировании.

2. Впервые разработан логико-математический метод решения прямых и обратных задач моделирования динамических систем на основе аппарата функциональных контекстно-свободных грамматик.

3. Построена база знаний в виде неполной функциональной контекстно-свободной грамматики, предназначенная для логико-математического моделирования линейных стационарных радиотехнических систем.

4. Получены результаты решения прямой и обратной задач моделирования радиотехнических динамических систем, представляющих собой линейные стационарные пассивные электрические цепи в виде *Г*-образных четырехполюсников, преобразующих входной дискретный радиотехнический сигнал в выходной дискретный сигнал.

5. Разработан алгоритм для построения программного комплекса логико-математического моделирования динамических систем на основе функциональных грамматик.

СПИСОК ЛИТЕРАТУРЫ

1. Абельсон Х. Структура и интерпретация компьютерных программ [Текст] / Х. Абельсон, Дж. Сассман. – М. : Добросвет. – 2006. – 608 с.

2. Аршинский, Л.В. Применение векторного формализма в логике и логико-математчиеском моделировании [Текст] / Л.В. Аршинский // Онтология проектирования. – 2016. – Т. 6. – №4 (22). – С. 436-451.

3. Аршинский, Л.В. Построение агрегированных оценок систем как логический вывод на алгебраических сетях [Текст] / Л.В. Аршинский, Е.А. Асламова, А.Ю. Попов // Информационные технологии и проблемы математического моделирования сложных систем. – 2016. – №15. – С. 23-30.

4. Аршинский, Л.В. Учет компонентов структуры и компонентов влияния при логико-аксиологической оценке систем [Текст] / Л.В. Аршинский // Информационные и математические технологии в науке и управлении. – 2016. – №2. – С. 18-29.

5. Аршинский, Л.В. Учет компонентов структуры и компонентов влияния при логико-аксиологической оценке систем [Текст] / Л.В. Аршинский // Информационные и математические технологии в науке и управлении. – 2016. – №2. – С. 18-29.

6. Ахо, А. Теория синтаксического анализа, перевода и компиляции [Текст] : в 2-х т. / А. Ахо, Дж. Ульман ; пер. с англ. В.Н. Агафонова. – М. : Книга по требованию, 2012. – 613 с. – т. 1. Синтаксический анализ.

7. Бадмаев, Б.Б. Функциональное моделирование работы систем знаний [Текст] / Б.Б. Бадмаев, В.А. Кравченко, Д.Д. Чимитова // Международный журнал прикладных и фундаментальных исследований. – 2011. – №11. – С. 41-42.

8. Барендрегт, Х. Лямбда-исчисление. Его синтаксис и семантика [Текст] / Х. Барендрегт ; пер. с англ. Г.Е. Минца ; под ред. А.С. Кузичева. – М. : Мир, 1985. – 606 с.

9. Баскаков, С.И. Радиотехнические цепи и сигналы [Текст] / С.И. Баскаков.

– М. : Высшая школа, 2016. – 528 с.

10. Беланов, А.А. Решение алгебраических уравнений методом Лобачевского [Текст] / А.А. Беланов. – М. : Наука, 1989. – 96 с.

11. Большакова, Е.И. Основы программирования на языке Лисп [Текст] / Е.И. Большакова, Н.В. Груздева. – М. : МАКС Пресс, 2010. – 112 с.

12. Борисов, Ю.П. Математическое моделирование радиотехнических систем и устройств [Текст] / Ю.П. Борисов, В.В. Цветнов. – М. : Радио и связь, 1985. – 176 с.

13. Бычков И.В. Методы и средства организации параллельных и распределенных вычислений на основе парадигмы модульного программирования / И.В. Бычков, Г.А. Опарин, А.П. Новопашин, И.А. Сидоров, С.А. Горский // Вестник Кемеровского государственного университета. – 2012. – №4-2 (52). – С. 22-30.

14. Волкова, И.А. Формальные грамматики и языки. Элементы теории трансляции [Текст] / И.А. Волкова, А.А. Вылиток, Т.В. Руденко. – М. : Изд. отд. ВМиК МГУ, 2009. – 115 с.

15. Вольфенгаген, В.Э. Аппликативные вычислительные системы и концептуальный метод проектирования систем знаний [Текст] / В.Э. Вольфенгаген, В.Я. Яцук ; под ред. Л.А. Майбороды. – М.: МО СССР, 1987. – 255 с.

16. Вольфенгаген В.Э. Парадигма функционального программирования [Текст] / В.Э. Вольфенгаген, Л.Ю. Исмаилова, С.В. Косиков. –М. : АО «Центр ЮрИнфоР», 2012. – 96 с.

17. Вольфенгаген, В.Э. Концептуальный язык управления представлениями информации для обеспечения программирования по примерам [Текст] / В.Э. Вольфенгаген, Л.Ю. Исмаилова, С.В. Косиков // Наука и образование в жизни общества: сборник научных трудов по материалам международной научно-практической (г. Тамбов, 29 ноября 2013 г.). Часть 15. – Тамбов: ООО «Консалтинговая компания Юком», 2013. – С. 29-30

18. Гаврилова, Т.А. Базы знаний интеллектуальных систем [Текст] / Т.А. Гаврилова, В.Ф. Хорошевский. – СПб. : Питер, 2000. – 384 с.

19. Галанин, М.П. Методы численного анализа математических моделей [Текст] / М.П. Галанин, Е.Б. Савенков. – М.: МГТУ им. Н.Э. Баумана, 2010. – 590 с.

20. Грищенко, В.Н. Сборочное программирование. Основы индустрии программных продуктов [Текст] / В.Н. Грищенко, Е.М. Лаврищева. – Киев : Наук. думка, 2009. – 372 с.

