

РОССИЙСКАЯ АКАДЕМИЯ НАУК
СИБИРСКОЕ ОТДЕЛЕНИЕ
ИНСТИТУТ ДИНАМИКИ СИСТЕМ И ТЕОРИИ УПРАВЛЕНИЯ

На правах рукописи

Бутаков Михаил Игоревич

**Инструментальное средство синтеза
и исполнения транслирующих программ на основе позитивно-
образованных формул**

Специальность 05.13.11 – Математическое и программное обеспечение
вычислительных машин, комплексов и компьютерных сетей

Диссертация на соискание ученой степени
кандидата технических наук

Научный руководитель:
к.ф.-м.н., доц. В.И. Курганский

Иркутск – 2012

Оглавление

Введение.....	4
Глава 1. Синтез транслирующих программ на основе позитивно-образованных формул.....	14
1.1. Язык и исчисление позитивно-образованных формул	17
1.2. Доказательство существования транслирующих программ в исчислении позитивно-образованных формул	20
1.3. Конструктивный фрагмент исчисления позитивно-образованных формул для решения задач трансляции	25
1.4. Синтез транслирующих программ. Принципиальная схема решения задач трансляции на основе исчисления позитивно-образованных формул	29
1.5. Выводы.....	34
Глава 2. Регулярные и LL(1)-грамматики. Существование транслирующей программы.....	36
2.1. Распознающие позитивно-образованные формулы для регулярных правосторонних детерминированных грамматик.....	37
2.2. Распознающие позитивно-образованные формулы для LL(1)-грамматик.....	47
2.3. Выводы.....	67
Глава 3. Компонент синтеза транслирующих программ	68
3.1. Условия и общая схема применения компонента	69
3.2. Организация данных и диалоговый интерфейс компонента.....	73
3.3. Выводы.....	75
Глава 4. Примеры применения компонента синтеза транслирующих программ	84
4.1. Технология решения учебных задач трансляции	85
4.1.1. Учебные трансляторы. Требования и структура	90
4.1.2. Пример учебного транслятора.....	92

4.2. Контроль цепочек входных воздействий в диалоговой системе таможенного контроля Терминал.....	95
4.2.1. Назначение и структура диалоговой системы	96
4.2.2. Логико-лингвистическая модель диалога	98
4.2.3. Реализация модели.....	104
4.3. Выводы	105
Заключение	107
Литература	111
Приложение 1. Множество направляющих символов для LL(1)-грамматик ...	122
Приложение 2. Акт об использовании результатов работы	124

Введение

При решении прикладных задач часто приходится решать задачи трансляции. Примерами программ, где решаются задачи трансляции, являются: AutoCad [38], Microsoft Office (VBA) [46], 1С:Предприятие [45], предметно-ориентированные языки и системы программирования и др [65].

Применение встроенных в прикладные программы средств трансляции и исполнения программного кода [3], формируемого пользователем, позволяет решать ряд важных задач, а именно: связывать воедино отдельные функциональные блоки на основе целостной формальной спецификации; синтезировать новые программы обработки данных на основе имеющихся подпрограмм.

Одним из подходов к решению указанных задач является представление формального языка и входной цепочки в виде логической теории первого порядка [63]. Применение логических средств для решения задач трансляции позволяет:

- Адаптировать средства интеллектуализации к терминологии и свойствам предметной области;
- Сократить время и трудоемкость разработки прикладных программ со встроенными средствами трансляции;
- Повысить качество компонентов прикладной программы за счет их синтеза в результате дедуктивного вывода из спецификаций компонентов и исходных данных;
- Проверять непротиворечивость спецификаций, разработанных на основе известных алгоритмов и т.д.

В связи с этим развитие средств конструирования трансляторов, встраиваемых в прикладные программы, на основе логического подхода является актуальным.

При разработке встроенных систем трансляции на основе логического подхода огромное значение имеет компактное представление знаний и хорошая

адаптация к решаемой задаче. Позитивно-образованные формулы в сравнении с классическими логическими исчислениями обеспечивает более эффективные стратегии логического вывода и обладают рядом важных свойств, необходимых при разработке представленных систем: формулы языка имеют простую и регулярную структуру; позитивно-образованное представление для первопорядковых формул более компактно в сравнение с языком дизъюнктов; сохранение эвристической структуры знания при выводе; хорошая адаптация к решаемой задаче, за счет разработки дополнительных стратегий логического вывода.

Работа основана на результатах, полученных Васильевым С.Н. и его учениками в области логического вывода и приложений логического вывода в составе пакетов прикладных программ; Ершовым А.П. в области доказательного программирования; Манна З. и Уолдингером Р. в области дедуктивного синтеза программ; Тыугу Э.Х. в области структурного синтеза программ. Существенное влияние на результаты диссертации оказали работы Н. Хомского, А.Ахо и Дж.Ульмана, посвященные моделям и методам трансляции.

Синтез идей логического вывода и его приложений в управлении, а также классических методов трансляции, результатов доказательного программирования и дедуктивного синтеза программ позволили получить простой и универсальный метод автоматического синтеза транслирующих программ на основе позитивно-образованных формул.

Цель работы – обеспечить разработку и применение транслирующих программ в составе прикладных программ для формальных языков 2 и 3 классов по Хомскому [68] на основе позитивно-образованных формул.

Для достижения указанной цели были поставлены следующие задачи:

1. Разработать постановку и схему решения задачи трансляции в исчислении ПОФ;

2. Разработать алгоритмы построения распознающих ПОФ по регулярной правосторонней детерминированной грамматике и LL(1)-грамматике;
3. Разработать инструментальное средство для конструирования и применения в составе прикладных программ транслирующих ПОФ;
4. Применить разработанное инструментальное средство на практике.

Объектом исследования является теория и практика спецификации транслирующих программ при помощи логического языка первого порядка. Под транслирующей программой понимается программа, которая принимает на вход цепочку символов (исходный код), представленную на одном языке, и преобразует её в выходную цепочку символов (объектный код) на другом языке.с

Предмет исследования составляет метод автоматического синтеза транслирующих программ обработкой ее спецификации в виде ПОФ.

В исследовании **применены методы** математической логики, формальных языков, теории трансляции и синтеза программ, а также теория программирования.

Научную новизну диссертационного исследования составляют:

1. Постановка и решение задачи трансляции доказательством существования и синтезом транслирующей программы, отличающиеся от известных использованием исчисления ПОФ.
2. Инструментальное средство для конструирования транслирующих программ и их применения в составе прикладных программ, отличающееся от известных использованием универсального распознавателя КС-языков в виде ПОФ.
3. Постановка и схема решения задачи контроля диалога человека и объектной программы на основе формальной спецификации требований к диалогу двухуровневой иерархией формальных языков,

каждый из которых задается ПОФ. Схема отличается от известных применением ПОФ.

4. . Постановка и схема отличаются от известных использованием ПОФ.

Транслирующие ПОФ эффективнее других решений задач трансляции (на основе логической теории первого порядка) так как они имеют более развитые средства управления логическим выводом и позволяют представить задачу трансляции более компактно. Использование ПОФ для решения задач трансляции позволяет объединить решение задачи единым формализмом с другими знаниями в рамках интеллектуальных подсистем.

Практическая значимость. Разработанное инструментальное средство конструирования транслирующих программ на основе ПОФ дает разработчикам возможность дополнить разрабатываемые прикладные программы подсистемами трансляции, визуальной разработки и отладки формальных языков. Причем, процесс отладки формальных языков значительно упрощается за счет интерпретация вывода ПОФ в терминах вопросно-ответных процедур. Это, в свою очередь, улучшает функциональные возможности и гибкость прикладных программ.

Представленные алгоритмы преобразования регулярных и LL(1)-грамматик в ПОФ могут быть использованы для построения более сложных инструментов решения задач трансляции на основе формул первопорядковой логики.

Представление требований к диалогу человека и объектной программы в виде двухуровневой иерархии формальных языков может применяться для решения задач контроля диалога во многих прикладных диалоговых системах.

Разработанное инструментальное средство компонент синтеза транслирующих программ (КСТП), алгоритмы преобразования грамматик в ПОФ и представление требований к диалогу в виде ПОФ применены при решении двух практических задач – разработки учебных трансляторов и контроля диалога в диалоговой системе Терминал.

Достоверность полученных в работе результатов подтверждена доказательством эквивалентности грамматического разбора регулярных и LL(1)- языков процессу доказательства существования соответствующей транслирующей программы.

Эффективность полученных в работе результатов подтверждена опытом практической эксплуатации разработанного автором диссертационной работы инструментального средства. Данное инструментальное средство применяется при решении учебных задач трансляции в ИМЭИ ИГУ в рамках курса «Языки программирования и методы трансляции» с 2007 г. по настоящее время. С помощью КСТП решена задача контроля диалога в рамках специализированной диалоговой системы контроля лесоматериалов. Диалоговая система контроля лесоматериалов рекомендована ФТС России к экспериментальному опробованию в таможенных органах СТУ и ДВТУ¹.

Соответствие диссертации паспорту научной специальности. В диссертации представлена новая модель и схема решения задачи трансляции в исчислении ПОФ. Результаты работы могут быть использованы при проектировании прикладных программ и систем, включающих трансляцию формальных языков. Одним из аспектов использования является контроль допустимости цепочек входных воздействий в составе диалогового интерфейса прикладной программы. Таким образом, тематика диссертации соответствует пунктам 1, 7 области исследований специальности 05.13.11.

Апробация работы. Основные результаты работы представлены на научно-теоретической конференции молодых ученых, посвященной 60-летию Великой Победы (Иркутск, 2005), на VII Всероссийском симпозиуме по прикладной и промышленной математике (Кисловодск, 2006), на VIII школе-семинаре молодых ученых «Математическое моделирование и

¹О распространении системы электронного поштучного учета. Письмо ФТС России от 11 марта 2008 №01-11/9055)

информационные технологии: управление, искусственный интеллект, прикладное программное обеспечение, технологии программирования» (Иркутск, 2006), на IX школе-семинаре молодых ученых «Математическое моделирование и информационные технологии» (Иркутск, 2007), на XIV Байкальской Всероссийской конференции «Информационные и математические технологии в науке и управлении» (Иркутск, 2009), на III школе-семинаре молодых ученых «Инфокоммуникационные и вычислительные технологии и системы» (Улан-Удэ, 2010), на 4-й Всероссийской мультikonференции по проблемам управления (с. Дивногорское, 2011), на объединенном семинаре Института динамики систем и теории управления СО РАН (2008), на кафедре информационных систем Института математики, экономики и информатики ГОУ ВПО ИГУ (2010).

Результаты диссертации были получены в рамках проекта СО РАН «Интеллектуальные методы и инструментальные средства создания и анализа интегрированных распределенных информационно-аналитических и вычислительных систем для междисциплинарных исследований с применением ГИС, GRID и Web-технологий» (№ гос. Регистрации 01.2.00708582), 2007–2009 гг.

Публикации и личный вклад автора. По теме диссертации опубликовано 15 печатных работ, включая 3 статьи [11, 18, 19] в журналах, рекомендованных ВАК для опубликования результатов диссертационных работ, 5 публикаций [9, 12, 13, 14, 16] в трудах региональных, всероссийских, 6 публикаций [10, 17, 20, 39, 40, 41] в научных сборниках и журналах, 1 свидетельство [56] об официальной регистрации программ в Федеральной службе по интеллектуальной собственности, патентам и товарным знакам.

Результаты главы 1 опубликованы в работах [19, 20, 39, 40]; результаты главы 2 опубликованы в [17, 18, 19, 20, 39, 40]; результаты главы 3 опубликованы в [9, 10, 14, 15, 19]; результаты главы 4 опубликованы в [11, 12, 13, 16, 18, 40, 56].

Все результаты, выносимые на защиту, получены автором лично. В основных публикациях по теме диссертации научному руководителю принадлежат постановки исследуемых задач и некоторые теоретические результаты. В работах [1717] и [1818] Курганской О.В. принадлежит результаты по диаграммному представлению транслирующей ПОФ. В работе [1256] Королькову Ю.Д. принадлежат методические и технологические вопросы, связанные с организацией учебного процесса. Все результаты по разработке и реализации алгоритмов принадлежат автору лично.

Все выносимые на защиту положения получены лично. Из совместных работ с В. И. Курганским, О.В. Курганской и Ю.Д. Корольковым в диссертацию включены только те результаты, которые получены автором лично.

В рамках диссертации **получены и выносятся на защиту следующие результаты:**

1. Подмножество языка ПОФ, стратегия вывода τ , постановка и принципиальная схема решения задачи трансляции доказательством существования и синтезом транслирующей программы на основе исчисления ПОФ.

2. Инструментальное средство – компонент синтеза транслирующих программ – для конструирования транслирующих позитивно-образованных формул, а также применения этих формул в составе прикладных программ.

3. Постановка и схема решения задачи контроля диалога человека и объектной программы на основе формальной спецификации требований к диалогу двухуровневой иерархией формальных языков, каждый из которых задается ПОФ. Задача контроля диалога решена применением КСТП.

Диссертация состоит из введения, четырех глав, заключения, библиографии и двух приложений. Общий объем диссертации – 123 страниц. Библиография включает 103 наименования. Работа иллюстрирована 20 рисунками и содержит 4 таблицы.

Глава 1 посвящена постановке и схеме решения задачи трансляции на основе логического вывода в частном случае конструктивного фрагмента исчисления позитивно-образованных формул. В пункте 1.1. приводится описание языка позитивно-образованных формул и построенного в данном языке исчисления. Язык ПОФ является языком первого порядка. Исчисление ПОФ состоит из одного правила вывода. Суть доказательства ПОФ – ее последовательное преобразование с помощью правила вывода. Пункт 1.2 посвящен доказательству существования транслирующих программ в исчислении ПОФ. Под транслирующей программой понимается программа, которая принимает на вход цепочку символов (исходный код), представленную на одном языке, и преобразует её в выходную цепочку символов (объектный код) на другом языке. Пункт 1.3 посвящен описанию подмножества языка ПОФ и выделению частного случая конструктивного фрагмента исчисления позитивно-образованных формул для решения задач трансляции. Основу частного случая конструктивного фрагмента составляют формулы из описанного подмножества языка ПОФ, стандартное правило вывода ω [22] и новая стратегия вывода τ , необходимая для учета особенностей задач трансляции. В п.1.4 введено понятие транслирующей позитивно-образованной формулы и представлена принципиальная схема решения задач трансляции на основе исчисления позитивно-образованных формул. Принципиальная схема решения задачи трансляции основана на идее автоматического синтеза программ [24, 27, 32, 61, 63, 70 с. 242, 93, 94, 103], когда транслирующая программа строится доказательством теоремы ее существования из заранее подготовленного комплекта транслирующих подпрограмм.

В главе 2 приведены алгоритмы построения распознающих позитивно-образованных формул по заданным грамматикам: регулярной правосторонней детерминированной грамматике и LL(1)-грамматике. Для этих грамматик доказана эквивалентность доказательства теоремы существования

транслирующей программы преобразованиями распознающей позитивно-образованной формулы и грамматического вывода.

В главе 3 представлено инструментальное средство – компонент синтеза транслирующих программ (КСТП). Компонент предназначен для конструирования транслирующих позитивно-образованных формул на основе регулярных правосторонних детерминированных и LL(1)-грамматик, а также применения этих формул в составе прикладных программ. В п. 3.1 разработаны условия и схема применения компонента. Компонент синтеза транслирующих программ (КСТП) обеспечивает: конструирование и испытания на примерах транслирующих ПОФ при проектировании прикладной программы; доказательство существования транслирующей программы для заданной входной цепочки языка, синтез и исполнение транслирующей программы при исполнении прикладной программы. Компонент может быть внедрен в инструментальную среду разработки на платформе .Net и использован в прикладной программе на любом из доступных языков программирования для платформы .Net [53, 62, 90]. Пункт 3.2 посвящен организации данных и диалоговому интерфейсу КСТП. Основу КСТП составляют объектные представления транслирующей ПОФ и формальной грамматики.

Четвертая глава посвящена применениям КСТП при решении практических задач. В п. 4.1 представлена технология конструирования учебных трансляторов². Учебный транслятор представляет собой GUI-приложение для платформы .Net. Технологический базис транслятора составляют несколько экземпляров КСТП (по одному для каждого типа лексем и один для синтаксиса входного языка) и программный код объемом около 100 строк исходного текста для управления процессом трансляции. Все шаги этого

² Технология внедрена в Институте математики, экономики и информатики Иркутского государственного университета. Набор компонентов, обеспечивающих данную технологию, включает КСТП и зарегистрирован Иркутским государственным университетом в Федеральной службе по интеллектуальной собственности, патентам и товарным знакам.