21. Гросс, М. Теория формальных грамматик [Текст] / М. Гросс, А. Лантен ; пер. с фр. И.А. Мельчука ; под ред. А.В. Гладкого. – М. : Мир, 1971. – 296 с.

22. Девятков, В.В. Системы искусственного интеллекта [Текст] : учеб. пособие для вузов / В.В. Девятков. – М. : Изд-во МГТУ им Н.Э. Баумана, 2001. – 352 с.

23. Денисов, А.М. Введение в теорию обратных задач [Текст] / А.М. Денисов. – М. : Изд-во МГУ, 1994. – 208 с.

24. Ильин, А.В. Символьное моделирование в информатике [Текст] / А.В. Ильин, В.Д. Ильин. – М. : Институт проблем и информатики РАН, 2011. – 204 с.

25. Ильин, А.В. S-моделирование задач и конструирование программ [Текст] /А.В. Ильин, В.Д. Ильин. – М. : Институт проблем и информатики РАН, 2012. – 146 с.

26. Ильин, А.В. Систематизация знаний о программируемых задачах [Текст] / А.В. Ильин, В.Д. Ильин // Системы и средства информатики. – 2014. – Т.24. – №3. – С.192-203.

27. Ильин, А.В. Создание человеко-машинной среды решения задач [Текст] / А.В. Ильин, В.Д. Ильин // Системы и средства информатики. – 2016. – Т.26. – № 4. – С.149-161.

28. Исмаилова, Л.Ю. Специализация концептуальных моделей на основе определённых дескрипций [Электронный ресурс] / Л.Ю. Исмаилова, С.В. Косиков // Современные проблемы науки и образования. – 2013. – № 3. – Режим доступа: https://science-education.ru/ru/article/view?id=9387.

29. Калиткин, Н.Н. Численные методы. Книга 1 (Численный анализ) [Текст] / Н.Н. Калиткин, Е.А. Альшина. – М.: Издательский центр «Академия», 2013. –

299 с.

30. Калиткин, Н.Н. Численные методы. Книга 2 (Методы математической физики) [Текст] / Н.Н. Калиткин, П.В. Корякин. – М.: Издательский центр «Академия», 2013. – 303 с.

31. Кахро, М.И. Инструментальная система программирования ЕС ЭВМ (ПРИЗ) [Текст] / М.И. Кахро, А.П. Калья, Э.Х. Тыугу. – М. : Финансы и статистика, 1988. – 181 с.

32. Козлов, О.С. Построение математических моделей электрических цепей в программных комплексах структурного моделирования [Электронный ресурс] / О.С. Козловский, Л.М. Скворцов // Инженерный вестник. – 2012. – №7. – Режим доступа: https://science-education.ru/ru/article/view?id=9387.

33. Корухова, Ю.С. Система дедуктивного синтеза программ / Ю.С. Корухова, В.Н. Пильщиков // Интеллектуализация обработки информации: материалы Международной научной конференции – Симферополь, 2002. – С. 124-125.

34. Корухова, Ю.С. Система дедуктивного синтеза программ [Текст] / Ю.С. Корюхова, В.Н. Пильщиков // Научно-теоретический журнал «Искусственный интеллект». – 2002. – № 2. – С. 451-459.

35. Корухова, Ю.С. Дедуктивный синтез программ с использованием волновых правил [Текст] / Ю.С. Корухова // Программные системы и инструменты. – М. : Изд. отд. фак. ВМК МГУ, 2005. – С. 6-16.

36. Корухова, Ю.С. Система автоматического синтеза функциональных программ [Текст] : дис. … канд. физ.-мат. наук: 05.13.11 / Ю.С. Корухова. – М., 2005. – 124 с.

37. Корухова, Ю. С. Автоматический синтез программ [Текст] / Ю.С. Корухова // Программы специальных курсов факультета ВМК МГУ. – М. : МАКС Пресс, 2010. – С. 185–187.

38. Кравченко, В.А. Представление знаний в функциональных грамматиках [Текст] / В.А. Кравченко, П.Б. Могнонов, Д.Н. Чимитов // Вестник Сибирского государственного аэрокосмического университета им. академика М.Ф. Решетнева.

– 2011. – №5 (38). – С. 55-61.

39. Кравченко, В.А. Решение задач посредством функциональных грамматик [Текст] / В.А. Кравченко // Теоретические и прикладные вопросы современных информационных технологий: материалы XI Всероссийской научно-технической конференции (г. Улан-Удэ, 13-20 августа 2012). – Улан-Удэ : Изд-во ВСГУТУ, 2012. – С. 399-402.

40. Кравченко, В.А. Моделирование поиска решения с помощью функциональных грамматик [Текст] / В.А. Кравченко // Вестник Бурятского государственного университета. – 2012. – №9. – С. 33-41.

41. Кравченко, В.А. Применение решателя задач на основе аппарата функциональных грамматик в радиотехнике [Текст] / В.А. Кравченко, Д.Н. Чимитов // Компьютерные технологии в науке, в технике, в искусстве: материалы Всероссийской научной конференции, часть 2 (Таганрог, 15 июня 2013 г.). – Таганрог : Изд-во ТТИ ЮФУ, 2013. – С. 17-23.

42. Кравченко, В.А. Решатель задач на основе аппарата функциональных грамматик [Текст] / В.А. Кравченко, П.Б. Могнонов // Компьютерные технологии в науке, в технике, в искусстве: материалы Всероссийской научной конференции, часть 2 (Таганрог, 15 июня 2013 г.). – Таганрог : Изд-во ТТИ ЮФУ, 2013. – С. 11-17.

43. Кравченко, В.А. Программная реализация решателя задач на основе метода функциональных грамматик [Текст] / В.А. Кравченко, П.Б. Могнонов, Д.Н. Чимитов // Вестник Восточно-Сибирского государственного университета технологий и управления. – 2013. – №6. – С. 36-42.

44. Кравченко, В.А. Автоматизированный поиск решения задач методом функциональных грамматик [Текст] / В.А. Кравченко // Современные системы искусственного интеллекта и их приложения в науке: материалы II Всероссийской научной Интернет-конференции с международным участием (г. Казань, 14 мая 2014 г.). – Казань : ИП Синяев Д.Н., 2014. – С. 48-50.

45. Кравченко, В.А. Моделирование решения технических задач аппаратом функциональных грамматик [Текст] / В.А. Кравченко // Дифференциальные

уравнения и математическое моделирование: материалы конференции (г. Улан-Удэ, 22-27 июня 2015 г.). – Улан-Удэ : Изд-во ВСГУТУ, 2015. – С. 165-167.

46. Кравченко, В.А. Системы знаний на основе функциональных грамматик [Текст] / В.А. Кравченко, Д.Ш. Ширапов // Инновации на основе информационных и коммуникационных технологий: материалы XII Международной научно-практической конференции (г. Сочи, 1-10 октября 2015 г.). – М. :НИУ ВШЭ, 2015. – С. 178-179.

47. Кравченко, В.А. Построение баз знаний для решения задач методом функциональных грамматик [Текст] / В.А. Кравченко, Д.Ш. Ширапов // Вестник Бурятского государственного университета. – 2015. – №9 «Математика и информатика». – С. 96-102.

48. Кравченко, В.А. Автоматизированный синтез программ в интеллектуальных системах на основе функциональных грамматик [Текст] / В.А. Кравченко, Д.Н. Чимитов // Инновационные, информационные и коммуникационные технологии: сборник трудов XIII Международной научно-практической конференции (г. Сочи, 1-10 октября 2016 г.). – М. : Ассоциация вып. и сотр. ВВИА им. проф. Жуковского, 2016. – С. 275-277.

49. Кравченко, В.А. Использование функциональных грамматик в концептуальном программировании [Текст] / В.А. Кравченко, Д.Ш. Ширапов, С.И. Олзоева // Вестник Бурятского государственного университета. Математика и информатика. – 2016. – №4. – С. 3-12.

50. Кравченко, В.А. Функциональное логико-математическое моделирование динамических систем [Текст] / В.А. Кравченко, Д.Ш. Ширапов, Д.Н. Чимитов // Техника и технологии: новые перспективы развития: сборник статей Международной научно-практической конференции (г. Самара, 25 сентября 2017 г.). – Уфа: АЭТЕРНА, 2017. – С. 31-37.

51. Кравченко, В.А. Логико-математическое моделирование на основе функциональных грамматик [Текст] / В.А. Кравченко, Д.Ш. Ширапов, Д.Н. Чимитов // Техника и технологии: новые перспективы развития: сборник статей Международной научно-практической конференции (г. Самара, 25 сентября

2017 г.). – Уфа: АЭТЕРНА, 2017. – С. 37-42.

52. Кравченко, В.А. Способ последовательно-параллельной декомпозиции при логико-математическом моделировании динамических систем [Текст] / В.А. Кравченко, Д.Ш. Ширапов, Д.Н. Чимитов / В.А. Кравченко, Д.Ш. Ширапов, Д.Н. Чимитов // Современные технологии в мировом научном пространстве: сборник статей Международной научно-практической конференции (г. Уфа, 28 сентября 2017 г.): В 3 ч. Ч.2 – Уфа : АЭТЕРНА, 2017. – С. 48-56.

53. Кравченко, В.А. Метод логико-математического моделирования на основе функциональных грамматик [Текст] / В.А. Кравченко, Д.Ш. Ширапов, Д.Н. Чимитов // Инновационные, информационные и коммуникационные технологии: сборник трудов XIV Международной научно-практической конференции (г. Сочи, 1-10 октября 2017 г.). – М. : Ассоциация вып. и сотр. ВВИА им. проф. Жуковского, 2017. – С. 389-392.

54. Кубенский, А.А. Функциональное программирование [Текст] / А.А. Кубенский. – М. : Изд-во Юрайт, 2017. – 348 с.

55. Лаврищева, Е.М. Теория объектно-компонентного моделирования [Текст] / Е.М. Лаврищева. – М.: ИСП РАН, 2016. – 52 с.

56. Лавров, С.С. Автоматическая обработка данных. Язык Лисп и его реализация [Текст] / С.С. Лавров, Г.С. Силагадзе – М. : Наука, 1978. – 176 с.

57. Лавров, С.С. СПОРА – система программирования с автоматическим синтезом программ [Текст] / И.О. Бабаев, С.С. Лавров, Г.А. Нецветаева, Ф.А. Новиков, Г.М. Шувалов / Применение методов математической логики: тезисы докладов III конференции. – Таллин, 1983. – С. 29 – 41.

58. Лавров, С.С. Программирование. Математические основы, средства, теория [Текст] / С.С. Лавров. – СПб. : БХВ-Петербург, 2002. – 320 с.

59. Ларионов, А.А. Параллельные схемы алгоритмов автоматического доказательства теорем в исчислении позитивно-образованных формул [Текст] / А.А. Ларионов, Е.А. Черкашин // Дистанционное и виртуальное обучение. – 2012. – №2. – С. 93-100.

60. Лебедев, А.С. Разработка и исследование системы концептуального

программирования с использованием лингвистического процессора [Текст] : дис. … канд. техн. наук: 05.13.11 / А.С. Лебедев. – М. , 2011. – 178 с.