процесса обеспечиваются обращением к соответствующим функциональным членам экземпляров КСТП. В п. 4.2 рассмотрена задача контроля цепочек входных воздействий в составе диалогового средства таможенного контроля Терминал³. Это средство представляет собой мобильное вычислительное устройство, оснащенное программой сбора учетных данных и результатов контрольных измерений лесоматериалов необработанных. Для контроля цепочек входных воздействий разработана логико-лингвистическая модель, которая представляет собой комплект транслирующих позитивно-образованных формул. Транслирующие ПОФ разработаны для элементов управления и диалоговых форм и реализованы в виде экземпляров КСТП. Особенность применения КСТП в данном случае заключается в том, что элементами алфавитов формальных языков, которым принадлежат правильные цепочки входных воздействий, являются кортежами. Кортеж это четверка, состоящая из идентификатора элемента диалога, идентификатора события, идентификатора параметра и значения параметра. Доказательство существования, синтез и исполнение транслирующих программ осуществляются в пошаговом режиме по мере формирования входных цепочек.

В заключении обсуждаются перспективы применения и развития разработанного в диссертации аппарата транслирующих позитивно-образованных формул.

³ В соответствии с решением коллегии ФТС России средство таможенного контроля Терминал принято к экспериментальному опробованию в составе новой технологии контроля лесоматериалов необработанных (О распространении системы электронного поштучного учета. Письмо ФТС России от 11 марта 2008 №01-11/9055)

Глава 1. Синтез транслирующих программ на основе позитивно-образованных формул

Поиски систематических методов программирования, обладающих свойством доказательности, имеют уже достаточно длительную историю, восходящую к первым дням электронной вычислительной техники [32]. Среди современных видов доказательного программирования широкое применение получили следующие направления:

1. **Синтез программ.** Предпосылкой данного направления явились попытки исследователей использовать средства автоматического доказательства теорем для решения задач, связанных с контролем имеющихся и синтезом новых аппаратных и программных систем. Одни из первых работ в области синтеза программы были выполнены Саймоном и Грином [70 с. 227], а также Манна и Валденгером [63 с. 38].

Первые полученные результаты в направлении синтеза программ касались формального синтеза алгоритмов и основывались на идее высказанной Саймоном [61, с. 428]. Общий замысел идеи состоит в том, что должна быть доказана теорема с утверждением, что «существует программа p , удовлетворяющая некоторой спецификации». Если удастся ограничить это доказательство конструктивной формой, то появляется возможность извлечь из результатов доказательства требуемую программу.

В синтезе программ существуют различные подходы, среди которых можно выделить следующие: дедуктивный синтез [93] и структурный синтез [44,63].

2. **Планирование вычислений.** Под планированием вычислений понимается планирование вычислительного процесса при решении прикладных задач. Данное направление сформировалось на основе исследований в части поиска решений в пространстве состояний,

доказательства теорем и теории управления, а также на основании практических потребностей робототехники, составления расписаний и других проблемных областей [61, с. 554].

Наиболее заметными результатами в части планирования вычислений является применение булевого моделирования [50, 96], а также конструктивного фрагмента исчисления позитивно-образованных формул (ПОФ) [22, 26, 27, 80, 104] при решении специальных задач планирования вычислений и управления.

Теоретическое исследование проблемы планирования вычислений связано с доказательством теорем существования и единственности разрешающей композиции, а также с разработкой алгоритмов ее построения [50].

Данная глава посвящена одной из частных задач, которая может быть решена методами синтеза программ и планирования вычислений, – задаче трансляции. При решении задач трансляции метод синтеза программ представлен атрибутными грамматиками. Техника атрибутных грамматик существенно развита в системе логического программирования Prolog[91] и системе автоматического синтеза программ ПРИЗ[63].

Задачи трансляции привлекательны не только тем, что являются одной из основ практически всех технологий программирования, но и возможностью их эффективного применения при решении прикладных задач. При решении прикладных задач определенную популярность получили две модели теории трансляции – регулярные выражения и конечные автоматы. Наиболее известными приложениями регулярных выражений являются язык программирования Perl⁴ [37] и командный интерпретатор операционных

⁴ Основной особенностью языка Perl считается богатый набор команд для работы с текстом, в том числе реализованных в виде регулярных выражений

систем семейства Unix⁵ [3 с. 166, 58]. Наиболее распространенными прикладными областями для конечных автоматов являются тестирование (отладка) программного обеспечения [7 с. 237, 64] и автоматное программирование [31, 33, 74, 75]. Успешность и популярность этих разработок объясняются в первую очередь наличием механизмов относительно простого применения моделей и методов трансляции в составе прикладных программ. Известны и другие применения методов трансляции при решении прикладных задач (например, [35 с. 125, 43, 54, 63 с. 241, 82, 83, 88, 89]).

Разработанные в диссертации механизмы синтеза прикладных программ обеспечивают при решении прикладных задач синтетическое применение имеющегося базиса прикладных подпрограмм и моделей трансляции, более сложных по сравнению с регулярными множествами, регулярными грамматиками и конечными автоматами. Эти механизмы по своей сути являются вариантом реализации концепции пакетов знаний [28] на основе спецификации прикладных задач позитивно-образованными формулами (ПОФ).

Ниже в п.1.1 приводится описание языка позитивно-образованных формул и построенного в данном языке исчисления. Язык ПОФ является языком первого порядка. Формулы языка ПОФ имеют древовидную структуру с вершинами в виде кванторов всеобщности или существования. Исчисление ПОФ состоит из одного правила вывода. Суть доказательства ПОФ – ее последовательное преобразование с помощью правила вывода.

В п.1.2 приводится процедура доказательства теоремы существования транслирующей программы в исчислении позитивно-образованных формул. Под транслирующей программой понимается программа, которая принимает на вход цепочку символов (исходный код), представленную на одном языке, и

⁵ В командном интерпретаторе ОС Unix регулярные выражения используются в виде параметров команд и предназначены для манипуляции вводом-выводом

преобразует её в выходную цепочку символов (объектный код), на другом языке.

В п.1.3 описано подмножество язык ПОФ и представлен частный случай конструктивного фрагмента исчисления позитивно-образованных формул для решения задач трансляции. Описанное подмножество языка ПОФ содержит формулы, пригодные для решения задачи трансляции. Основу конструктивного фрагмента составляют: стандартное правило ω доказательства позитивно-образованных формул [22] и новая стратегия вывода τ , необходимая для учета особенностей задач трансляции.

В п.1.4 введено понятие транслирующей позитивно-образованной формулы и представлена принципиальная схема решения задач трансляции на основе исчисления позитивно-образованных формул. Принципиальная схема основана на идее автоматического синтеза программ [28, 63, 93, 94, 103], когда транслирующая программа строится из заранее подготовленного комплекта транслирующих подпрограмм (шаблонов команд запуска транслирующих подпрограмм) на основе доказательства теоремы существования транслирующей программы.

1.1. Язык и исчисление позитивно-образованных формул

Введем язык ПОФ L и исчисление данного языка J .

Язык L – полный язык первого порядка. Алфавит языка L – множество символов следующих видов[105]:

- $x, y, z, \dots$ – символы, обозначающие переменные;
- $a, b, c, \dots$ – константы;
- $+, -, f, g, \dots$ – функциональные символы различной арности;
- $\geq, =, A, B \dots$ – предикатные символы различной арности;
- $(,)$ – запятая, скобки (для уточнения строения формул);

Атомы языка L имеют вид $R^k(t_1, \dots, t_k)$, где R^k – предикатный символ k -арности, а каждая t_i , $i = 1 \dots k$ – переменная или константа. Примерами атомов языка L являются $x \geq 0$, $A(2, y)$.

Конечное множество атомов, записанных через запятую, и пропозициональные константы **T** (истина) и **F** (ложь) – являются конъюнктами в языке L .

Элементарные формулы языка L имеют вид:

- $\exists$ -формулы: $\exists X : A$
- $\forall$ -формулы: $\forall X : A \vee \mathbf{F}$

где X – множество переменных, A – конъюнкт.

Если $\Phi = \Phi_1, \dots, \Phi_k$ является множеством $\forall$ -формул, то $\exists Y : B \Phi$ является $\exists$ -формулой. При $k \geq 2$ ветвление формул множества Φ понимается как конъюнктивное.

Если $\Psi = \Psi_1, \dots, \Psi_k$ является множеством $\exists$ -формул, то $\forall Y : B \Psi$ является $\forall$ -формулой. При $k \geq 2$ ветвление формул множества Ψ понимается как дизъюнктивное.

Основными особенностями формул языка L являются [105, с. 13716]:

- Любая формула, записанная на языке L , имеет крупноблочную структуру, все элементы которой позитивные кванторы;
- Язык ПОФ полон относительно выразительности языка классического исчисления предикатов первого порядка;
- Формулы языка L имеют простую и регулярную структуру. Под регулярностью понимается поочередное появления в каждой ветви узлов $\forall$ -формул и $\exists$ -формул;
- Отрицание ПОФ получается “инвертированием” символов $\forall$, $\exists$;
- Представление первопорядковых формул на языке ПОФ является более компактным, чем в языке дизъюнктов метода резолюций. Это же верно и для пропозициональных формул, в сравнении с

представлениями в стандартных нормальных формах (дизъюнктивных и конъюнктивных).

Далее в работе будем использовать следующую запись формул языка L : $\forall \mathbf{T}\{\Psi, \exists X : A \Phi\}$ [22 с. 137, 23], т.е.

$$F = \forall : \mathbf{T} \begin{cases} \exists X : A \Phi \\ \Psi \end{cases}$$

где: Ψ – множество $\exists$ -формул принадлежащих языку L , Φ – множество

ветвей вида $\forall Y : B \{\exists C_i : Z_i \Psi_i\}_{i=1,k} \stackrel{def}{=} \forall Y : B \begin{cases} \exists C_1 : Z_1 \Psi_1 \\ \exists C_k : Z_k \Psi_k \end{cases}$, $\Psi_1, \dots, \Psi_k$ – множество

$\forall$ -формул принадлежащих языку L .

Исчисление J в языке L имеет единственное правило вывода ω [22 с. 130, 105], основой которого является понятие подстановки θ .

Любой непосредственный последователь $\forall Y : B$ базы $\exists X : A$ называется вопросом к базе $\exists X : A$. Говорят, что вопрос $\forall Y : B$ к базе $\exists X : A$ имеет ответ (ответную подстановку) θ тогда и только тогда, когда θ – отображение $Y \rightarrow X$ и $B\theta \subseteq A$.

Пусть ПОФ F имеет вид $\forall \mathbf{T}\{\Psi, \exists X : A \Phi\}$, а Φ содержит подформулу $\forall Y : B \{\exists Z_i : C_i \Psi_i\}_{i=1,k}$, тогда результатом ωF применения правила вывода ω к вопросу $\forall Y : B$ с соответствующей ответной подстановкой θ будет формула $\omega F = \forall \mathbf{T}\{\Psi, \{\exists X \cup Z_i : A \cup C_i \theta \Phi \cup \Psi_i \theta\}_{i=1,k}\}$ [105, с. 13714].

Логический вывод ПОФ в исчислении J представляет собой цепочку эквивалентных преобразований исходной формулы F по правилу ω , заканчивающуюся формулой $\forall : \mathbf{T} \exists : \mathbf{F}$. Если ПОФ не содержит ветви $\exists : \mathbf{F}$, то она невыводима.

В качестве преимуществ использования исчисления J приводят следующие [105, с. 13717]:

- Исчисление состоит из одно правила вывода ω ;

- Техника вывода (опровержения) формул языка L анализирует ближайшую окрестность корня ПОФ – базу и вопросы к базе. Это позволяет сфокусировать внимание на локальном фрагменте ПОФ без потери свойства полноты вывода;
- Техника вывода может формулироваться в содержательных терминах вопросно-ответной процедуры, вместо технических терминов формальной выводимости;
- Техника вывода хорошо совместима с эвристиками конкретных приложений, а также общими эвристиками управления выводом;
- Исчисление J можно изменять. Семантику исчисления J (конструктивность, не монотонность и др.) можно варьировать путем ограничения применения правила вывода ω .

Далее для решения задач трансляции будем рассматривать подмножество языка ПОФ, ограничиваясь формулами специального вида.

1.2. Доказательство существования транслирующих программ в исчислении позитивно-образованных формул

На программирование можно смотреть как на составление планов решений задач. При таком рассмотрении, процесс программирования начинается с составления общего плана решения задачи. Далее составленный план детализируется и, наконец, исполняется. Достаточно детальный и соответственно оформленный план решения задачи – это программа. Такой подход появился в период с 1970 по 1980 гг. и получил название синтез программ.

Синтез программ – это научное направление, отличительная особенность которого заключается в построении программы на основе доказательства теоремы существования данной программы [32]. Под планированием вычислений понимают разновидность синтеза программы G на основе заранее заданного комплекта базисных подпрограмм $G_1, \dots, G_n$ [24].

Рассмотрим задачу синтеза программы G на основе ее спецификации и комплекта спецификаций базисных подпрограмм $G_1, \dots, G_n$.

Теорема о существовании синтезируемой программы [21, 22, 63] на узком языке исчисления предикатов имеет вид

$$H \rightarrow S \quad (1.1.1)$$

$$\text{где: } H = \left(\forall X_1 P_1 K_1 \rightarrow \exists Y_1 R_1 K_1, Y_1 \wedge \dots \wedge \forall X_n P_n K_n \rightarrow \exists Y_n R_n K_n, Y_n \right), \quad S = \forall X P K \rightarrow \exists Y R K, Y.$$

Поясним обозначения. Формулы $\forall X_1 P_1 K_1 \rightarrow \exists Y_1 R_1 K_1, Y_1, \dots, \forall X_n P_n K_n \rightarrow \exists Y_n R_n K_n, Y_n, \forall X P K \rightarrow \exists Y R K, Y$ - спецификации базисных программ $G_1, \dots, G_n$ и целевой программы G соответственно. Формулы $P_1 K_1, \dots, P_n K_n, P K$ играют роль предусловий рассматриваемых программ, а формулы $R_1 K_1, Y_1, \dots, R_n K_n, Y_n, R K, Y$ - их постусловий. В случае синтеза транслирующей программы формула $\forall X P K \rightarrow \exists Y R K, Y$ является ее спецификацией, а формулы $\forall X_1 P_1 K_1 \rightarrow \exists Y_1 R_1 K_1, Y_1, \dots, \forall X_n P_n K_n \rightarrow \exists Y_n R_n K_n, Y_n$ - спецификациями транслирующих подпрограмм.

Если теорема (1.1.1) доказана, то из этого доказательства и шаблонов команд запуска базисных подпрограмм $G_1, \dots, G_n$, связанных с определенными элементами доказательства, строится транслирующая программа.

Доказательство существования синтезируемой программы эквивалентно доказательству тождественности формулы (1.1.1) или противоречивости отрицания этой формулы. Перепишем теорему о существовании синтезируемой программы (1.1.1) в виде ПОФ:

1. Построим и преобразуем отрицание формулы (1.1.1).

$$\neg(H \rightarrow S) \equiv H \& \neg S$$

2. Спустим отрицание в S непосредственно до атомов.

$$\begin{aligned}\neg S &= \neg \forall X \ P \ \neg \rightarrow \exists Y \ R \ , Y \equiv \exists X \ P \ \neg \wedge \neg \forall Y \ R \ , Y \equiv \\ &\exists X \ P \ \neg \wedge \exists Y \neg R \ , Y \equiv \exists X \ P \ \neg \wedge \exists Y \ R \ , Y \rightarrow False \stackrel{def}{=} S'\end{aligned}$$

3. Переписав полученную формулу $H \& S'$ в символике ПОФ, имеем

$$\exists X : P \ \left\{ \begin{array}{l} \forall X_1 : P_1 \ \neg \exists Y_1 : R_1 \ , Y_1 \dots \\ \forall X_n : P_n \ \neg \exists Y_n : R_n \ , Y_n \dots \\ \forall Y : R \ , Y \ \neg F \end{array} \right. \quad (1.1.2)$$

Формула (1.1.2) является теоремой существования транслирующей программы записанной в виде ПОФ.

Под транслирующей программой понимается программа, которая принимает на вход цепочку символов (исходный код), представленную на одном языке, и преобразует её в выходную цепочку символов (объектный код) на другом языке. Теорема о существовании транслирующей программы строится по входной цепочке символов, требованиям к структуре входных цепочек и дополнительным условиям трансляции.

Элементы $P_1 \ , \dots, P_n \ P$ и $R_1 \ , Y_1 \ , \dots, R_n \ , Y_n \ R \ , Y$ формулы (1.1.2) являются конъюнктами, которые строятся из атомов, символизирующих возможность разбора элемента входной цепочки, успешный разбор элемента входной цепочки, разбор языковой конструкции, обозначенной нетерминальным символом грамматики, а также успешное завершение разбора.

Формула вида (1.1.2) должна иметь хотя бы одну переменную вывода. Это номер или другой идентификатор разбираемого символа входной цепочки. Переменная вывода – это переменная при корневом кванторе существования формулы (1.1.2). Переменная вывода используется в качестве программной переменной в транслирующей программе. Формула вида (1.1.2) может иметь большее количество переменных вывода, в случае их необходимости для управления механизмами диагностирования и исправления ошибок во входной цепочке.