61. Лукичев, А.С. Использование атрибутов в технологии SYNTAX [Текст] / А.С. Лукичев // Вестник Петербургского университета. Серия 1. – 2005. – Вып. 2. – С. 64-73.

62. Люгер, Дж. Ф. Искусственный интеллект: стратегии и методы решения сложных проблем [Текст] / Дж. Ф. Люгер ; пер. с англ. Н.И. Галагана и др.; под ред. Н.Н. Куссуль. – М. : Изд. дом «Вильямс», 2003. – 864 с.

63. Мозговой, М.В. Машинный семантический анализ русского языка и его применения [Текст] : дис. … канд. физ.-мат. наук: 05.13.11 / М.В. Мозговой. – СПб., 2006. – 116 с.

64. Мозговой, М.В. Простая вопросно-ответная система на основе семантического анализатора русского языка [Текст] / М.В. Мозговой // Вестник СПб университета. – 2006. – сер. 10. – вып. 1. – С. 116-122.

65. Монаков, А.А. Основы математического моделирования радиотехнических систем [Текст] / А.А. Монаков. – СПб. : ГУАП, 2005. – 100 с.

66. Нейлор, К. Как построить свою экспертную систему [Текст] / К. Нейлор ; пер. с англ. Н.Н. Слепова. – М. : Энергоатомиздат, 1991. – 286 с.

67. Нильсон, Н. Искусственный интеллект: методы поиска решений [Текст] / Н. Нильсон ; пер. с англ. В.Л. Стефанюка ; под ред. С.В. Фомина. – М. : Мир, 1973. – 272 с.

68. Новиков, Ф.А. Искусственный интеллект: представление знаний и методы поиска решений [Текст] / Ф.А. Новиков. – СПб.: Изд-во Политехн. ун-та, 2010. – 240 с.

69. Новопашин, А.П. Инструментальные средства организации параллельных вычислений в пакетах прикладных программ [Текст] / А.П. Новопашин, И.А. Сидоров, С.А. Горский // Параллельные вычислительные технологии: труды Международной научной конференции (г. Москва, 28 марта – 01 апреля 2011 г.). – Челябинск: Изд. центр ЮУрГУ, 2011. – С. 244-253.

70. Новосельцев, В.Б. Теория структурных функциональных моделей [Текст]

/ В.Б. Новосельцев // Сибирский математический журнал. – 2006. – №.6. – С. 1342-1354.

71. Новосельцев, В.Б. Реализация эффективного алгоритма синтеза линейных функциональных программ [Текст] / В.Б. Новосельцев, А.Е. Пинжин // Известия Томского политехнического университета. – 2008. – Т.312 – №.5. – С. 32-35.

72. Опарин, Г.А. Язык описания модели предметной области в пакетах прикладных программ [Текст] / Г.А. Опарин, А.Г. Феоктистов, С.А. Горский // Программные продукты и системы. – 2011. – №1. – С. 34-36.

73. Пинжин, А.Е. Эффективный алгоритм синтеза программ с условиями и подпрограммами [Текст] / В.Б. Новосельцев, А.Е. Пинжин // Известия Томского политехнического университета. – 2008. – Т.313 – №.5. – С. 77-84.

74. Пирс, Б. Типы в языках программирования [Текст] / Б. Пирс ; пер. с англ. Г. Бронникова, А. Отта ; под. ред. А. Махоткина. – М. : Изд-во «Лямбда пресс» : «Добросвет», 2011. – 656 с.

75. Пискорский, Д.С. Особенности математического моделирования сложных радиотехнических систем (РТС) [Текст] / Д.С. Пискорский, Н.В. Вдовина // Вестник Южно-Уральского государственного университета. Серия: Компьютерные технологии, управление, радиоэлектроника. – 2013. – Том 13. – №3. – С. 145-149.

76. Подколзин, А.С. Компьютерное моделирование логических процессов. Архитектура и языки решателя задач [Текст] / А.С. Подколзин – М. ФИЗМАТЛИТ, 2008. – 1024 с.

77. Попов, В.П. Основы теории цепей [Текст] / В.П. Попов. – М.: Юрайт, 2015. – 696 с.

78. Попов, Э.В. Экспертные системы: Решение неформализованных задач в диалоге с ЭВМ [Текст] / Э.В. Попов. – М.: Наука, 1987 – 288 с.

79. Путилов, Г.П. Система естественно-языкового концептуального программирования Nalaps [Текст] / Megaling 2009: материалы Международной научной конференции (г. Киев, 21-26 сентября 2009 г.). – Киев, 2009. – С.65-66.

80. Рассел, С. Искусственный интеллект: современный подход [Текст] / С. Рассел, П. Норвиг ; пер. с англ. и ред. К.А. Птицыной. – М. : Изд. дом «Вильямс», 2017. – 1408 с.

81. Роджерс Х. Теория рекурсивных функций и эффективная вычислимость [Текст] / Х. Роджерс ; пер. с англ. В.А. Душевского и др. ; под ред. В.А. Успенского. – М. : Мир, 1972. – 624 с.

82. Романовский, П.И. Ряды Фурье. Теория поля. Аналитические и специальные функции. Преобразование Лапласа [Текст] / П.И. Романовский. – М.: Наука, 1973. – 336 с.

83. Свидетельство № 2013613951 Российская Федерация. Решатель задач на основе метода использования функциональных грамматик : свидетельство об офиц. регистрации программы для ЭВМ / В. А. Кравченко ; заявитель и правообладатель ФГБОУ ВПО «ВСГУТУ». – № 2013612012 ; заявл. 12.03.2013 ; зарегистрировано в реестре программ для ЭВМ 19.04.2013.