В случае разбора входной цепочки слева направо по одному символу и ограничения входной строки $C = c_1 \dots c_n$ справа специальным символом конца входной строки $\#$, спецификация транслирующей программы имеет вид $\forall u : u = 1, P \llcorner_{1,u} \bar{_} Q \llcorner_{_} \bar{_} \rightarrow \exists : RT \llcorner_{_} \bar{_}$. Где: u - переменная вывода, $Q \llcorner_{_} \bar{_}$ - атом, символизирующий разбор соответствующей языковой конструкции, $P \llcorner_{1,u} \bar{_}$ - атом, символизирующий возможность разбора входного символа c_1 , расположенного в позиции u входной цепочки, $RT \llcorner_{_} \bar{_}$ - атом, символизирующий успешный разбор символа конца строки $\#$.

Спецификации транслирующих подпрограмм $\forall X_i : P_i \llcorner_{i,_} \bar{_} \rightarrow \exists Y_i : R_i \llcorner_{i,Y_i} \bar{_}$ строятся по требованиям к входной цепочке. В последнем случае механизм трансляции на основе ПОФ обеспечивает диагностирование ошибки во входной строке и/или исправление ошибки. Все это осуществляется исполнением соответствующей транслирующей подпрограммы.

Спецификация транслирующей подпрограммы должна иметь, по меньшей мере, один формальный параметр – это номер позиции или идентификатор разбираемого входного символа. Формальный параметр – это переменная при кванторах $\forall$ или $\exists$ в составе вопросов ПОФ. Другие формальные параметры могут потребоваться для управления механизмами диагностирования и исправления ошибок во входной строке.

Таким образом, теорема существования транслирующей программы записанной в виде ПОФ включает транслируемую входную цепочку символов, требования к структуре входных цепочек и дополнительным условиям трансляции. Требования к структуре входной цепочки могут быть получены, например, из контекстно-свободной грамматики.

Рассмотрим пример. Пусть дана грамматика $G = \langle V_T, V_N, P, S \rangle$, где: V_T – алфавит терминальных символов, V_N – множество нетерминальных символов,

P – конечное множество правил, S – аксиома грамматики. Определим G следующим образом:

$$V_T = \{u, .\}^6, V_N = \{A, B, C\}, S = A,$$

$$P = A \rightarrow u \quad A \rightarrow uA \quad A \rightarrow .B \quad B \rightarrow uC \quad B \rightarrow .C \quad C \rightarrow uC \quad C \rightarrow u \quad .$$

Это регулярная грамматика описывающая язык дробных констант без знака.

Теорема существования транслирующей программы (1.1.2) для трансляции цепочки $C = 80.7$ в грамматике G имеет вид (1.1.3).

Поясним элементы формулы (1.1.3). Часть вопросов ПОФ построена по правилам грамматики G , например, по правилу $A \rightarrow uA$ построен вопрос $\forall s : P(u, s), Q(A) \exists : R(u, s), Q(A)$, где: атом $P \# , s$ означает, что обобщенный терминальный символ $u \in V_T$ находится в позиции s исходной цепочки C и может быть разобран, атом $Q \#$ означает, что нетерминальный символ $A \in V_N$ разобран, атом $R \# , s$ означает, что обобщенный терминальный символ $u \in V_T$ находится в позиции s исходной цепочки C и успешно разобран. Другая часть ПОФ построена по символам входной цепочки, например, вопросы $\forall s : s = 1, R(8, s) \exists : s = 2, P(0, s),$ $\forall s : s = 2, R(0, s) \exists : s = 3, P(., s)$ и $\forall s : s = 3, R(., s) \exists : s = 4, P(7, s), PT(\#)$ построены по первым трем символам входной цепочки, где атом $PT \#$ означает что терминальный символ конца строки $\#$ может быть разобран.. База ПОФ $\exists u : u = 1, P(8, u), Q(A)$ соответствует начальному состоянию разбора, она построена по аксиоме грамматики и первому символу входной цепочки. Вопрос $\forall : RT(\#) \exists : F$ соответствует окончанию разбора и построен по символу конца входной цепочки, где атом $RT \#$ означает что терминальный символ конца строки $\#$ разобран.

⁶ Символ u является обобщенным терминальным символом и служит для обозначения цифр [36]

Алгоритм построения формул вида (1.1.3) по заданной грамматике подробно описан в главе 2 настоящей работы.

Доказательство формулы (1.1.3) представлено ниже в п. 1.3.

$$\forall : \mathbf{T} \quad \exists u : u = 1, P(8, u), Q(A) \left\{ \begin{array}{l} \forall s : P(u, s), PT(\#), Q(A) \exists : RT(\#) \\ \forall s : P(u, s), Q(A) \exists : R(u, s), Q(A) \\ \forall s : P(., s), Q(A) \exists : R(., s), Q(B) \\ \forall s : P(u, s), Q(B) \exists : R(u, s), Q(C) \\ \forall s : P(u, s), PT(\#), Q(B) \exists : RT(\#) \\ \forall s : P(u, s), Q(C) \exists : R(u, s), Q(C) \\ \forall s : P(u, s), PT(\#), Q(C) \exists : RT(\#) \\ \forall : RT(\#) \exists : \mathbf{F} \\ \forall s : s = 1, R(8, s) \exists : s = 2, P(0, s) \\ \forall s : s = 2, R(0, s) \exists : s = 3, P(., s) \\ \forall s : s = 3, R(., s) \exists : s = 4, P(7, s), PT(\#) \end{array} \right. \quad (1.1.3)$$

1.3. Конструктивный фрагмент исчисления позитивно-образованных формул для решения задач трансляции

Определим подмножество языка L . Формулы данного подмножества имеют специальный вид (1.1.4) и предназначены для решения задачи трансляции. Формула вида (1.1.4) – это уточнение формулы вида (1.1.2) для решения задачи трансляции.

$$F = \forall : \mathbf{T} \quad U_B \quad U_Q \quad (1.1.4)$$

где $U_B = \exists u : K_S \quad \bar{K}_D$, а $U_Q = \forall s : P_1(s) \exists R_1(s) \dots \forall s : P_n(s) \exists R_n(s) \forall : R \quad \bar{\mathbf{F}}$.

Часть $\forall : \mathbf{T}$ в начале формулы необходима для построения дизъюнктивного ветвления при доказательстве ПОФ.

Поясним обозначения. Конъюнкты $K_S \quad \bar{K}_D$, $P_1 \quad \bar{K}_1 \dots, P_n \quad \bar{K}_n \quad \bar{P} \quad \bar{K}$, $R_1 \quad \bar{K}_1, Y_1 \dots, R_n \quad \bar{K}_n, Y_n \quad \bar{R} \quad \bar{K}, Y$ состоят из атомов $RT \quad \bar{K}$, $R \quad \bar{K} \quad s$, $PT \quad \bar{K}$, $P \quad \bar{K} \quad s$, $Q \quad \bar{K}$ и $EOP()$.

Атом $RT \quad \bar{K}$ означает, что терминальный символ $t \in V_T$ успешно разобран.

Атом $R \quad \bar{K} \quad s$ означает, что терминальный символ $t \in V_T$ находится в позиции s

исходной цепочки C и успешно разобран. Аналогично, атомы $PT \llbracket _$ и $P \llbracket s _$ означают, что терминальный символ $t \in V_T$ находится в позиции s исходной цепочки C и может быть разобран. Атом $Q \llbracket _$ означает, что нетерминальный символ $Z \in V_N$ разобран. Атом $EOP()$ означает, что разбор цепочки C доказательством формулы F успешно завершен.

Далее формулу U_B будем называть базой ПОФ, а конъюнкты, составляющие U_Q , будем называть вопросами ПОФ.

Далее формулы вида (1.1.4) будем называть распознающими позитивно-образованными формулами.

Напомним, доказательство позитивно-образованной формулы представляет собой ее многократные преобразования по правилу ω [22]. Применение правила ω заключается в поиске подходящего вопроса в U_Q и соответствующей обработке базы U_B . В нашем случае параметрами однократного применения правила ω являются ответная подстановка $\theta: u \rightarrow s$ и один из вопросов Q части U_Q формулы (1.1.2).

Рассмотрим один из вариантов решения задачи трансляции:

- При описании структуры цепочки используются нетерминальные и терминальные символы;
- Разбор входной цепочки выполняется слева направо по одному символу;
- Конец входной цепочки обозначен специальным терминальным символом;
- Нетерминальным символам, т.е. конструкциям формального языка, могут соответствовать транслирующие подпрограммы. Транслирующая подпрограмма задает преобразование конструкции формального языка или диагностирует ошибку.

С учетом представленных соглашений выделим частный случай конструктивного фрагмента исчисления ПОФ путем дополнения стандартного правила вывода ω новой стратегией вывода τ . Использование стратегии

вывода предотвращает заикливание и устраняет неопределенность при доказательстве ПОФ.

Обозначим через ρ , ω_1 и ω_2 составные элементы стратегии вывода τ .

Элемент ρ стратегии τ заключается в разделении базы U_B позитивно-образованной формулы F на два конъюнкта – конъюнкт синтеза и конъюнкт вывода. Конъюнкт синтеза K_S составляют атомы уже использованные при преобразовании по правилу ω формулы F , а конъюнкт вывода K_D – атомы которые можно использовать в данный момент при применении правила ω . После каждого успешного применения правила ω конъюнкт синтеза пополняется новыми атомами из конъюнкта вывода, а в конъюнкт вывода копируется постусловие обрабатываемого вопроса⁷. Формальные параметры постусловия при этом должны быть заменены переменными вывода в соответствии с ответной подстановкой θ .

Элемент ω_1 стратегии τ заключается в поиске подходящего вопроса для выполнения преобразования формулы F по правилу ω путем просмотра конъюнкта вывода слева направо. Глубина просмотра при проверке выполнимости вопроса определяется мощностью множества атомов, составляющих конъюнкт данного вопроса. Критерием применимости правила ω к данному вопросу является принадлежность каждого атома предусловия вопроса, обработанного ответной подстановкой θ , конъюнctu вывода.

Элемент ω_2 стратегии τ заключается в пополнении конъюнкта вывода атомами постусловия, примененного в рамках правила ω вопроса, путем их приписывания к конъюнctu вывода только слева.

Однократное применение правила ω с учетом стратегии вывода τ заключается в следующей обработке ПОФ:

⁷ Под обрабатываемым вопросом понимается вопрос, к которому было применено правило вывода ω .

1. Применение элемента ω_1 стратегии τ и выполнение ответной подстановки $\theta: u \rightarrow s$ для выделенного вопроса Q .
2. Применение элемента ω_2 стратегии τ , т.е. копирование постусловия вопроса Q в конъюнкт вывода K_D в случае принадлежности атомов, составляющих конъюнкт предусловия обрабатываемого вопроса Q конъюнкту вывода K_D .
3. Применение элемента ρ стратегии τ , т.е. перемещение атомов предусловия обработанного вопроса Q из конъюнкта вывода K_D в конъюнкт синтеза K_S в случае принадлежности атомов, составляющих конъюнкт предусловия обрабатываемого вопроса Q , конъюнкту вывода K_D .

Далее обработку позитивно-образованной формулы по правилу $\omega(Q, \theta)$ будем называть выполнением вопроса Q .

Невозможность применения элемента ω_1 стратегии вывода τ означает недоказуемость распознающей ПОФ. Случай, когда конъюнкт вывода содержит **F**, означает, что распознающая позитивно-образованная формула успешно доказана. Если по правилу ω_1 найден более чем один вопрос, то формула преобразуется к виду с дизъюнктивным ветвлением баз (по одной на каждый вопрос).

Вернемся к примеру (1.1.3). При доказательстве имеет место следующая последовательность значений⁸ конъюнкта вывода K_D в составе базы U_B :

⁸ Разделяющий символ " $\Rightarrow$ " - служит для обозначения шага логического вывода ПОФ.

$$\begin{aligned}
&u = 1, P(8, u), Q(A) \Rightarrow u = 1, R(8, u), Q(A) \Rightarrow \\
&u = 2, P(0, u), Q(A) \Rightarrow u = 2, R(0, u), Q(A) \Rightarrow \\
&u = 3, P(., u), Q(A) \Rightarrow u = 3, R(., u), Q(B) \Rightarrow \\
&u = 4, P(7, u), PT(\#), Q(B) \Rightarrow u = 4, RT(\#) \Rightarrow \\
&u = 4, F
\end{aligned}$$

Вывод : строка 80.7 разобрана.

1.4. Синтез транслирующих программ. Принципиальная схема решения задач трансляции на основе исчисления позитивно-образованных формул

Получение кода транслирующей программы из доказательства распознающей ПОФ требует наличия в формуле дополнительных данных об обработке, как отдельных элементов входной цепочки, так и ее синтаксических конструкций. Для записи такой «технической» информации введем в выделенное подмножество языка L следующий нелогический элемент, именуемый шаблоном команды запуска транслирующей подпрограммы:

$[a = P(b)]$

где: P – имя обрабатывающей (транслирующей) подпрограммы, b – входная переменная транслирующей подпрограммы, a – выходная переменная. В качестве входных и выходных переменных используются переменные вывода позитивно-образованной формулы и текущий элемент входной цепочки. Если транслирующая подпрограмма не возвращает значение, то нелогический элемент принимает вид $[P(b)]$.

Введенный нелогический элемент может быть связан с атомами начальной установки конъюнкта вывода K_D или атомами постусловия вопросов. Отметим, что введенный элемент не влияет на доказательство позитивно-образованной формулы.

Далее формулы вида (1.1.4) с нелогическими элементами будем называть транслирующими.

Рассмотрим пример. Пусть требования к входным цепочкам заданы LL(1)-грамматикой $G = \langle V_T, V_N, P, S \rangle$, где

$$V_T = \{+, *, i\}, V_N = \{A, B, C, D, E, F\}, S = A,$$

$$P = \left\{ \begin{array}{l} A \rightarrow CB \quad B \rightarrow +CB \quad B \rightarrow \varepsilon \quad C \rightarrow ED \quad D \rightarrow *ED \\ D \rightarrow \varepsilon \quad E \rightarrow iF \quad F \rightarrow iF \quad F \rightarrow \varepsilon \end{array} \right\};$$

Данная грамматика описывает язык арифметических выражений. Пример строки, которую может породить грамматика, имеет вид $i + i$.

Теорема существования транслирующей программы для трансляции цепочки $C = i + i$ в грамматике G имеет вид (1.1.5). Алгоритм построения ПОФ по LL(1)-грамматике подробно описан в главе 2 настоящей работы.

$$\forall : \mathbf{T} \quad \exists u : u = 1, P(i, u), Q(A), EOP() \left\{ \begin{array}{l} \forall s : P(+, s), Q(B) \exists : R(+, s), Q(C), Q(B) \\ \forall s : P(*, s), Q(D) \exists : R(*, s), Q(E), Q(D) \\ \forall s : P(i, s), Q(E) \exists : R(i, s), Q(F) \\ \forall s : P(i, s), Q(F) \exists : R(i, s), Q(F) \\ \forall s : P(i, s), Q(A) \exists : P(i, s), Q(C), Q(B) \\ \forall s : P(i, s), Q(C) \exists : P(i, s), Q(E), Q(D) \\ \forall s : PT(\#), Q(B) \exists : PT(\#) \\ \forall s : P(+, s), Q(D) \exists : P(+, s) \\ \forall s : PT(\#), Q(D) \exists : PT(\#) \\ \forall s : P(+, s), Q(F) \exists : P(+, s) \\ \forall s : P(*, s), Q(F) \exists : P(*, s) \\ \forall s : PT(\#), Q(F) \exists : PT(\#) \\ \forall s : s = 1, R(i, s) \exists : s = 2, P(+, s) \\ \forall s : s = 2, R(+, s) \exists : s = 3, P(i, s) \\ \forall s : s = 3, R(i, s) \exists : PT(\#) \\ \forall PT(\#), EOP() \exists : \mathbf{F} \end{array} \right. \quad (1.1.5)$$

Формула (1.1.6) представляет собой пример транслирующей позитивно-образованной формулы. Она получена дополнением формулы (1.1.5) нелогическими элементами, а именно: шаблонами команд запуска транслирующих подпрограмм. Шаблоны команд запуска транслирующих

подпрограмм выделены квадратными скобками и размещаются непосредственно за атомами, применение которых на успешном шаге доказательства позитивно-образованной формулы приводит к пополнению транслирующей программы новой командой запуска соответствующей транслирующей подпрограммы.

$$\begin{aligned}
 & \forall : \mathbf{T} \exists u: u = 1, P(i, u), Q(A)[\mathbf{u = START()}], EOP()[\mathbf{FINISH()}] \\
 & \left\{ \begin{aligned}
 & \forall s : P(+, s), Q(B) \exists : R(+, s), Q(C), Q(B)[\mathbf{add()}] \\
 & \forall s : P(*, s), Q(D) \exists : R(*, s), Q(E), Q(D)[\mathbf{mult()}] \\
 & \forall s : P(i, s), Q(E) \exists : R(i, s), Q(F) \\
 & \forall s : P(i, s), Q(F) \exists : R(i, s), Q(F)[\mathbf{push(u)}] \\
 & \forall s : P(i, s), Q(A) \exists : P(i, s), Q(C), Q(B) \\
 & \forall s : P(i, s), Q(C) \exists : P(i, s), Q(E), Q(D) \\
 & \forall s : PT(\#), Q(B) \exists : PT(\#) \\
 & \forall s : P(+, s), Q(D) \exists : P(+, s) \\
 & \forall s : PT(\#), Q(D) \exists : PT(\#) \\
 & \forall s : P(+, s), Q(F) \exists : P(+, s) \\
 & \forall s : P(*, s), Q(F) \exists : P(*, s) \\
 & \forall s : PT(\#), Q(F) \exists : PT(\#) \\
 & \forall s : s = 1, R(i, s) \exists : s = 2[\mathbf{u = plus1(u)}], P(+, s) \\
 & \forall s : s = 2, R(+, s) \exists : s = 3[\mathbf{u = plus1(u)}], P(i, s) \\
 & \forall s : s = 3, R(i, s) \exists PT(\#) \\
 & \forall PT(\#), EOP() \exists \mathbf{F}
 \end{aligned} \right. \tag{1.1.6}
 \end{aligned}$$

Команда запуска транслирующей подпрограммы строится из ее шаблона, в который при необходимости подставляются переменная вывода и значение текущего символа входной цепочки.

Так как входная цепочка входит в состав транслирующей ПОФ, то транслирующая программа представляет собой последовательность команд безусловного запуска базисных транслирующих подпрограмм.

Команда запуска транслирующей подпрограммы включается в транслирующую программу после успешного шага доказательства ПОФ с

применением атома, с которым был связан шаблон этой команды. По существу, транслирующая программа представлена последовательностью нелогических элементов в составе конъюнкта синтеза K_s .

Применение транслирующей ПОФ базируется на ее инкапсуляции в программный компонент. Под компонентом понимается конструктивная единица среды программирования с определенным интерфейсом. В данном случае интерфейс компонента состоит из методов, свойств и событий. Интерфейс компонента предназначен для управления доказательством существования ПОФ, синтезом и исполнением транслирующей программы.

Рассмотрим транслирующую программу для ПОФ (1.1.6). Транслирующая программа выполняется в объектной среде, представленной на рис. 1.1. Напомним, что в состав позитивно-образованной формулы включена входная цепочка $i + i$. В объектной среде входная цепочка из рассматриваемого примера представлена коллекцией (рис.1.1). Транслирующая программа обеспечивает исполнение распознанной входной цепочки из трех лексем со значениями “0.34”, “+” и “7.5”. Дескрипторы этих лексем i , $+$, i соответственно.

Описание транслирующих подпрограмм, использованных в формуле на (1.1.6), приведено в таблице 1.1.

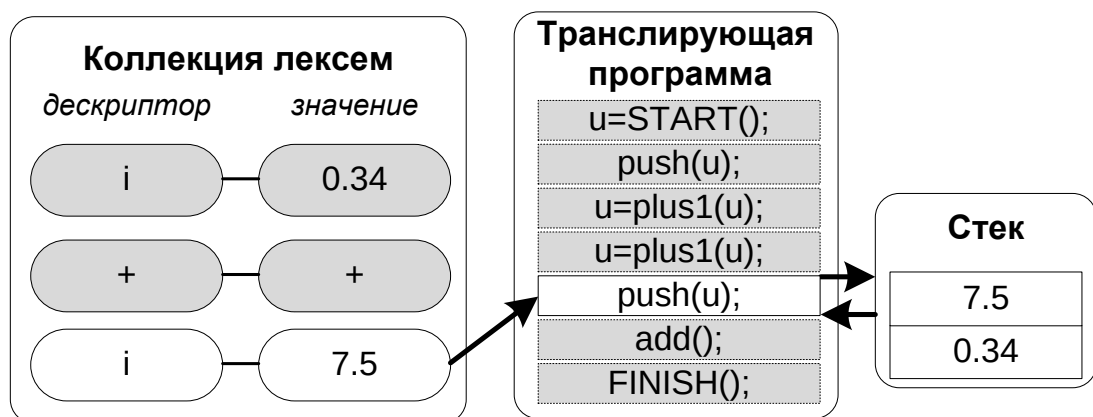


Рис. 1.1. Основные элементы объектной среды для исполнения транслирующей программы примера. Схема работы транслирующей подпрограммы push

Успешное доказательство транслирующей ПОФ означает существование транслирующей программы. ПОФ (1.1.6) была успешно доказана. Синтезированная в процессе доказательства транслирующая программа имеет вид:

$$u = START \quad \leftarrow ; push \quad \leftarrow ; u = Plus1 \quad \leftarrow ; \\ u = Plus1 \quad \leftarrow ; push \quad \leftarrow ; add \quad \leftarrow ; FINISH \quad \leftarrow ;$$

Для исполнения транслирующей программы необходима специальная среда. Данная среда должна включать стек, коллекцию лексем и транслирующие подпрограммы.

Транслирующие подпрограммы примера

Таблица 1.1.

№	Шаблон команды	Описание
1	u=START();	Подпрограмма инициализации процесса трансляции. Создает стек, присваивает начальное значение переменной вывода.
2	FINISH();	Подпрограмма завершения процесса трансляции. Выводит содержимое стека в выходную строку.
3	u=Plus1(u);	Подпрограмма увеличения на единицу значения переменной вывода.
4	push(u);	Подпрограмма работы со стеком. Записывает в стек значение текущей лексемы.
5	add();	Подпрограмма работы со стеком. Вынимает два верхних значения стека и помещает их сумму в стек.
6	mult();	Подпрограмма работы со стеком. Вынимает два верхних значения стека и помещает их произведение в стек.

Рассмотрим принципиальную схему решения задачи трансляции. Данная схема в выделенном частном случае конструктивного фрагмента исчисления ПОФ приведена на рис.1.3. Входными данными являются требования к входной цепочке, входная цепочка, шаблоны команд запуска транслирующих подпрограмм и дополнительные условия трансляции. По входным данным строится ПОФ специального вида. Построенная ПОФ доказывается. По итогам доказательства синтезируется транслирующая программа. Далее транслирующая программа исполняется, а результат ее исполнения записывается в выходную цепочку.

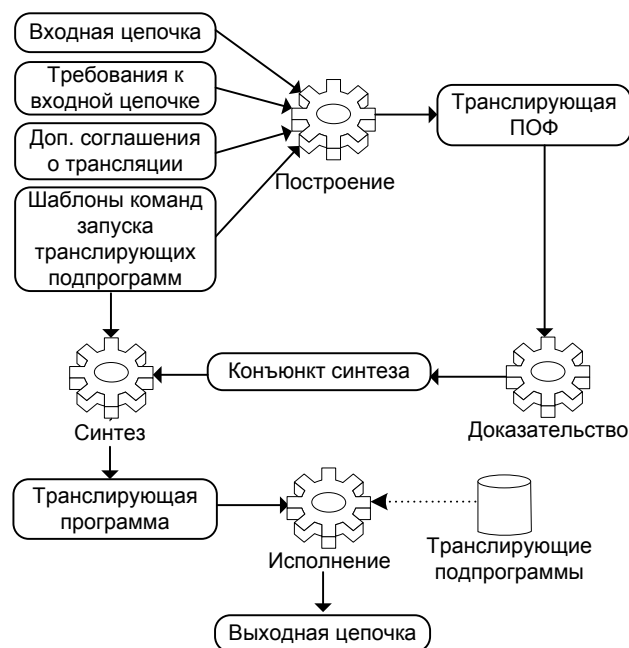


Рис.1.3. Принципиальная схема решения задачи трансляции в частном случае конструктивного фрагмента исчисления ПОФ

При решении задач трансляции можно строить иерархические композиции транслирующих ПОФ. Транслирующая программа, синтезированная при выводе одной транслирующей ПОФ, может быть использована в качестве одной из транслирующих подпрограмм в транслирующей ПОФ более высокого уровня. В частности, таким образом была решена задача контроля диалога, требования к которому были представлены двухуровневой иерархией формальных языков [11].

1.5. Выводы

В данной главе разработано подмножество языка L . Данное подмножество задает распознающие ПОФ вида (1.1.4). На основе распознающих ПОФ разработана схема решения задачи трансляции. Введено понятие транслирующей позитивно-образованной формулы, обеспечивающее при доказательстве существования транслирующей программы ее синтез. Разработана схема решения задачи трансляции на основе позитивно-образованных формул.

Это один из основных результатов диссертационной работы, который выносится на защиту. Результат опубликован в [17-20].

Глава 2. Регулярные и LL(1)-грамматики. Существование транслирующей программы

При решении задачи трансляции, как правило, используют пару математических моделей – формальную грамматику и распознающий автомат. С помощью формальной грамматики описывают множество цепочек символов, составляющих формальный язык. Далее формальную грамматику преобразуют в распознающий автомат, а автомат – в программу трансляции. Преобразование распознающего автомата в транслятор можно автоматизировать. На этом построена индустрия так называемых компиляторов компиляторов [36, 48].

Классификация формальных грамматик и соответствующих им распознающих автоматов была выполнена Хомским [68] при изучении естественных языков. Разработчики трансляторов языков программирования для описания лексики и синтаксиса часто используют нормальную форму Бэкуса-Наура [см., например 34].

Следует отметить, что решение практических задач трансляции требует одновременной работы с несколькими формальными языками. Так, языки программирования часто рассматривают как двухуровневую иерархию формальных языков. Нижний уровень составляют регулярные языки, каждый из которых задает лексемы определенного типа, а верхний – одна из КС-грамматик, которая задает синтаксис.

Построенная формальными методами программа распознавания входного текста требует значительной доработки. На этом этапе, в частности, решаются задачи семантического анализа, исполнения или синтеза объектной программы.

Работа с регулярными грамматиками и остальными КС-грамматиками предполагает построение распознавателей. Способы построения данных распознавателей у каждого автора свои (например, [36, 48, 59]). Кроме этого построенные распознаватели отличаются друг от друга по алгоритму работы и структуре, так, например, распознаватели для регулярной грамматики не

содержит символа конца строки, а распознаватели для LL и LR грамматик ведут работу со стеком.

Рассматриваемое в диссертации решение задачи трансляции основано на формулах вида (1.1.4) (распознающие ПОФ). Каждая такая формула строится по заданной КС-грамматике. Распознающая ПОФ обеспечивает представление процесса распознавания входной цепочки формального языка ее доказательством (выводом). Решение задачи трансляции путем доказательства распознающей ПОФ позволяет получить универсальный и единый распознаватель для КС-грамматик.

Ниже п.2.1 представлен алгоритм построения распознающей позитивно-образованной формулы для регулярной правосторонней детерминированной грамматики и доказано утверждение об эквивалентности процессов грамматического разбора и доказательства позитивно-образованной формулы. В п.2.2. аналогичная работа выполнена для LL(1)-грамматик. Эти грамматики были выбраны для исследования в связи с их высокой практической значимостью и хорошей изученностью.

2.1. Распознающие позитивно-образованные формулы для регулярных правосторонних детерминированных грамматик

Обозначим $G = \{V_T, V_N, P, S\}$ контекстно-свободную грамматику с множеством терминальных символов V_T , множеством нетерминальных символов V_N , множеством правил P и аксиомой S [36]. Язык порождаемый данной грамматикой будем обозначать $L(G)$.

При преобразовании правил грамматики в элементы распознающей ПОФ будем использовать обозначение $\Downarrow$. Оно символизирует преобразование цепочки $\gamma = \gamma_1 \dots \gamma_k \in V_T \cup V_N^*$ в конъюнкт K по следующим правилам. Если выполнено условие $\gamma_i \in V_T$, то $\Downarrow_i \equiv RT \Downarrow_i$. Если выполнено условие $\gamma_i \in V_N$, то $\Downarrow_i \equiv Q \Downarrow_i$. Таким образом, последовательность символов грамматики

преобразуется в конъюнкт в составе ПОФ. Например, в случае $V_T = 1,2,3$, а $V_N = A,B,C$ цепочка $\gamma = A2B3C$ преобразуется в конъюнкт $K = Q(A), RT(2), Q(B), RT(3), Q(C)$.

Все другие используемые ниже обозначения введены в главе 1 настоящей работы, либо вводятся по мере необходимости.

Алгоритм построения распознающей ПОФ F для регулярной грамматики включает начальную установку позитивно-образованной формулы F , обработку правил грамматики G и обработку элементов входной цепочки C .

Начальная установка. Если входная строка имеет вид $C = c_1$, положим $U_B = \exists u : u = 1, P(c_1, u), PT(\#), Q(S)$, иначе, если $C = c_1, \dots, c_n, n > 1$, положим $U_B = \exists u : u = 1, P(c_1, u), Q(S)$. Положим $U_Q = \forall : RT(\#) \exists : F$.

Обработка правил грамматики G .

1. Для каждого правила вида $Z \rightarrow aY$, где $Z, Y \in V_N, a \in V_T$, пополним U_Q вопросом $\forall s : P(a, s), Q(Y) \exists : R(a, s), Q(Z)$;
2. Для каждого правила вида $Z \rightarrow a$, где $Z \in V_N, a \in V_T$, пополним U_Q вопросом $\forall s : P(a, s), PT(\#), Q(Z) \exists : RT(\#)$.

Обработка элементов входной цепочки⁹ $C = c_1 \dots c_n, n > 1$:

1. Для каждого символа исходной цепочки $c_i, i = 1, \dots, n-2$ пополним U_Q вопросом $\forall s : s = i, R(c_i, s) \exists : s = i+1, P(c_{i+1}, s)$;
2. Для символа c_{n-1} пополним U_Q вопросом $\forall s : s = n-1, R(c_{n-1}, s) \exists : s = n, P(c_n, s), PT(\#)$.

⁹ Если входная цепочка состоит из одного символа ($n=1$), то он учитывается в разделе алгоритма **Начальная установка** и дополнительной обработки не требует.

Построенную таким образом формулу¹⁰ обозначим $F_R \langle G, C \rangle$ и назовем распознающей позитивно-образованной формулой.

Рассмотрим пример.

Пусть дана регулярная грамматика $G = \langle V_T, V_N, P, S \rangle$, где

$$V_T = \{a, b\}, V_N = \{A, B, C\}, S = A,$$

$$P = A \rightarrow aA \mid A \rightarrow bB \mid B \rightarrow aC \mid B \rightarrow bC \mid C \rightarrow a \mid C \rightarrow b.$$

Распознающая позитивно-образованная формула $F_R \langle G, C \rangle$ для решения задачи разбора строки $C = 80.7$ в грамматике G имеет вид:

$$\exists u : u = 1, P(8, u), Q(A) \left\{ \begin{array}{l} \forall s : P(a, s), PT(\#), Q(A) \exists : RT(\#) \\ \forall s : P(a, s), Q(A) \exists : R(a, s), Q(A) \\ \forall s : P(b, s), Q(A) \exists : R(b, s), Q(B) \\ \forall s : P(a, s), Q(B) \exists : R(a, s), Q(C) \\ \forall s : P(a, s), PT(\#), Q(B) \exists : RT(\#) \\ \forall s : P(a, s), Q(C) \exists : R(a, s), Q(C)) \\ \forall s : P(a, s), PT(\#), Q(C) \exists : RT(\#) \\ \forall : RT(\#) \exists : F \\ \forall s : s = 1, R(8, s) \exists : s = 2, P(0, s) \\ \forall s : s = 2, R(0, s) \exists : s = 3, P(b, s) \\ \forall s : s = 3, R(b, s) \exists : s = 4, P(7, s), PT(\#) \end{array} \right. \quad (2.1.1)$$

Рассмотрим вопрос связи доказательства распознающей ПОФ с выводом цепочки в регулярной грамматике, для этого докажем утверждение.

Утверждение 1. Для всякой праволинейной детерминированной регулярной (автоматной) грамматики G и входной цепочки C формула $F_R(G, C)$ доказуема тогда и только тогда, когда $C \in L(G)$.

Доказательство. Рассмотрим последовательность $\Phi = \varphi_0, \dots, \varphi_n$ грамматического разбора цепочки C в грамматике G и последовательность

¹⁰ Для упрощения записи корень позитивно-образованной формулы $\forall T$ здесь и далее будем опускать.

$K = K_D^0, \dots, K_D^m$ состояний конъюнкта K_D при доказательстве формулы $F_R \mathbb{G}, C^-$. Построим взаимно-однозначное соответствие между ними. Обозначим его ψ .

Элементу φ_0 сопоставим элемент K_D^0 и наоборот, т.е. положим $\langle K_D^0 \rangle = \psi \langle \varphi_0 \rangle$, а $\varphi_0 = \psi^{-1} \langle K_D^0 \rangle$.