84. Серебряков, В.А. Основы конструирования компиляторов [Текст] / В.А. Серебряков, М.П. Галочкин. – М. : Изд-во URSS. – 2001. – 174 с.

85. Сергиенко, А.Б. Цифровая обработка сигналов [Текст] / А.Б. Сергиенко. – Спб. : БХВ-Петербург, 2015. – 768 с.

86. Смолин, Д.В. Введение в искусственный интеллект [Текст] / Д.В. Смолин. – М.: ФИЗМАТЛИТ, 2007. – 264 с.

87. Советов, Б. Я. Моделирование систем [Текст] / Б.Я. Советов, С.А. Яковлев. – М. : Юрайт, 2016. – 344 с.

88. Страницы истории отечественных ИТ [Текст] / сост. Э.М. Пройдаков. – М. : Альпина Паблишер, 2015. Т.1 – 2015. – 265 с.

89. Тарасик, В.П. Математическое моделирование технических систем [Текст] / В.П. Тарасик. – М : НИЦ ИНФРА-М, 2016. – 592 с.

90. Таунсенд, К. Проектирование и программная реализация экспертных систем на персональных ЭВМ [Текст] / К. Таунсенд, Д. Фохт ; пер. с англ. Г.С. Осипова. – М. : Финансы и статистика, 1990. – 320 с.

91. Тей, А. Логический подход к искусственному интеллекту: от

классической логики к логическому программированию [Текст] / А. Тей и др. ; пер. с англ. П.П. Пермякова. – М. : Мир, 1990. – 432 с.

92. Трофимова, Т.И. Курс физики [Текст] / Т.И. Трофимова. – М. : Высшая школа, 2016. – 516 с.

93. Тузов, В.А. Функциональные грамматики [Текст] / В.А. Тузов // Численные методы и вопросы организации вычислений. 4, Зап. научн. сем. ЛОМИ, том 102. – 1980. – С. 123-137.

94. Тузов, В.А. Математическая модель языка [Текст] / В.А. Тузов. – Л. : Изд-во Ленингр. ун-та, 1984. – 176 с.

95. Тузов, В.А. Языки представления знаний [Текст] / В.А. Тузов. – Л. : Изд-во Ленингр. ун-та, 1990. – 120 с.

96. Тузов, В.А. Компьютерная семантика русского языка [Текст] / В.А. Тузов. – СПб. : Изд-во СПбГУ, 2003. – 391 с.

97. Тыугу, Э.Х. Концептуальное программирование [Текст] / Э.Х. Тыугу. – М. : Наука, 1984. – 256 с.

98. Федорченко, Л.Н. О регуляризации контекстно-свободных грамматик [Текст] / Л.Н. Федорченко // Известия вузов. Приборостроение. – 2006. – Т.49. – №11. – С. 50-54.

99. Федорченко, Л.Н. Подход к реализации атрибутов в системе SynGT [Текст] / Л.Н. Федорченко // Евразийский научный журнал. – 2015. – № 12. – С. 530-536

100. Феоктистов А.Г. Технология имитационного моделирования распределенных пакетов знаний [Текст] / А.Г. Феоктистов // Вестник Томского государственного университета. – 2006. – №18. – С. 248.

101. Феоктистов, А.Г. Логико-вероятностный алгоритм планирования вычислительных заданий [Текст] / А.Г. Феоктистов // Фундаментальные и прикладные научные исследования: сборник статей Международной научно-практической конференции (г. Екатеринбург, 5 ноября 2015 г.). – Уфа : АЭТЕРНА, 2015. – С. 106-110.

102. Феоктистов, А.Г. Язык спецификации вычислительных моделей в

масштабируемых пакетах прикладных программ [Текст] / А.Г. Феоктистов, С.А. Горский // Современные наукоемкие технологии. – 2016. – №.7-1. – С. 84-88.

103. Филд, А. Функциональное программирование [Текст] / А. Филд, П. Хариссон ; пер. с англ. М. В. Горбатовой и др. ; под ред. В.А. Горбатова. – М. : Мир, 1993. – 637 с.

104. Хант, Э. Искусственный интеллект [Текст] / Э. Хант ; пер. с англ. Д.А. Белова и Ю.И. Крюкова ; под ред. В.Л. Стефанюка. – М. : Мир, 1978. – 560 с.

105. Хендерсон, П. Функциональное программирование: применение и реализация [Текст] / П. Хендерсон ; пер. с англ. Л. Т. Петровой. – М. : Мир, 1993. – 349 с.

106. Хювёнен, Э. Мир Лиспа [Текст] : в 2-х т. / Э. Хювёнен, И. Сеппянен. – М. : Мир, 1990. Т. 1 : Введение в язык Лисп и функциональное программирование. – 1990. – 458 с.

107. Хювёнен, Э. Мир Лиспа [Текст] : в 2-х т. / Э. Хювёнен, И. Сеппянен. – М. : Мир, 1990. Т. 2 : Методы и системы программирования. – 1990. – 332 с.

108. Чимитов, Д.Н. Принцип активизации в программировании [Текст] / Д.Н. Чимитов, Б.Б. Бадмаев, В.А. Кравченко // Инфокоммуникационные и вычислительные технологии и системы (ИКВТС-2010): материалы III международной конференции (г. Улан-Удэ, 6-11 сентября 2010 г.). – Улан-Удэ : Изд-во Бурятского госуниверситета, 2010. – С. 285-288.

109. Шарый, С.П. Курс вычислительных методов [Текст] / С.П. Шарый. – Новосибирск : ИВТ СО РАН, 2016. – 527 с.