Обозначим $\Phi_i = \varphi_0, \dots, \varphi_i$ последовательность из первых i шагов грамматического разбора, а $\bar{K}_r = K_D^0, \dots, K_D^r$ последовательность соответствующих им состояний конъюнкта вывода K_D . Положим, что соответствие ψ между последовательностями Φ_i и $\bar{K}_r$ установлено. Пусть $C = \alpha t \tau, \alpha \in V_T^*, t \in V_T, \tau \in V_N$. Элементу $\varphi_{i+1} = \alpha Z, \alpha \in V_T^*, Z \in V_N$ сопоставим тот из элементов $K_D^{r+k}, k \geq 1$, который имеет вид $\forall P(t, s) Q(Z) \dots$, а также все предшествующие ему элементы $K_D^{r+1}, \dots, K_D^{r+k-1}$. Т.е. положим $\langle K_D^{r+1}, \dots, K_D^{r+k} \rangle = \psi \langle \varphi_{i+1} \rangle$, а $\varphi_{i+1} = \psi^{-1} \langle K_D^{r+1}, \dots, K_D^{r+k} \rangle$.

Отметим ряд свойств последовательности K значений конъюнкта вывода K_D , порождаемых при доказательстве формулы $F_R \mathbb{G}, C^-$:

- По построению $F_R \mathbb{G}, C^-$ и в соответствии с принятыми особенностями ее доказательства всякий элемент последовательности K содержит ровно один атом вида $P \mathbb{A} s^-$, либо $R \mathbb{A} s^-$, либо $PT \mathbb{A}^-$, либо $RT \mathbb{A}^-$. Исключение составляет последний шаг доказательства.
- Порождение элементов последовательности K детерминировано по построению формулы $F_R \mathbb{G}, C^-$ и в силу детерминизма анализа, заданного регулярной грамматикой.

Продолжим рассмотрение ПОФ 2.1.1. Покажем соответствие между грамматическим разбором исходной строки $C = 80.7$ в грамматике десятичных

дробей G и последовательностью конъюнктов K_D формулы 2.1.1. Вывод цепочки $C = 80.7$ в грамматике G имеет вид $A \Rightarrow 8A \Rightarrow 80A \Rightarrow 80.B \Rightarrow 80.7$. Соответствие между последовательностью вывода этой строки и состояниями базы U_B ПОФ при доказательстве формулы $F_R \mathbf{G}, C$ приведено в таблице 2.1.

Соответствие между последовательностью грамматического разбора цепочки 80.7 и состояниями конъюнкта вывода ПОФ

Таблица 2.1

№	Шаг разбора	Состояние конъюнкта
1	$\varphi_0 = A$	$K_D^0 = \exists u : u = 1, P(8, u), Q(A)$
2	$\varphi_1 = 8A$	$K_D^1 = \exists u : u = 1, R(8, u), Q(A)$ $K_D^2 = \exists u : u = 2, P(0, u), Q(A)$
3	$\varphi_2 = 80A$	$K_D^3 = \exists u : u = 2, R(0, u), Q(A)$ $K_D^4 = \exists u : u = 3, P(., u), Q(A)$
4	$\varphi_3 = 80.B$	$K_D^5 = \exists u : u = 3, R(., u), Q(B)$ $K_D^6 = \exists u : u = 4, P(7, u), PT(\#), Q(B)$
5	$\varphi_4 = 80.7$	$K_D^7 = \exists u : u = 4, RT(\#)$ $K_D^8 = \exists u : u = 4, F$

В ходе доказательства базы индукции и индуктивного перехода рассматриваются следующие возможные случаи:

- а. ни один шаг грамматического вывода невозможен;
- б. текущий шаг грамматического вывода возможен, он успешно завершает вывод;
- с. текущий шаг грамматического вывода возможен, за ним, по меньшей мере, будет еще один шаг вывода.

База индукции. Пусть $\varphi_0 = S$. Рассмотрим случаи а, б, с.

Случай (а).

1. Пусть ни один шаг вывода невозможен, при этом $\varphi_0 = S$, $K_D^0 = \exists u : u = 1, P(c_1, u), Q(S)$, $C = c_1 c_2 \dots$. Это означает, что грамматика G не имеет правила вида $S \rightarrow c_1 Y$ и в части U_Q формулы $F_R(G, C)$ нет ни одного вопроса вида $\forall s : P(c_1, s), Q(S) \exists : R(c_1, s), Q(Y)$. А это значит, что и вывод цепочки C , и доказательство формулы прекратятся.

Проиллюстрируем данный случай примером. Модифицируем рассмотренную выше грамматику десятичных дробей $G = \langle V_T, V_N, P, S \rangle$ так, чтобы имел место рассматриваемый случай: $V_N = \{A, B, C\}$, $V_T = \{., \cdot\}$, $S = A$, $P = \{A \rightarrow \cdot A \rightarrow \cdot B \quad B \rightarrow \cdot C \quad C \rightarrow \cdot C \rightarrow \cdot\}$, $C = .0$ - входная цепочка. Тогда формула $F_R(G, C) = U_B U_Q$ имеет вид:

$$F_R(G, C) = \exists u : u = 1, P(., u), Q(A) \left\{ \begin{array}{l} \forall s : P(\cdot, s), PT(\#), Q(A) \exists : RT(\#) \\ \forall s : P(\cdot, s), Q(A) \exists : R(\cdot, s), Q(B) \\ \forall s : P(., s), Q(B) \exists : R(., s), Q(C) \\ \forall s : P(\cdot, s), Q(C) \exists : R(\cdot, s), Q(C) \\ \forall s : P(\cdot, s), PT(\#), Q(C) \exists : RT(\#) \\ \forall RT(\#) \exists : \mathbf{F} \\ \forall s : s = 1, R(., s) \exists : s = 2, P(0, s), PT(\#) \end{array} \right.$$

Очевидно, $C \notin L(G)$ и нет ни одного вопроса, который мог бы быть выбран для применения правила ω для базы $U_B = K_D^0 = \exists u : u = 1, P(., u), Q(A)$

2. Пусть ни один шаг лексического анализа невозможен, при этом $\varphi_0 = S$, $C = c_1$, $U_0 = \exists u : u = 1, P(c_1, u), PT(\#), Q(S)$. Это означает, что грамматика G не имеет правила вида $S \rightarrow c_1$ и в части U_Q формулы $F_R(G, C)$ нет ни одного вопроса вида $\forall s : P(c_1, s), PT(\#), Q(S) \exists : RT(\#)$. А это значит, что и вывод цепочки C , и доказательство формулы прекратятся.

Проиллюстрируем данный случай примером. Модифицируем рассмотренную выше грамматику десятичных дробей $G = \langle V_T, V_N, P, S \rangle$ так, чтобы имел место рассматриваемый случай: $V_N = \{A, B, C\}$, $V_T = \{., \cdot\}$, $S = A$, $P = \{A \rightarrow \cdot A \rightarrow \cdot B \quad B \rightarrow \cdot C \quad C \rightarrow \cdot C \rightarrow \cdot\}$, $C = .$ - исходная цепочка. Тогда формула $F_R(G, C) = U_B U_Q$ имеет вид:

$$F_R(G, C) = \exists u : u = 1, P(., u), PT(\#), Q(A) \left\{ \begin{array}{l} \forall s : P(u, s), PT(\#), Q(A) \exists RT(\#) \\ \forall s : P(u, s), Q(A) \exists R(u, s), Q(B) \\ \forall s : P(., s), Q(B) \exists R(., s), Q(C) \\ \forall s : P(u, s), Q(C) \exists R(u, s), Q(C) \\ \forall s : P(u, s), PT(\#), Q(C) \exists RT(\#) \\ \forall RT(\#) \exists \mathbf{F} \end{array} \right.$$

Очевидно, $C \notin L(G)$ и нет ни одного вопроса, который мог бы быть выполнен в состоянии $U_B = K_D^0 = \exists u : u = 1, P(., u), PT(\#), Q(A)$

Справедливо и обратное. Если вывод формулы прекращен в состоянии K_D^0 конъюнкта вывода K_D , то по построению формулы $F_R(G, C)$ нет правила грамматики, которое можно было бы применить для разбора символа c_1 .

Случай (b). Пусть текущий шаг разбора возможен и он завершает разбор, то есть $\varphi_0 = S$, $K_D^0 = \exists u : u = 1, P(c_1, u), PT(\#), Q(S)$, $\varphi_1 = c_1$, и тогда $C = c_1$. Значит, примененное для разбора правило грамматики имеет вид $S \rightarrow c_1$ и среди вопросов ПОФ имеет место вопрос вида $\forall s : P(c_1, s), PT(\#), Q(S) \exists : RT(\#)$. Этот вопрос будет выполнен, новое состояние конъюкта вывода примет вид: $K_D^0 = \exists u : u = 1, RT(\#)$. Далее будет выполнен вопрос $\forall RT(\#) \exists : \mathbf{F}$, который завершит доказательство формулы, при этом конъюнкт вывода будет иметь вид $K_D = \exists u : u = 1, \mathbf{F}$. А это значит, что формула $F(G, C)$ доказана.

Проиллюстрируем данный случай примером. Пусть дана грамматика десятичных дробей $G = \langle V_T, V_N, P, S \rangle$, $V_N = \{A, B, C\}$, $V_T = \{u, .\}$, $S = A$, $P = \{A \rightarrow u \quad A \rightarrow uA \quad A \rightarrow .B \quad B \rightarrow uC \quad C \rightarrow uC \quad C \rightarrow u\}$, $C = 8$ - исходная строка. Формула $F_R(G, C)$ имеет вид:

$$F_R(G, C) = \exists u : u = 1, P(8, u), PT(\#), Q(A) \left\{ \begin{array}{l} \forall s : P(u, s), PT(\#), Q(A) \exists : RT(\#) \\ \forall s : P(u, s), Q(A) \exists : R(u, s), Q(A) \\ \forall s : P(., s), Q(A) \exists : R(., s), Q(B) \\ \forall s : P(u, s), Q(B) \exists : R(u, s), Q(C) \\ \forall s : P(u, s), Q(C) \exists : R(u, s), Q(C) \\ \forall s : P(u, s), PT(\#), Q(C) \exists : RT(\#) \\ \forall RT(\#) \exists \mathbf{F} \end{array} \right.$$

Вывод данной цепочки имеет вид $A \Rightarrow 8$, а последовательность состояний конъюнкта вывода при ее доказательстве имеет вид

$$K_D^0 = \exists u : u = 1, P(8, u), PT(\#), Q(A),$$

$$K_D^1 = \exists u : u = 1, RT(\#),$$

$$K_D^2 = \exists u : u = 1, \mathbf{F}.$$

Справедливо и обратное. Если доказательство формулы $F_R(G, C)$ осуществляется последовательным выполнением приведенных выше вопросов, то по ее построению имеется правило грамматики G вида $S \rightarrow c_1$. Применение этого правила позволит успешно завершить лексический анализ на шаге $\varphi_1 = c_1$.

Случай (с). Текущий шаг разбора возможен, за ним, по меньшей мере, будет еще один шаг разбора, то есть $C = c_1 c_2 \dots c_n, n > 1$, $K_D^0 = \exists u : u = 1, P(c_1, u), Q(S)$, $\varphi_0 = S$, $\varphi_1 = c_1 Y, Y \in V_N$. Шаг $\varphi_1 = c_1 Y, Y \in V_N$ получен путем применения правила вида $S \rightarrow c_1 Y$. Согласно пункту 2.1 и пункту 3.1 алгоритма, среди вопросов U_Q формулы $F_R(G, C)$ имеют место вопросы $\forall s : P(c_1, s), Q(S) \exists : R(c_1, s), Q(Y)$ и $\forall s : s = 1, R(c_1, s) \exists : s = 2, P(c_2, s)$. Эти вопросы будут последовательно выполнены. Новое состояние конъюнкта вывода K_D^2 примет вид $\exists u : u = 2, P(c_2, u), Q(Y)$. Справедливо и обратное, если при доказательстве последовательно была применена рассмотренная выше

пара вопросов, то по построению формулы $F_R(G, C)$ имеется правило грамматики вида $S \rightarrow c_1 Y$.

Проиллюстрируем данный случай примером. Пусть дана грамматика десятичных дробей $G = \langle V_T, V_N, P, S \rangle$, где $V_N = \{A, B, C\}$, $V_T = \{u, .\}$, $S = A$, $P = \{A \rightarrow u, A \rightarrow uA, A \rightarrow .B, B \rightarrow uC, C \rightarrow uC, C \rightarrow u\}$ и $C = .9$ - исходная строка. Формула $F_R(G, C)$ имеет вид:

$$F_R(G, C) = \exists u : u = 1, P(., u), Q(A) \left\{ \begin{array}{l} \forall s : P(u, s), PT(\#), Q(A) \exists : RT(\#) \\ \forall s : P(u, s), Q(A) \exists : R(u, s), Q(A) \\ \forall s : P(., s), Q(A) \exists : R(., s), Q(B) \\ \forall s : P(u, s), Q(B) \exists : R(u, s), Q(C) \\ \forall s : P(u, s), Q(C) \exists : R(u, s), Q(C) \\ \forall s : P(u, s), PT(\#), Q(C) \exists : RT(\#) \\ \forall RT(\#) \exists : \mathbf{F} \\ \forall s : s = 1, R(., s) \exists : s = 2, P(9, s), PT(\#) \end{array} \right.$$

Вывод данной цепочки имеет вид $A \Rightarrow .B \Rightarrow \dots$, а последовательность состояний конъюнкта вывода формулы $F_R(G, C)$ при ее доказательстве имеет вид, $K_D^0 = \exists u : u = 1, P(8, u), PT(\#), Q(A)$, $K_D^1 = \exists u : u = 1, R(9, u), Q(A)$.

Индуктивный переход. Пусть $\Phi_i = \varphi_0, \dots, \varphi_i$ и $\bar{K}_r = K_D^0, \dots, K_D^r$.

Случай (а). Справедливость случая (а) доказывается точно так же, как и для базы индукции.

Случай (б). Текущий шаг разбора возможен, он завершает разбор, то есть $\varphi_i = c_1 c_2 \dots c_{n-1} Z, Z \in V_N$, $\varphi_{i+1} = c_1 \dots c_n$. Тогда согласно пункту 3.2 алгоритма $K_D^r = \exists u : u = n, P(c_n, u), PT(\#), Q(Z)$.

Элемент вывода получен применением правила $Z \rightarrow c_n$. Тогда согласно пункту 2.2 и пункту 2.3 среди вопросов U_Q формулы $F_R(G, C)$ имеют место вопросы $\forall s : P(c_n, s), PT(\#), Q(Z) \exists : RT(\#)$ и $\forall : RT(\#) \exists : \mathbf{F}$. Эти вопросы будут

последовательно выполнены, новое состояние конъюнкта вывода примет вид $\exists u : u = n, \mathbf{F}$. Это означает, что формула $F_R(G, C)$ доказана.

Справедливо и обратное. Завершение последовательности вывода формулы $F_R(G, C)$ элементами $K_D^r, \dots, K_D^{r+k}$, где $K_D^{r+k} = \mathbf{F}$ означает принадлежность одному из элементов последовательности $K_D^r, \dots, K_D^{r+k}$ атома $P(c_n, n)$, а это, свою очередь, означает применение правила грамматики вида $Z \rightarrow c_n$, которое и завершает разбор.

Проиллюстрируем случай примером. Пусть дана грамматика десятичных дробей $G = \langle V_T, V_N, P, S \rangle$, где $V_N = \{A, B, C\}$, $V_T = \{u, .\}$, $S = A$, $P = \{A \rightarrow u, A \rightarrow uA, A \rightarrow .B, B \rightarrow uC, C \rightarrow uC, C \rightarrow u\}$ и $C = .98$ - исходная строка. Формула $F_R(G, C)$ имеет вид:

$$F_R(G, C) = \exists u : u = 1, P(., u), Q(A) \left\{ \begin{array}{l} \forall s : P(u, s), PT(\#), Q(A) \exists : RT(\#) \\ \forall s : P(u, s), Q(A) \exists : R(u, s), Q(A) \\ \forall s : P(., s), Q(A) \exists : R(., s), Q(B) \\ \forall s : P(u, s), Q(B) \exists : R(u, s), Q(C) \\ \forall s : P(u, s), Q(C) \exists : R(u, s), Q(C)) \\ \forall s : P(u, s), PT(\#), Q(C) \exists : RT(\#) \\ \forall RT(\#) \exists : \mathbf{F} \\ \forall s : s = 1, R(., s) \exists : s = 2, P(9, s) \\ \forall s : s = 2, R(9, s) \exists : s = 3, P(8, s), PT(\#) \end{array} \right.$$

Вывод данной цепочки имеет вид $A \Rightarrow \dots \Rightarrow .9C \Rightarrow .98$, а последовательность состояний конъюнкта вывода формулы $F_R(G, C)$ при ее доказательстве имеет вид

$$K_D^r = \exists u : u = 2, P(9, u), Q(C),$$

$$K_D^{r+1} = \exists u : u = 2, R(9, u), Q(C),$$

$$K_D^{r+2} = \exists u : u = 3, P(8, u), PT(\#), Q(C),$$

$$K_D^{r+3} = \exists u : u = 3, RT(\#),$$

$$K_D^{r+4} = \exists u : u = 3, F.$$

Случай (с). Справедливость случая (с) доказывается точно так же, как и для базиса индукции.