110. Ярышкина, Н.В. Автоматический синтез цифровых схем по их функциональному описанию [Текст] / Н.В. Ярышкина, П.Б. Могнонов // Вестник БГУ. – 2016. – №3. – С. 80-86.

111. Ярышкина, Н.В. Описание цифровых схем с помощью λ-выражений [Текст] / Н.В. Ярышкина, П.Б. Могнонов // Вестник БГУ. – 2016. – №3. – С. 72-79.

112. Arshinsky, L.V. Logical-Mathematical Modeling of Complex Subject Areas Based On the Logics Vector Semantics [Text] / L.V. Arshinsky, V.L. Arshinsky, A.P. Khomenko, S.K. Kargapoltsev, B.P. Korolkov, V.S. Aslamova // Far East Journal

of Mathematical Sciences. – 2017. – T. 101. – №4. – P. 813-823.

113. Dinechin, Ch. The XL Programming Language [Electronic resource] / Ch. Dinechin – http:// http://xlr.sourceforge.net/language.

114. Knuth, D.E. Semantics of context-free languages [Text] / D.E. Knuth // Mathematical Systems Theory. – 1968. – 2:2. – P. 127-145.

ПРИЛОЖЕНИЕ А

(обязательное)

Листинг программного комплекса логико-математического моделирования

(Св-во №2013613951 от 19.04.2013)

```
(defun выполнить (x) (cond ((= x nil) (prints "Функции не
   добавлены.")) (T (выполнить+ x nil))))
(defun выполнить+ (x w) (cond ((= x nil) (prints "Все функции
   добавлены.")) (T (выполнить+ (cdr x) (eval (car x))))))
(defun ввести (x) (ввести+ x nil))
(defun ввести+ (x w) (cond ((= (cdr x) nil) (set (car (car
   x)) (eval (элемент (car x) 2)))) (T (ввести+ (cdr x) (set
   (car (car x)) (eval (элемент (car x) 2)))))))
 (defun соед (x y) (cond ((= x nil) y) (T (cons (car x) (соед
   (cdr x) y)))))
(defun добав (x y) (соед x (list y)))
(defun добав1 (x y) (добав (обрез x) (добав (послед x) y)))
(defun послед (x) (cond ((= (cdr x) nil) (car x))(T (послед
   (cdr x)))))
(defun дубл (x) (добав x (послед x)))
(defun обрез (x) (cond ((= (cdr x) nil) nil) (T (обрез+ (cdr
   x) (list (car x))))))
(defun обрез+ (x y) (cond ((= (cdr x) nil) y) (T (обрез+
   (cdr x) (добав y (car x))))))
(defun начало (x n) (cond ((= n 2) (list (car x))) ((= n 1)
   nil) (T (соед (list (car x)) (начало (cdr x) (- n 1))))))
(defun конец (x n) (cond ((= n 1) x) (T (конец (cdr x) (- n
   1)))))
```

```
(defun элемент (x n) (cond ((= n 1) (car x)) (T (элемент
   (cdr x) (- n 1)))))
(defun замена (x y n) (соед (начало x n) (соед y (конец x (+
   n 1)))))
(defun вставка (x y n) (соед (начало x n) (cons y (конец x
   n))))
(defun вставка1 (x y n) (соед (начало x n) (cons (добав
   (элемент x n) y) (конец x (+ n 1)))))
(defun содерж (x y) (cond ((= x nil) nil) ((= y (car x)) T)
   (T (содерж (cdr x) y))))
(defun пересеч (x y) (cond ((= y nil) nil) ((содерж x (car
   y)) T) (T (пересеч x (cdr y)))))
(defun включ (x y) (cond ((= x nil) T) ((содерж y (car x))
   (включ (cdr x) y)) (T nil)))
(defun располож1 (x y) (располож1+ x y 1))
(defun располож1+ (x y n) (cond ((= x nil) 0) ((= y (car
   (car x))) n) (T (располож1+ (cdr x) y (+ n 1)))))
(defun колич (x) (cond ((= x nil) 0) (T (+ 1 (колич (cdr
x))))))
(defun печать (x) (печать+ x nil))
(defun печать+ (x w) (cond ((= x nil) nil) (T (печать+ (cdr
   x) (печать+2 (car x) (prints " "))))))
(defun печать+2 (x w) (prints x))
(defun перв (x) (cond ((= x nil) nil) (T (cons (car (car
   x))(перв (cdr x))))))
(defun функции (x) (функции+ x nil 1 0 nil))
(defun функции+ (x v m n w) (cond ((= x nil) (list v w)) (T
   (cond ((= n 0) (функции+ (cdr x) (добав v (cons (car
   (car x)) (list (колич (элемент (car x) 2))))) (+ m 1)
```

```
   (располож1 v (car (car (cdr x)))) (cons (cons 'defun
   (car x)) w))) (T (list (list 0 (car (car x)) n m)))))))
(defun правило (x y n) (cond ((= n 0) (добав y (list (car
   x) (cdr x)))) (T (вставка1 y (cdr x) n))))
(defun продукции (x) (продукции+ x nil))
(defun продукции+ (x y) (cond ((= x nil) y) (T (продукции+
   (cdr x) (правило (car x) y (располож1 y (car (car
   x))))))))
(defun проверка1 (x y) (проверка1+ (cdr x) y (располож1 y
   (car (car (cdr x))))))
(defun проверка1+ (x y n) (cond ((= x nil) nil) ((= n 0)
   (list 1 (car (car x)))) (T (cond ((= (колич (cdr (car
   x))) (car (cdr (элемент y n)))) (проверка1+ (cdr x) y
   (располож1 y (car (car (cdr x)))))) (T (list 2 (car (car