Таким образом, $\langle K_D^{r+1}, \dots, K_D^{r+k} \rangle = \psi \circ \varphi_{i+1}^{-1}$, а $\varphi_{i+1} = \psi^{-1} \circ \langle K_D^{r+1}, \dots, K_D^{r+k} \rangle$.

Существование взаимнооднозначного отображения ψ между последовательностью элементов лексического анализа Φ при разборе исходной строки S и последовательностью состояний конъюнктов вывода при доказательстве формулы $F_R(G, C)$ и означает справедливость утверждения 1.

2.2. Распознающие позитивно-образованные формулы для LL(1)-грамматик

Алгоритм построения позитивно-образованной формулы F для LL(1)-грамматики похож на алгоритм из п.2.1. Он также включает начальную установку, обработку правил грамматики G и обработку элементов входной цепочки S .

Начальная установка. Положим $U_B = \exists u : u = 1, P \circlearrowleft_{1,u} \bar{Q} \circlearrowright \bar{EOP}()$, а $U_Q = \forall : PT \circlearrowleft \bar{Q} \circlearrowright \bar{EOP}() \exists : F$.

Обработка правил грамматики G .

1. Для каждого правила вида $Z \rightarrow a\gamma$, где $Z \in V_N, a \in V_T, \gamma \in V^*$, пополним U_Q вопросом $\forall s : P \circlearrowleft_{1,s} \bar{Q} \circlearrowright \exists : R \circlearrowleft_{1,s} \bar{X}$.

2. Обозначим $DS \circlearrowleft_{1,\gamma}$ множество направляющих символов правила $Z \rightarrow \gamma$ (см. приложение 1 к настоящей работе). Для каждого правила вида $Z \rightarrow \gamma$, где $Z \in V_N, \gamma = A\beta, A \in V_N, \beta \in V^*$, и каждого $t \in DS \circlearrowleft_{1,\gamma}$ пополним U_Q вопросом $\forall s : P \circlearrowleft_{1,s} \bar{Q} \circlearrowright \exists : P \circlearrowleft_{1,s} \bar{Q}(A), \beta$, если $t \neq \#$ и вопросом $\forall s : PT \circlearrowleft \bar{Q} \circlearrowright \exists : PT \circlearrowleft \bar{Q}(A), \beta$, если $t = \#$.

3. Для каждого терминального символа $t \in V_T$ и правила $Z \rightarrow \alpha t \beta, Z \in V_N, \alpha \in V^+, \beta \in V^*$ пополним U_Q вопросом $\forall s: P \# s, \bar{R} T \# \exists: R \# s$.

4. Введем символ конца исходной цепочки $\#, \# \notin (V_T \cup V_N)$. Для каждого правила вида $Z \rightarrow \varepsilon$ и элемента $t \in DS \# \varepsilon$ пополним U_Q вопросом $\forall s: P \# s, \bar{Q} \# \exists: P \# s$, если $t \neq \#$ и вопросом $\forall s: P T \# \bar{Q} \# \exists: P T \#$, если $t = \#$.

Обработка элементов входной цепочки C .

1. Для каждого символа входной цепочки $c_i, i < n$, пополним U_Q вопросом $\forall s: s = i, R \# i, s \bar{\exists}: s = i + 1, P \# i + 1, s$.
2. Пополним U_Q вопросом $\forall s: s = n, R \# n, s \bar{\exists}: P T \#$.

Построенную таким образом формулу обозначим $F_{LL} \# C$ и назовем распознающей позитивно-образованной формулой.

Пример. Пусть дана LL(1)-грамматика $G = \langle V_T, V_N, P, S \rangle$, где

$$V_T = \{+, *, i\}, V_N = \{A, B, C, D, E, F\}, S = A,$$

$$P = \left\{ \begin{array}{lllll} A \rightarrow CB & B \rightarrow +CB & B \rightarrow \varepsilon & C \rightarrow ED & D \rightarrow *ED \\ D \rightarrow \varepsilon & E \rightarrow iF & F \rightarrow iF & F \rightarrow \varepsilon & \end{array} \right\}.$$

Подробно рассмотрим шаги алгоритма построения распознающей ПОФ $F_{LL} \# C$ для решения задачи разбора строки $C = i + i$:

$$\text{Начальная установка: } U_B = \exists u: u = 1, P \# u, \bar{Q} \# \bar{EOP}()$$

$$U_Q = \forall: P T \# \bar{EOP}() \exists: F$$

Обработка правил грамматики:

1. Для правил грамматики $B \rightarrow +CB$, $D \rightarrow *ED$, $E \rightarrow iF$, $F \rightarrow iF$ пополним U_Q вопросами: $\forall s: P(+, s), Q(B) \exists: R(+, s), Q(C), Q(B)$

$$\forall s: P(*, s), Q(D) \exists: R(*, s), Q(E), Q(D)$$

$$\forall s : P(i, s), Q(E) \exists : R(i, s), Q(F)$$

$$\forall s : P(i, s), Q(F) \exists : R(i, s), Q(F)$$

2. Для правил грамматики $A \rightarrow CB$ и $C \rightarrow ED$ построим множества направляющих символов: $DS \clubsuit, CB \equiv \{i\}$, $DS \clubsuit, ED \equiv \{i\}$ и пополним U_Q соответствующими вопросами: $\forall s : P(i, s), Q(A) \exists : P(i, s), Q(C), Q(B)$

$$\forall s : P(i, s), Q(C) \exists : P(i, s), Q(E), Q(D)$$

3. Т.к. в грамматике нет правил вида $Z \rightarrow \alpha t \beta, Z \in V_N, \alpha \in V^+, \beta \in V^*$, то данный пункт пропускается.

4. Для правил $B \rightarrow \varepsilon$, $D \rightarrow \varepsilon$, $F \rightarrow \varepsilon$ построим множества направляющих символов: $DS \spadesuit, \varepsilon \equiv \{\#\}$, $DS \spadesuit, \varepsilon \equiv \{+, \#\}$, $DS \spadesuit, \varepsilon \equiv \{+, *, \#\}$ и пополним U_Q соответствующими вопросами: $\forall s : PT(\#), Q(B) \exists : PT(\#)$

$$\forall s : P(+, s), Q(D) \exists : P(+, s)$$

$$\forall s : PT(\#), Q(D) \exists : PT(\#)$$

$$\forall s : P(+, s), Q(F) \exists : P(+, s)$$

$$\forall s : P(*, s), Q(F) \exists : P(*, s)$$

$$\forall s : PT(\#), Q(F) \exists PT(\#)$$

Обработка элементов входной цепочки $C = i + i$:

1. Пополним U_Q вопросами $\forall s : s = 1, R(i, s) \exists : s = 2, P(+, s)$

$$\forall s : s = 2, R(+, s) \exists : s = 3, P(i, s)$$

2. Пополним U_Q последним вопросом $\forall s : s = 3, R(i, s) \exists : PT(\#)$

Таким образом, распознающая ПОФ для грамматики G будет иметь следующий вид:

$$\begin{array}{l}
\forall s : P(+, s), Q(B) \exists : R(+, s), Q(C), Q(B) \\
\forall s : P(*, s), Q(D) \exists : R(*, s), Q(E), Q(D) \\
\forall s : P(i, s), Q(E) \exists : R(i, s), Q(F) \\
\forall s : P(i, s), Q(F) \exists : R(i, s), Q(F) \\
\forall s : P(i, s), Q(A) \exists : P(i, s), Q(C), Q(B) \\
\forall s : P(i, s), Q(C) \exists : P(i, s), Q(E), Q(D) \\
\forall s : PT(\#), Q(B) \exists : PT(\#) \\
\forall s : P(+, s), Q(D) \exists : P(+, s) \\
\forall s : PT(\#), Q(D) \exists : PT(\#) \\
\forall s : P(+, s), Q(F) \exists : P(+, s) \\
\forall s : P(*, s), Q(F) \exists : P(*, s) \\
\forall s : PT(\#), Q(F) \exists : PT(\#) \\
\forall s : s = 1, R(i, s) \exists : s = 2, P(+, s) \\
\forall s : s = 2, R(+, s) \exists : s = 3, P(i, s) \\
\forall s : s = 3, R(i, s) \exists : PT(\#) \\
\forall PT(\#), EOP() \exists : \mathbf{F}
\end{array}
\quad \exists u : u = 1, P(i, u), Q(A), EOP() \quad (2.2.1)$$

Рассмотрим вопрос связи доказательства распознающей ПОФ с выводом цепочки в LL(1)-грамматике, для этого докажем утверждение.

Утверждение 2. Для всякой LL(1)-грамматики G и исходной цепочки C формула $F_{LL} \langle G, C \rangle$ доказуема тогда и только тогда, когда $C \in L \langle G \rangle$.

Доказательство. Рассмотрим последовательность $\Phi = \varphi_0, \dots, \varphi_n$ грамматического разбора цепочки C в грамматике G и последовательность $U = U_0, \dots, U_m$ состояний базы U_B при доказательстве формулы $F_{LL} \langle G, C \rangle$. Построим взаимнооднозначное соответствие между ними. Обозначим его χ .

Элементу φ_0 сопоставим элемент U_0 и наоборот, т.е. положим $\langle U_0 \rangle = \chi \langle \varphi_0 \rangle$, а $\varphi_0 = \chi^{-1} \langle U_0 \rangle$.

Обозначим $\Phi_i = \varphi_0, \dots, \varphi_i$ последовательность из первых i шагов грамматического разбора, а $\bar{U}_r = U_0, \dots, U_r$ последовательность

соответствующих им состояний базы данных U_B . Положим, что соответствие χ между последовательностями Φ_i и $\bar{U}_r$ установлено. Пусть $C = \alpha t \tau, \alpha \in V_T^*, t \in V_T, \tau \in V_T^*$. Элементу $\varphi_{i+1} = \alpha Z \beta, \alpha \in V_T^*, Z \in V_N, \beta \in V^*$ сопоставим тот из элементов $U_{r+k}, k \geq 1$, который имеет вид $\forall s: P(t, s)Q(Z)...$ или $\forall: PT(t)Q(Z)...$, а также все предшествующие ему элементы $U_{r+1}, \dots, U_{r+k-1}$. Т.е. положим $\langle U_{r+1}, \dots, U_{r+k} \rangle = \chi \Phi_{i+1}$, а $\varphi_{i+1} = \chi^{-1} \langle U_{r+1}, \dots, U_{r+k} \rangle$.

Отметим ряд свойств последовательности U состояний базы U_B , порождаемых при доказательстве формулы $F_{LL} \mathbb{G}, C$:

- По построению $F_{LL} \mathbb{G}, C$ и в соответствии с принятыми особенностями ее доказательства всякий элемент последовательности U (конъюнкт K_D) содержит ровно один атом вида $P \mathbb{A} s$, либо $R \mathbb{A} s$, либо $PT \mathbb{A}$. Исключения составляет последний шаг доказательства;
- Порождение элементов последовательности U детерминировано по построению формулы $F_{LL} \mathbb{G}, C$ и в силу детерминизма анализа, заданного LL(1)-грамматикой.

Вернемся к рассмотрению ПОФ 2.2.1. Покажем соответствие между грамматическим разбором исходной строки $C = i + i$ в грамматике арифметических выражений G и последовательностью конъюнктов K_D формулы 2.2.1. Вывод цепочки $C = i + i$ в грамматике G имеет вид $A \Rightarrow CB \Rightarrow EDB \Rightarrow iFDB \Rightarrow iDB \Rightarrow iB \Rightarrow i + CB \Rightarrow i + EDB \Rightarrow i + iFDB \Rightarrow i + iDB \Rightarrow i + iB \Rightarrow i + i$. Соответствие между последовательностью вывода этой строки и состояниями базы данных ПОФ при доказательстве формулы $F_{LL} \mathbb{G}, C$ приведено в таблице 2.2.

Соответствие между последовательностью вывода цепочки в LL(1)-грамматике и состояниями ПОФ **Таблица 2.2.**

№	Шаг разбора	Состояние конъюнкта
1	$\varphi_0 = A$	$K_D^0 = \exists u : u = 1, P(i, u), Q(A), EOP()$
2	$\varphi_1 = CB$	$K_D^1 = \exists u : u = 1, P(i, u), Q(C), Q(B), EOP()$
3	$\varphi_2 = EDB$	$K_D^2 = \exists u : u = 1, P(i, u), Q(E), Q(D), Q(B), EOP()$
4	$\varphi_3 = iFDB$	$K_D^3 = \exists u : u = 1, R(i, u), Q(F), Q(D), Q(B), EOP()$ $K_D^4 = \exists u : u = 2, P(+, u), Q(F), Q(D), Q(B), EOP()$
5	$\varphi_4 = iDB$	$K_D^5 = \exists u : u = 2, P(+, u), Q(D), Q(B), EOP()$
6	$\varphi_5 = iB$	$K_D^6 = \exists u : u = 2, P(+, u), Q(B), EOP()$
7	$\varphi_6 = i + CB$	$K_D^7 = \exists u : u = 2, R(+, u), Q(C), Q(B), EOP()$ $K_D^8 = \exists u : u = 3, P(i, u), Q(C), Q(B), EOP()$
8	$\varphi_7 = i + EDB$	$K_D^9 = \exists u : u = 3, P(i, u), Q(E), Q(D), Q(B), EOP()$
9	$\varphi_8 = i + iFDB$	$K_D^{10} = \exists u : u = 3, R(i, u), Q(F), Q(D), Q(B), EOP()$ $K_D^{11} = \exists u : u = 3, PT(\#), Q(F), Q(D), Q(B), EOP()$
10	$\varphi_9 = i + iDB$	$K_D^{12} = \exists u : u = 3, PT(\#), Q(F), Q(D), Q(B), EOP()$
11	$\varphi_{10} = i + iB$	$K_D^{13} = \exists u : u = 3, PT(\#), Q(D), Q(B), EOP()$
12	$\varphi_{11} = i + iB$	$K_D^{14} = \exists u : u = 3, PT(\#), Q(B), EOP()$
13	$\varphi_{12} = i + i$	$K_D^{15} = \exists u : u = 3, PT(\#), EOP()$ $K_D^{16} = \exists u : u = 3, F$

Докажем существование отображения χ методом математической индукции по длине последовательности Φ . При доказательстве базиса индукции и индуктивного перехода будем разбирать следующие случаи:

- a.** Текущий шаг грамматического разбора φ_i является последним его успешным шагом, дальнейший разбор не возможен.
- b.** Текущий шаг грамматического разбора φ_i является предпоследним шагом разбора. Следующий шаг грамматического разбора успешно завершает разбор.
- c.** За текущим шагом грамматического разбора φ_i будет, по меньшей мере, еще один успешный шаг φ_{i+1} .

В каждом из этих случаев будем учитывать правила построения распознающей позитивно-образованной формулы $F_{LL} \in C_{-}^{\sim}$.

База индукции. Рассмотрим случаи а, б и с. Пусть $\varphi_0 = S$,
 $U_B = \exists u : u = 1, P(c_1, u), Q(c_1, u), EOP()$.

Случай (а).

Пусть ни один шаг грамматического разбора не возможен. Это означает, что в части U_Q формулы $F_{LL}(\mathbf{G}, C)$ по ее построению нет ни одного вопроса вида $\forall s : P(c_1, s)Q(S)\exists \dots$. Следовательно, доказательство формулы $F_{LL}(\mathbf{G}, C)$ прекратится. Если доказательство формулы прекращено в состоянии U_0 базы U_B , то по построению формулы $F_{LL}(\mathbf{G}, C)$ нет правила грамматики $G : S \rightarrow \gamma$, которое можно было бы применить для разбора символа c_1 .

Проиллюстрируем данный случай примером. Рассмотрим грамматику $G = \langle V_T, V_N, P, S \rangle$, где: $V_T = \{+, i\}$, $V_N = \{A, B, C, D\}$, $S = A$,
 $P = A \rightarrow CB \quad B \rightarrow +CB \quad B \rightarrow \varepsilon \quad C \rightarrow iD \quad D \rightarrow iD \quad D \rightarrow \varepsilon$ и входную цепочку $C = +$. Тогда формула $F_{LL}(G, C) = U_B \cup U_Q$ имеет вид:

$$\exists u : u = 1, P(+, u), Q(A), EOP() \left\{ \begin{array}{l} \forall s : P(+, s), Q(B) \exists : R(+, s), Q(C), Q(B) \\ \forall s : P(i, s), Q(C) \exists : R(i, s), Q(D) \\ \forall s : P(i, s), Q(D) \exists : R(i, s), Q(D) \\ \forall s : P(i, s), Q(A) \exists : P(i, s), Q(C), Q(B) \\ \forall s : PT(\#), Q(B) \exists : PT(\#) \\ \forall s : P(+, s), Q(D) \exists : P(+, s) \\ \forall s : PT(\#), Q(D) \exists : PT(\#) \\ \forall s : s = 1, R(+, s) \exists : PT(\#) \\ \forall PT(\#), EOP() \exists : \mathbf{F} \end{array} \right.$$

Очевидно, что $C \notin L(G)$ и нет ни одного вопроса, который мог бы быть выбран для применения правила ω для базы $U_B = K_D^0 = \exists u : u = 1, P(+, u), Q(A), EOP()$

Случай (b).

Пусть $\varphi_1 = c_1$, $C = c_1$, примененное для разбора правило грамматики G имеет вид $S \rightarrow c_1$. По построению формулы $F_{LL} \mathbf{G}, C$ согласно пункту 1 обработки правил грамматики G среди вопросов U_Q формулы $F_{LL} \mathbf{G}, C$ имеет место вопрос $\forall s: P \mathbf{A}_1, s, Q \mathbf{G} \exists: R \mathbf{A}_1, s$. Этот вопрос будет выполнен. Новое состояние базы U_B имеет вид $\exists u: u=1, R \mathbf{A}_1, u, EOP()$. Согласно пункту 2 обработки элементов исходной строки при построении формулы $F_{LL} \mathbf{G}, C$ среди вопросов U_Q формулы $F_{LL} \mathbf{G}, C$ имеют место вопросы $\forall PT \mathbf{A}, EOP() \exists F, \forall s: s=1, R \mathbf{A}_1, s \exists PT \mathbf{A}$ и они будут последовательно выполнены. Это означает, что формула $F_{LL} \mathbf{G}, C$ доказана. Справедливо и обратное. Если доказательство формулы $F_{LL} \mathbf{G}, C$ осуществлено выполнением указанных выше трех вопросов, то по ее построению имеется правило грамматики G вида $S \rightarrow c_1$. Применение этого правила приведет к успешному завершению грамматического разбора на шаге $\varphi_1 = c_1$.

Проиллюстрируем данный случай примером. Рассмотрим грамматику $G = \langle V_T, V_N, P, S \rangle$, где: $V_T = \{+, i, *\}$, $V_N = \{A, B, C, D\}$, $S = A$, $P = A \rightarrow * \quad A \rightarrow CB \quad B \rightarrow +CB \quad B \rightarrow \varepsilon \quad C \rightarrow iD \quad D \rightarrow iD \quad D \rightarrow \varepsilon$ и входную цепочку $C = *$. Тогда формула $F_{LL}(G, C) = U_B U_Q$ имеет вид:

$$\exists u : u = 1, P(*, u), Q(A), EOP() \left\{ \begin{array}{l} \forall s : P(*, s), Q(A) \exists : R(*, s) \\ \forall s : P(+, s), Q(B) \exists : R(+, s), Q(C), Q(B) \\ \forall s : P(i, s), Q(C) \exists : R(i, s), Q(D) \\ \forall s : P(i, s), Q(D) \exists : R(i, s), Q(D) \\ \forall s : P(i, s), Q(A) \exists : P(i, s), Q(C), Q(B) \\ \forall s : PT(\#), Q(B) \exists : PT(\#) \\ \forall s : P(+, s), Q(D) \exists : P(+, s) \\ \forall s : PT(\#), Q(D) \exists : PT(\#) \\ \forall s : s = 1, R(*, s) \exists : PT(\#) \\ \forall PT(\#), EOP() \exists : F \end{array} \right.$$

Грамматический вывод данной цепочки имеет вид $A \Rightarrow *$, а последовательность состояний конъюнкта вывода при ее доказательстве имеет вид

$$K_D^0 = \exists u : u = 1, P(*, u), Q(A), EOP(),$$

$$K_D^1 = \exists u : u = 1, R(*, u), EOP(),$$

$$K_D^2 = \exists u : u = 1, PT(\#), EOP(),$$

$$K_D^3 = \exists u : u = 1, F.$$

Случай (с).

Пусть построена цепочка φ_1 . Возможны следующие варианты вида использованного на данном шаге грамматического разбора правила грамматики G :

1. $S \rightarrow c_1 Z \beta, Z \in V_N, \beta \in V^*$. Тогда $C = c_1 t \tau, t = c_2, \tau \in V_T^*$,
 $\varphi_1 = c_1 Z \beta, Z \in V_N, \beta \in V^*$. По построению формулы $F_{LL} \mathbf{G}, C^-$ согласно пункту 1 обработки правил грамматики G и пункту 1 обработки элементов исходной строки среди вопросов U_Q формулы $F_{LL} \mathbf{G}, C^-$ имеют место вопросы $\forall s : P \mathbf{A}_1, s, Q \mathbf{B} \exists R \mathbf{A}_1, s, Q(Z), [\beta]$ и $\forall s : s = 1, R \mathbf{A}_1, s \exists : s = 2, P \mathbf{A}_2, s^-$. Эти вопросы будут

последовательно выполнены. Новое состояние базы данных U_B имеет вид $\exists s : s = 2, P(+, s), Q(Z), [\beta], EOP()$. Справедливо и обратное, если при доказательстве формулы $F_{LL}(G, C)$ была последовательно применена указанная выше пара вопросов, то по ее построению имеется правило грамматики G вида $S \rightarrow c_1 Z \beta$.

Проиллюстрируем данный вариант примером. Рассмотрим грамматику $G = \langle V_T, V_N, P, S \rangle$, где: $V_T = \{+, -, i\}$, $V_N = \{A, B, C, D\}$, $S = A$, $P = A \rightarrow -CB \quad B \rightarrow +CB \quad B \rightarrow \varepsilon \quad C \rightarrow iD \quad D \rightarrow iD \quad D \rightarrow \varepsilon$ и входную цепочку $C = -i$. Тогда формула $F_{LL}(G, C) = U_B \cup U_Q$ имеет вид:

$$\exists u : u = 1, P(-, u), Q(A), EOP() \left\{ \begin{array}{l} \forall s : P(+, s), Q(B) \exists : R(+, s), Q(C), Q(B) \\ \forall s : P(-, s), Q(A) \exists : R(-, s), Q(C), Q(B) \\ \forall s : P(i, s), Q(C) \exists : R(i, s), Q(D) \\ \forall s : P(i, s), Q(D) \exists : R(i, s), Q(D) \\ \forall s : PT(\#), Q(B) \exists : PT(\#) \\ \forall s : P(+, s), Q(D) \exists : P(+, s) \\ \forall s : PT(\#), Q(D) \exists : PT(\#) \\ \forall s : s = 1, R(-, s) \exists : s = 2, P(i, s) \\ \forall s : s = 2, R(i, s) \exists : PT(\#) \\ \forall PT(\#), EOP() \exists : F \end{array} \right.$$

Вывод данной цепочки имеет вид $A \Rightarrow -CB \Rightarrow \dots$, а последовательность состояний конъюнкта вывода формулы $F_{LL}(G, C)$ при ее доказательстве имеет следующий вид,

$$K_D^0 = \exists u : u = 1, P(-, u), Q(A), EOP(),$$

$$K_D^1 = \exists u : u = 1, R(-, u), Q(C), Q(B), EOP() \dots$$

2. $S \rightarrow Z\beta, Z \in V_N, \beta \in V^*$ и $c_1 \in DS(S, ZB)$. Тогда $C = t\tau, t = c_1, \tau \in V_T^*$, $\varphi_1 = Z\beta, Z \in V_N, \beta \in V^*$. По построению формулы $F_{LL} \mathbf{G}, C$ согласно пункту 2 обработки правил грамматики G среди вопросов U_Q формулы $F_{LL} \mathbf{G}, C$ имеет место вопрос $\forall s: P \mathbf{A}_1, s, Q \mathbf{S} \exists P \mathbf{A}_1, s, Q(Z), [\beta]$ и он будет выполнен. Новое состояние базы данных U_B имеет вид $\forall s: P \mathbf{A}_1, s, Q(Z), [\beta], EOP()$. Справедливо и обратное.

Проиллюстрируем данный вариант примером. Рассмотрим грамматику $G = \langle V_T, V_N, P, S \rangle$, где: $V_T = \{+, i\}$, $V_N = \{A, B, C, D\}$, $S = A$, $P = A \rightarrow CB \ B \rightarrow +CB \ B \rightarrow \varepsilon \ C \rightarrow iD \ D \rightarrow iD \ D \rightarrow \varepsilon$ и входную цепочку $C = i + i$. Тогда формула $F_{LL}(G, C) = U_B \ U_Q$ имеет вид:

$$\exists u : u = 1, P(i, u), Q(A), EOP() \left\{ \begin{array}{l} \forall s : P(+, s), Q(B) \exists : R(+, s), Q(C), Q(B) \\ \forall s : P(i, s), Q(C) \exists : R(i, s), Q(D) \\ \forall s : P(i, s), Q(D) \exists : R(i, s), Q(D) \\ \forall s : P(i, s), Q(A) \exists : P(i, s), Q(C), Q(B) \\ \forall s : PT(\#), Q(B) \exists : PT(\#) \\ \forall s : P(+, s), Q(D) \exists : P(+, s) \\ \forall s : PT(\#), Q(D) \exists : PT(\#) \\ \forall s : s = 1, R(i, s) \exists : s = 2, P(+, s) \\ \forall s : s = 2, R(+, s) \exists : s = 3, P(i, s) \\ \forall s : s = 3, R(i, s) \exists : PT(\#) \\ \forall PT(\#), EOP() \exists : \mathbf{F} \end{array} \right.$$

Вывод данной цепочки имеет вид $A \Rightarrow CB \Rightarrow \dots$, а последовательность состояний конъюнкта вывода при ее доказательстве имеет вид

$$K_D^0 = \exists u : u = 1, P(i, u), Q(A), EOP(),$$

$$K_D^1 = \exists u : u = 1, P(i, u), Q(C), Q(B), EOP() \dots$$

3. $S \rightarrow Z_1 Z_2 \dots Z_h c_1 \beta$, $Z_1 \in V_N$, $Z_2 \dots Z_h \in V_N^*$, $\beta \in V^*$. Тогда

$$C = c_1 c_2 \tau, \tau \in V_T^*, \varphi_1 = Z_1 Z_2 \dots Z_h c_1 \beta, Z_1 \in V_N, Z_2 \dots Z_h \in V_N^*, \beta \in V^*. По$$

построению формулы $F_{LL} \langle G, C \rangle$ согласно пунктам 2 обработки

правил грамматики G среди вопросов U_Q формулы $F_{LL} \langle G, C \rangle$ имеет

место вопрос

$$\forall s : P \langle \varphi_1, s \rangle, Q \langle \exists P \langle \varphi_1, s \rangle, Q(Z_1), Q(Z_2), \dots, Q(Z_h), RT(c_1), [\beta] \rangle. \quad \text{Этот}$$

вопрос будет выполнен. Новое состояние базы данных U_B имеет вид

$$\forall s : P \langle \varphi_1, s \rangle, Q(Z_1), Q(Z_2), \dots, Q(Z_h), RT(c_1), [\beta], EOP(). \text{ Справедливо и}$$

обратное.

Проиллюстрируем данный вариант примером. Рассмотрим

грамматику $G = \langle V_T, V_N, P, S \rangle$, где: $V_T = \{ (,), +, i \}$, $V_N = \{ A, B, C, D \}$,

$$S = A, \quad P = \left\{ \begin{array}{l} A \rightarrow E(CB) \quad B \rightarrow +CB \quad B \rightarrow \varepsilon \quad C \rightarrow iD \\ D \rightarrow iD \quad D \rightarrow \varepsilon \quad E \rightarrow \varepsilon \quad E \rightarrow = \end{array} \right\} \quad \text{и входную}$$

цепочку $C = (i)$. Тогда формула $F_{LL}(G, C) = U_B \cup U_Q$ имеет вид на рис.

2.2.2:

Вывод данной цепочки имеет вид $A \Rightarrow E(CB) \Rightarrow (CB) \dots$, а

последовательность состояний конъюнкта вывода при ее

доказательстве имеет вид

$$K_D^0 = \exists u : u = 1, P((, u), Q(A), EOP(),$$

$$K_D^1 = \exists u : u = 1, P((, u), Q(E), RT((), Q(C), Q(B), RT()), EOP()$$

$$K_D^2 = \exists u : u = 1, P((, u), RT((), Q(C), Q(B), RT()), EOP()$$

$$K_D^3 = \exists u : u = 1, R((, u), Q(C), Q(B), RT()), EOP() \dots$$

$$\begin{aligned}
& \exists u : u = 1, P((), u), Q(A), EOP() \\
& \left\{ \begin{aligned}
& \forall s : P(+, s), Q(B) \exists : R(+, s), Q(C), Q(B) \\
& \forall s : P(i, s), Q(C) \exists : R(i, s), Q(D) \\
& \forall s : P(i, s), Q(D) \exists : R(i, s), Q(D) \\
& \forall s : P(=, s), Q(A) \exists : R(=, s) \\
& \forall s : P((), s), Q(A) \exists : P((), s), Q(E), RT((), Q(C), Q(B), RT()) \\
& \forall s : P(=, s), Q(A) \exists : P(=, s), Q(E), RT((), Q(C), Q(B), RT()) \\
& \forall s : P((), s), RT(()) \exists : R((), s) \\
& \forall s : P(), s), RT()) \exists : R(), s) \\
& \forall s : P(=, s), Q(E) \exists : P(=, s) \\
& \forall s : P((), s), Q(E) \exists : P((), s) \\
& \forall s : PT(\#), Q(B) \exists : PT(\#) \\
& \forall s : P(+, s), Q(D) \exists : P(+, s) \\
& \forall s : PT(\#), Q(D) \exists : PT(\#) \\
& \forall s : s = 1, R((), s) \exists : s = 2, P(i, s) \\
& \forall s : s = 2, R(i, s) \exists : s = 3, P(), s) \\
& \forall s : s = 3, R(), s) \exists : PT(\#) \\
& \forall PT(\#), EOP() \exists : F
\end{aligned} \right. \quad (2.2.2)
\end{aligned}$$

Индуктивный переход. Пусть $\Phi_i = \varphi_0, \dots, \varphi_i$ и $\bar{U}_r = U_0, \dots, U_r$.

Справедливость случая (а) доказывается точно так же, как и для базиса индукции.

Случай (b).

Пусть $\varphi_{i+1} = C$. Далее возможны варианты:

1. Пусть $\varphi_i = c_1 \dots c_{n-1} Z, Z \in V_N, U_B = \exists u : u = n, P(_, u), \bar{Q}(_, _, EOP())$ и примененное для разбора правило грамматики G имеет вид $Z \rightarrow c_n$.

По построению формулы $F_{LL}(_, C)$ согласно пункту 1 обработки правил грамматики G и пункту 2 обработки элементов исходной строки среди вопросов U_Q формулы $F_{LL}(_, C)$ имеют место вопросы

$$\forall s : P \langle_n, s \rangle, Q \langle_n \rangle \exists : R \langle_n, s \rangle, \quad \forall s : s = n, R \langle_n, s \rangle \exists PT \langle_n \rangle,$$

$\forall : PT(\#), EOP() \exists : F$ и они будут последовательно выполнены. Это означает, что формула $F_{LL} \langle_n \rangle, C \langle_n \rangle$ доказана.

Проиллюстрируем данный вариант примером. Рассмотрим грамматику $G = \langle V_T, V_N, P, S \rangle$, где: $V_T = \{+, i, ;\}$, $V_N = \{A, B, C\}$, $S = A$, $P = A \rightarrow CB \ B \rightarrow +CB \ B \rightarrow ; \ C \rightarrow i$ и входную цепочку $C = i + i ;$. Тогда формула $F_{LL}(G, C) = U_B U_Q$ имеет вид:

$$\exists u : u = 1, P(i, u), Q(A), EOP() \left\{ \begin{array}{l} \forall s : P(+, s), Q(B) \exists : R(+, s), Q(C), Q(B) \\ \forall s : P(i, s), Q(C) \exists : R(i, s) \\ \forall s : P(;, s), Q(B) \exists : R(;, s) \\ \forall s : P(i, s), Q(A) \exists : P(i, s), Q(C), Q(B) \\ \forall s : s = 1, R(i, s) \exists : s = 2, P(+, s) \\ \forall s : s = 2, R(+, s) \exists : s = 3, P(i, s) \\ \forall s : s = 3, R(i, s) \exists : s = 4, P(;, s) \\ \forall s : s = 4, R(;, s) \exists : PT(\#) \\ \forall PT(\#), EOP() \exists : F \end{array} \right.$$

Вывод данной цепочки имеет вид $A \Rightarrow CB \Rightarrow \dots \Rightarrow i + iB \Rightarrow i + i ;$, а последовательность состояний конъюнкта вывода при ее доказательстве имеет вид

$$K_D^1 = \exists u : u = 1, P(i, u), Q(A), EOP()$$

....

$$K_D^8 = \exists u : u = 4, P(;, u), Q(B), EOP()$$

$$K_D^9 = \exists u : u = 4, R(;, u), EOP()$$

$$K_D^{10} = \exists u : u = 4, PT(\#), EOP()$$

$$K_D^{11} = \exists u : u = 4, F$$

2. Пусть $\varphi_i = c_1 \dots c_n Z, Z \in V_N$, $U_B = \exists u : u = n, PT \langle_n \rangle, Q \langle_n \rangle, EOP()$ и примененное для разбора правило грамматики G имеет вид $Z \rightarrow \varepsilon$.