   x)))))))
(defun проверка (x y) (проверка+ x y x (проверка1 (car x) y)
   1))
(defun проверка+ (x y u v m) (cond ((= x nil) u) ((= v nil)
   (проверка+ (cdr x) y u (проверка1 (car (cdr x)) y)(+ m
   1))) (T (добав v m))))
(defun ошибка (x) (ошибка+ (car x) (элемент x 2) (элемент x
   3) (элемент x 4) (prints "База знаний не принята!")))
(defun ошибка+ (k f m n w) (cond ((= k 0) (печать (list
   'Функция f 'повторятся 'на m 'и n 'позициях.))) ((= k 1)
   (печать (list 'Функция f 'продукции m 'не 'задана. )))
   ((= k 2) (печать (list 'Функция f 'в 'продукции m
   'использует 'неправильное 'количество 'аргументов.)))))
(defun база (x y z) (база+ x (car (функции y)) (проверка
   (продукции z) (car (функции y)))(элемент (функции y) 2)))
```

```
(defun база+ (x y z v) (cond ((= v "Все функции добавлены.")
   (set x (list y z))) ((= (car y) 0) (ошибка y)) ((= (car
   z) 1) (ошибка z)) ((= (car z) 2) (ошибка z)) (T (база+ x
   y z (выполнить v)))))
(defun сокращ (x y) (cond ((= x nil) nil) ((содерж y (car
   (car x))) (сокращ (cdr x) y)) (T (cons (car x) (сокращ
   (cdr x) y )))))
(defun добстр (x y k) (cond ((= k 1) (добав x y)) (T (добав
   (обрез x) (добстр (послед x) y (- k 1))))))
(defun уменьш (x) (cond ((= x nil) nil) ((= (car x) 0)
   (уменьш (cdr x))) (T (cons (- (car x) 1) (cdr x)))))
(defun формир (x y z) (формир+ x (cdr y) (cdr z) (list (car
   z)) (list (car (car y))) (car (cdr (car y))) 1 1))
(defun формир+ (x y z v f n k i) (cond ((= x nil) f) ((= i
   n) (формир+ x (cdr y) (cdr z) (cons (car z) v) (добстр
   f (list (car (car y))) k) (car (cdr (car y))) (+ k 1)
   i)) ((= (car v) 1) (формир+ (cdr x) y z (уменьш (уменьш
   v)) (добстр f (car x) k) n (- k 1) (+ i 1))) (T
   (формир+ (cdr x) y z (уменьш v) (добстр f (car x) k) n
   k (+ i 1)))))
(defun коорд (x y) (коорд+ x y 1 (располож1 y (car x))))
(defun коорд+ (x y n m) (cond ((= x nil) '(0 0)) ((= m 0)
   (коорд+ (cdr x) y (+ n 1) (располож1 y (car (cdr x)))))
   (T (добав (list n) m))))
(defun состав (x y z) (cond ((= (cdr x) nil) (list (cons
   (car x) (cons (car y) (list (car z)))))) (T (cons (cons
   (car x) (cons (car y) (list (car z)))) (состав (cdr x)
   (cdr y) (cdr z))))))
(defun стрелка (x y) (стрелка+ x y (коорд (car x) y)))
```

```
(defun стрелка+ (x y z) (стрелка+2 x y (car z) (car (cdr
   z))))
(defun стрелка+2 (x y n m) (cond ((= n 0) x) (T (стрелка+3
   (car x) (элемент x 2) (элемент x 3) (элемент y m) n))))
(defun стрелка+3 (x y z u n) (стрелка+4 (list (замена x
   (cdr (car (cdr u))) n)) (list (добав y (car u))) (cond
   ((= z nil) (list (list (list (car (car (cdr u)))))))) (T
   (list (добав (добав1 z n) (list (car (car (cdr
   u))))))))(cdr (cdr u)) x n))
(defun стрелка+4 (x y z u v n) (cond ((= u nil) (состав x y
   z)) (T (стрелка+4 (добав x (замена v (cdr (car u)) n))
   (дубл y) (добав z (добав (обрез (car z)) (list (car
   (car u))))) (cdr u) v n))))
(defun стрелки (x y) (cond ((= (cdr x) nil) (стрелка (car
   x) y)) (T (соед (стрелка (car x) y) (стрелки (cdr x)
   y)))))
(defun тупики (x) (cond ((= x nil) nil) ((пересеч (car (car
   x)) (элемент (car x) 2)) (тупики (cdr x))) (T (cons
   (car x) (тупики (cdr x))))))
(defun ствол (x y) (стрелка (cons (list x) (cons nil (list
   nil))) y))
(defun поиск (x y) (cond ((= x nil) nil) ((включ (car (car
   x)) y) (car x)) (T (поиск (cdr x) y))))
(defun аргум (x y) (cond ((= x nil) nil) (T (cons (элемент
   (элемент y (располож1 y (car (car x)))) 2) (аргум (cdr
   x) y)))))
(defun уровень (x y z u) (уровень+ (тупики (стрелки x y)) y
   z u))
```

```
(defun уровень+ (x y z u) (cond ((= x nil) (prints "Задача
   не имеет решения!")) (T (уровень+2 x y z (поиск x z)
   u))))
(defun уровень+2 (x y z v u) (cond ((= v nil) (уровень x y
   z u)) (T (формир (car v) (элемент v 3) (аргум (элемент v
   3) u)))))
(defun решение (x y) (решение+ (car x) (сокращ (элемент y 2)
   (cdr x)) (cdr x) (car y)))
(defun решение+ (x y z u) (уровень+ (тупики (ствол x y)) y
   z u))
(defun решение1 (x y) (eval (решение1+ (cons (car x) (перв
   (cdr x))) y (ввести (cdr x)))))
(defun решение1+ (x y z) (решение x y))
```

ПРИЛОЖЕНИЕ Б

(обязательное)

Листинг базы знаний по радиотехнике линейных стационарных систем (фрагмент)

```
(defun умнож (x y) (умнож+ x y (колич x) (колич y)))
(defun умнож+ (x y m n) (умнож+2 x y x m n m (- (+ m n) 1)))
(defun умнож+2 (x y z m n k l) (cond ((= y nil) (нули l))
   ((= x nil) (умнож+2 z (cdr y) z k (- n 1) k l)) (T (прибав
   (умнож+2 (cdr x) y z (- m 1) n k l) (* (car x) (car y))
   (- (+ l 2) (+ m n))))))
(defun прибав (x m n) (замена x (list (+ (элемент x n) m))
   n))
(defun нули (x) (cond ((= x 1) (list 0)) (T (cons 0 (нули (-
   x 1))))))
 (defun слож (x y) (слож+ x y (колич x) (колич y)))
(defun слож+ (x y m n) (cond ((> m n) (слож+ y x n m)) (T
   (слож+2 x y (+ (- n m) 1)))))
(defun слож+2 (x y k) (cond ((= x nil) y) (T (прибав (слож+2
   (cdr x) y (+ k 1)) (car x) k))))
(defun слождр (x y) (cond ((equal (зн x) (зн y)) (list (слож
   (чс x) (чс y)) (зн x))) (T (list (слож (умнож (чс x) (зн
   y)) (умнож (чс y) (зн x))) (умнож (зн x) (зн y))))))
(defun чс (x) (car x))
(defun зн (x) (car (cdr x)))
(defun умнождр (x y) (cond ((equal (чс x) (зн y)) (list (чс
   y) (зн x))) ((equal (чс y) (зн x)) (list (чс x) (зн y)))
   (T (list (умнож (чс x) (чс y)) (умнож (зн x) (зн y))))))
```

```
(defun умнождр (x y) (сокр (cond ((equal (чс x) (зн y)) (list
   (чс y) (зн x))) ((equal (чс y) (зн x)) (list (чс x) (зн
   y))) (T (list (умнож (чс x) (чс y)) (умнож (зн x) (зн
   y)))))))
(defun сокр (x) (cond ((= (послед (чс x)) 0) (cond ((=
   (послед (зн x)) 0) (сокр (list (обрез (чс x)) (обрез (зн
   x))))) (T (сокр+ x)))) (T (сокр+ x))))
(defun сокр+ (x) (cond ((= (cdr (зн x)) nil) (list (делсп
   (чс x) (car (зн x))) (list 1))) (T x)))
(defun делсп (x y) (cond ((= (cdr x) nil) (list (/ (car x)
   y))) (T (cons (/ (car x) y) (делсп (cdr x) y)))))
(defun делдр (x y) (умнождр x (обрдр y)))
(defun обрдр (x) (list (зн x) (чс x)))
(defun R (res) (list (list res) (list 1)))
(defun L (ind) (list (list ind 0) (list 1)))
(defun C (cup) (list (list 1) (list cup 0)))
(defun парал (x y) (делдр (умнождр x y) (слождр x y)))
(defun посл (x y) (слождр x y))
(defun схема (x y) (list x y))
(defun производ (x) (производ+ x (- (колич x) 1)))
(defun производ+ (x n) (cond ((= (cdr (cdr x)) nil) (list
   (car x))) (T (cons (* (car x) n) (производ+ (cdr x) (- n
   1))))))
(defun град (x) (привед (* (/ x _pi) 180)))
(defun привед (x) (cond ((<= x -180) (привед (+ x 360))) ((>
   x 180) (привед (- x 360))) (T x)))
(defun рад (x) (* (/ (привед x) 180) _pi))
(defun мод (x) (sqr (+ (кв (Re x)) (кв (Im x)))))
(defun кв (x) (* x x))
(defun Re (x) (car x))
```

```
(defun Im (x) (car (cdr x)))
(defun арг (x) (град (cond ((= (Re x) 0) (cond ((> (Im x) 0)
   (/ _pi 2)) ((< (Im x) 0) (/ (- 0 _pi) 2)) (T 0))) ((> (Re
   x) 0) (atn (/ (Im x) (Re x)))) (T (cond ((< (Im x) 0) (-
   (atn (/ (Im x) (Re x))) _pi)) (T (+ (atn (/ (Im x) (Re
   x))) _pi)))))))
(defun вещч (x) (* (car x) (cos (рад (car (cdr x))))))
(defun мнимч (x) (* (car x) (sin (рад (car (cdr x))))))
(defun показ (x) (list (модуль x) (аргум x)))
(defun алгебр (x) (list (вещч x) (мнимч x)))
(defun f2 (x) (слождр (car x) (car(cdr x))))
(defun f4 (x y) (делдр y x))
(defun f5 (x) (car (cdr x)))
(defun f7 (x y) (умнождр x y))
(defun f14 (x) (list (компл (чс x) (четн (чс x))) (компл (зн
   x) (четн (зн x)))))
(defun компл (x k) (list (вещ x k) (мним x k)))
(defun четн (x) (cond ((= (cdr x) nil) nil) ((= (cdr (cdr
   x)) nil) T)  (T (четн (cdr (cdr x))))))
(defun вещ (x k) (cond ((= k T) (прореж (cdr x))) (T (прореж
   x))))
(defun мним (x k) (cond ((cdr x) nil) (list 0)) (T (cond ((=
   k nil) (прореж (cdr x))) (T (прореж x)))))
(defun прореж (x) (cond ((= (cdr (cdr x)) nil) (list (car
   x))) (T (cons (car x) (прореж (cdr (cdr x)))))))
```