По построению формулы $F_{LL} \mathbb{G}, C^-$ согласно пункту 4 обработки правил грамматики G и пункту 2 обработки элементов исходной строки среди вопросов U_Q формулы $F_{LL} \mathbb{G}, C^-$ имеют место вопросы $\forall : PT \# , Q \# \exists : PT(\#) , \quad \forall : PT \# , EOP() \exists : F$ и они будут последовательно выполнены. Это означает, что формула $F_{LL} \mathbb{G}, C^-$ доказана.

Проиллюстрируем данный вариант примером. Рассмотрим грамматику $G = \langle V_T, V_N, P, S \rangle$, где: $V_T = \{+, i\}$, $V_N = \{A, B, C\}$, $S = A$, $P = A \rightarrow CB \quad B \rightarrow +CB \quad B \rightarrow \varepsilon \quad C \rightarrow i$ и входную цепочку $C = i + i$. Тогда формула $F_{LL}(G, C) = U_B U_Q$ имеет вид:

$$\exists u : u = 1, P(i, u), Q(A), EOP() \left\{ \begin{array}{l} \forall s : P(+, s), Q(B) \exists : R(+, s), Q(C), Q(B) \\ \forall s : P(i, s), Q(C) \exists : R(i, s) \\ \forall s : P(i, s), Q(A) \exists : P(i, s), Q(C), Q(B) \\ \forall s : PT(\#), Q(B) \exists : PT(\#) \\ \forall s : s = 1, R(i, s) \exists : s = 2, P(+, s) \\ \forall s : s = 2, R(+, s) \exists : s = 3, P(i, s) \\ \forall s : s = 3, R(i, s) \exists : PT(\#) \\ \forall PT(\#), EOP() \exists : F \end{array} \right.$$

Вывод данной цепочки имеет вид $A \Rightarrow CB \Rightarrow \dots \Rightarrow i + iB \Rightarrow i + i$, а последовательность состояний конъюнкта вывода при ее доказательстве имеет вид

$$\begin{aligned} K_D^1 &= \exists u : u = 1, P(i, u), Q(A), EOP(), \\ K_D^2 &= \exists u : u = 1, P(i, u), Q(C), Q(B), EOP(), \\ &\dots \\ K_D^8 &= \exists u : u = 3, PT(\#), Q(B), EOP(), \\ K_D^9 &= \exists u : u = 4, R(+, u), EOP(), \\ K_D^{10} &= \exists u : u = 4, PT(\#), EOP(), \end{aligned}$$

$$K_D^{11} = \exists u : u = 4, \mathbf{F}.$$

Справедливо и обратное. Завершение последовательности доказательства формулы $F_{LL} \mathbf{G}, C_{-}$ элементами $U_{r+1}, \dots, U_{r+k}$, где $U_{r+k} = \mathbf{F}$, означает принадлежность одному из элементов последовательности $U_{r+1}, \dots, U_{r+k}$ атомов $u = n, R \mathbf{A}_n, u_{-}$. А это, в свою очередь означает наличие в грамматике G правила вида $Z \rightarrow \alpha, \alpha \in V_T$ или $Z \rightarrow \varepsilon$, применение которого успешно завершает грамматический разбор.

Случай (с).

Пусть $\varphi_i = c_1 \dots c_j Z \beta, Z \in V_N, \beta \in V^*$,

$U_B = \exists u : u = j+1, P \mathbf{A}_{j+1}, u_{-}, Q \mathbf{A}_{-}, [\alpha], EOP()$ и путем применения правила $Z \rightarrow \eta$ построена цепочка φ_{i+1} . Рассмотрим случаи, когда:

1. $\eta = c_{j+1} X \beta, X \in V_N, \beta \in V^*$. Тогда $\varphi_{i+1} = c_1 \dots c_{j+1} Z \beta, Z \in V_N, \beta \in V^*$ и цепочка $C = c_1 \dots c_{j+1} t \tau, t = c_{j+2}, \tau \in V_T^*$. По построению формулы $F_{LL} \mathbf{G}, C_{-}$ согласно пунктам 1 и 3 обработки правил грамматики G и пункту 1 обработки элементов исходной строки среди вопросов U_Q формулы $F_{LL} \mathbf{G}, C_{-}$ имеет место вопросы $\forall s : P \mathbf{A}_{j+1}, s_{-}, Q \mathbf{A}_{-} \exists R \mathbf{A}_{j+1}, s_{-}, Q(X), [\beta]$, либо $\forall s : P \mathbf{A}_{j+1}, s_{-}, RT \mathbf{A}_{j+1} \exists R \mathbf{A}_{j+1}, s_{-}$ и вопрос $\forall s : s = j+1, R \mathbf{A}_{j+1}, s_{-} \exists : s = j+2, P \mathbf{A}_{j+2}, s_{-}$. Новое состояние базы данных U_B после выполнения последнего вопроса имеет вид $\exists u : u = j+2, P \mathbf{A}_{j+1}, u_{-}, Q(X), [\beta], [\alpha], EOP()$.
2. $\eta = X \beta, X \in V_N, \beta \in V^*$. Тогда $\varphi_{i+1} = c_1 \dots c_j Z \beta, Z \in V_N, \beta \in V^*$ и цепочка $C = c_1 \dots c_j t \tau, t = c_{j+1}, \tau \in V_T^*$. По построению формулы

$F \in \mathbb{G}, C \in \mathbb{C}$ согласно пункту 2 обработки правил грамматики G среди вопросов U_Q формулы $F \in \mathbb{G}, C \in \mathbb{C}$ имеет место вопрос вида $\forall s : P \in \mathbb{P}_j, s \in \mathbb{S}, Q \in \mathbb{Q} \exists P \in \mathbb{P}_j, s \in \mathbb{S}, Q(X), [\beta]$ и он будет выполнен. Новое состояние базы данных U_B имеет вид $\forall s : P \in \mathbb{P}_j, s \in \mathbb{S}, Q(X), [\beta], [\alpha], EOP()$.

3. $\eta = Z_1 Z_2 \dots Z_h c_{j+1} \beta$, $Z_1 \in V_N$, $Z_2 \dots Z_h \in V_N^*$, $\beta \in V^*$. Тогда $\varphi_{i+1} = c_1 \dots c_j Z_1 Z_2 \dots Z_h c_{j+1} \beta$, $Z_1 \in V_N$, $Z_2 \dots Z_h \in V_N^*$, $\beta \in V^*$ и цепочка $C = c_1 \dots c_j t \tau, t = c_{j+1}, \tau \in V_T^*$. По построению формулы $F_{LL} \in \mathbb{G}, C \in \mathbb{C}$ согласно пункту 2 обработки правил грамматики G среди вопросов U_Q формулы $F_{LL} \in \mathbb{G}, C \in \mathbb{C}$ имеет место вопрос $\forall s : P \in \mathbb{P}_{j+1}, s \in \mathbb{S}, Q \in \mathbb{Q} \exists P \in \mathbb{P}_{j+1}, s \in \mathbb{S}, Q(Z_1), Q(Z_2), \dots, Q(Z_h), RT(c_{j+1}), [\beta]$. Этот вопрос будет выполнен. Новое состояние базы данных U_B имеет вид $\forall s : P \in \mathbb{P}_{j+1}, s \in \mathbb{S}, Q(Z_1), Q(Z_2), \dots, Q(Z_h), RT(c_{j+1}), [\beta], EOP()$. Справедливо и обратное.

4. $\eta = \varepsilon$. Тогда возможны подслучаи:

- 4.1. $\varphi_i = c_1 \dots c_j Z Z_1 Z_2 \dots Z_h \beta$, $Z \in V_N$, $Z_1 \in V_N, \dots, Z_h \in V_N$, $\beta \in V^*$. Тогда $\varphi_{i+1} = c_1 \dots c_j Z_1 Z_2 \dots Z_h \beta$, $Z_1 \in V_N, \dots, Z_h \in V_N$, $\beta \in V^*$ и цепочка $C = c_1 \dots c_j \tau, \tau \in V_T^*$. По построению формулы $F_{LL} \in \mathbb{G}, C \in \mathbb{C}$ согласно пункту 4 обработки правил грамматики G среди вопросов U_Q формулы $F_{LL} \in \mathbb{G}, C \in \mathbb{C}$ имеет место вопрос $\forall s : P \in \mathbb{P}_{j+1}, s \in \mathbb{S}, Q \in \mathbb{Q} \exists P \in \mathbb{P}_{j+1}, s \in \mathbb{S}$. Этот вопрос будет выполнен. Новое состояние базы данных U_B имеет вид

$\forall s : P \blacktriangleleft_{j+1}, s, \underline{\quad}, Q(Z_1), Q(Z_2), \dots, Q(Z_h), [\beta], EOP()$. Справедливо и обратное.

4.2. $\varphi_i = c_1 \dots c_j Z c_{j+1} \beta$, $Z \in V_N$, $\beta \in V^*$. Тогда $\varphi_{i+1} = c_1 \dots c_j c_{j+1} \beta$,

$\beta \in V^*$ и цепочка $C = c_1 \dots c_j t \tau, t = c_{j+1}, \tau \in V_T^*$. База

$U_B = \exists u : u = j+1, P \blacktriangleleft_{j+1}, u, \underline{\quad}, Q \blacktriangleleft_{j+1}, RT(c_{j+1})[\alpha], EOP()$. По

построению формулы $F_{LL} \blacktriangleleft, C \underline{\quad}$ согласно пунктам 3-4

обработки правил грамматики G среди вопросов U_Q формулы

$F_{LL} \blacktriangleleft, C \underline{\quad}$ имеют место вопросы

$\forall s : P \blacktriangleleft_{j+1}, s, \underline{\quad}, RT(c_{j+1}) \exists R \blacktriangleleft_{j+1}, s, \underline{\quad}$ и $\forall s : P \blacktriangleleft_{j+1}, s, \underline{\quad}, Q \blacktriangleleft_{j+1} \exists P \blacktriangleleft_{j+1}, s, \underline{\quad}$.

Эти вопросы будут выполнены. Новое состояние базы данных

U_B имеет вид $\forall s : R \blacktriangleleft_{j+1}, s, \underline{\quad}, [\beta], EOP()$. Справедливо и

обратное.

Проиллюстрируем этот случай примером. Рассмотрим грамматику

$G = \langle V_T, V_N, P, S \rangle$, где $V_T = \{+, *, i, (,)\}$, $V_N = \{A, B, C, D, E\}$, $S = A$,

$P = \left\{ \begin{array}{l} A \rightarrow CB \quad B \rightarrow +CB \quad B \rightarrow \varepsilon \quad C \rightarrow ED \quad D \rightarrow *ED \\ D \rightarrow \varepsilon \quad E \rightarrow (A) \quad E \rightarrow i \end{array} \right\}$ и входную

цепочку $C = (i) + i$. Тогда формула $F_{LL}(G, C) = U_B \cup U_Q$ имеет вид:

$$\exists u : u = 1, P((), u), Q(A), EOP() \left\{ \begin{array}{l} \forall s : P(+, s), Q(B) \exists : R(+, s), Q(C), Q(B) \\ \forall s : P(*, s), Q(D) \exists : R(*, s), Q(E), Q(D) \\ \forall s : P((), s), Q(E) \exists : R((), s), Q(A), RT() \\ \forall s : P((), s), RT() \exists : R((), s) \\ \forall s : P(i, s), Q(E) \exists : R(i, s) \\ \forall s : P(i, s), Q(A) \exists : P(i, s), Q(C), Q(B) \\ \forall s : P((), s), Q(A) \exists : P((), s), Q(C), Q(B) \\ \forall s : P(i, s), Q(C) \exists : P(i, s), Q(E), Q(D) \\ \forall s : P((), s), Q(C) \exists : P((), s), Q(E), Q(D) \\ \forall s : P(+, s), Q(D) \exists : P(+, s) \\ \forall s : P((), s), Q(D) \exists : P((), s) \\ \forall s : PT(\#), Q(D) \exists : PT(\#) \\ \forall s : P((), s), Q(B) \exists : P((), s) \\ \forall s : PT(\#), Q(B) \exists : PT(\#) \\ \forall s : s = 1, R((), s) \exists : s = 2, P(i, s) \\ \forall s : s = 2, R(i, s) \exists : s = 3, P((), s) \\ \forall s : s = 3, R((), s) \exists : s = 4, P(+, s) \\ \forall s : s = 4, R(+, s) \exists : s = 5, P(i, s) \\ \forall s : s = 5, R(i, s) \exists : PT(\#) \\ \forall PT(\#), EOP() \exists : \mathbf{F} \end{array} \right.$$

Вывод входной цепочки имеет вид

$$A \Rightarrow \dots \Rightarrow (EDB)DB \Rightarrow (iDB)DB \Rightarrow \dots \Rightarrow (i) + CB \Rightarrow i + EDB \Rightarrow \dots \Rightarrow (i) + i$$

а последовательность состояний конъюнкта вывода при ее доказательстве имеет вид

$$K_D^1 = \exists u : u = 1, P((), u), Q(A), EOP()$$

$$K_D^2 = \exists u : u = 1, P((), u), Q(C), Q(B), EOP()$$

$$K_D^3 = \exists u : u = 1, P((), u), Q(C), Q(B), EOP()$$

$$K_D^4 = \exists u : u = 1, P((), u), Q(E), Q(D), Q(B), EOP()$$

$$K_D^5 = \exists u : u = 1, R((), u), Q(A), RT(), Q(D), Q(B), EOP()$$

$$K_D^6 = \exists u : u = 1, R((), u), Q(A), RT(), Q(D), Q(B), EOP()$$

$$\begin{aligned}
K_D^7 &= \exists u : u = 2, P(i, u), Q(A), RT(), Q(D), Q(B), EOP() \\
K_D^8 &= \exists u : u = 2, P(i, u), Q(C), Q(B), RT(), Q(D), Q(B), EOP() \\
K_D^9 &= \exists u : u = 2, P(i, u), Q(E), Q(D), Q(B), RT(), Q(D), Q(B), EOP() \\
K_D^{10} &= \exists u : u = 2, R(i, u), Q(D), Q(B), RT(), Q(D), Q(B), EOP() \\
&\dots \\
K_D^{15} &= \exists u : u = 4, P(+, u), Q(D), Q(B), EOP() \\
K_D^{16} &= \exists u : u = 4, P(+, u), Q(B), EOP() \\
K_D^{17} &= \exists u : u = 4, R(+, u), Q(C), Q(B), EOP() \\
K_D^{18} &= \exists u : u = 5, P(i, u), Q(C), Q(B), EOP() \\
K_D^{19} &= \exists u : u = 5, P(i, u), Q(E), Q(D), Q(B), EOP() \\
&\dots \\
K_D^{23} &= \exists u : u = 5, PT(\#), EOP() \\
K_D^{24} &= \exists u : u = 5, \mathbf{F}
\end{aligned}$$

Справедливо и обратное. Среди элементов последовательности состояний $U_{r+1}, U_{r+2}, \dots$ базы данных формулы $F_{LL} \mathbf{G}, C^-$ по ее построению найдется состояние $U_{r+k}, k \geq 1$, имеющее вид $\forall s : P(t, s)Q(Z)\exists\dots$ или $\forall PT(t)Q(Z)\exists\dots$. Последовательность $U_{r+1}, \dots, U_{r+k}$ порождена выполнением вопросов к базе данных, построенных по правилу грамматики $Z \rightarrow \eta$ и исходной строке C . Очевидно, что элемент грамматического разбора φ_{i+1} , полученный применением правила $Z \rightarrow \eta$ к элементу грамматического разбора φ_i , удовлетворяет соответствию χ .

Таким образом, $\langle U_{r+1}, \dots, U_{r+k} \rangle = \chi \mathbf{G}_{i+1}^-$, а $\varphi_{i+1} = \chi^{-1} \mathbf{G}_{i+1}^- \langle U_{r+1}, \dots, U_{r+k} \rangle^-$.

Существование взаимнооднозначного отображения χ между последовательностью элементов грамматического разбора Φ при обработке исходной цепочки C и состояний базы данных U при доказательстве формулы $F_{LL} \mathbf{G}, C^-$ означает справедливость утверждения 2.

2.3. Выводы

В данной главе представлены алгоритмы построения распознающих позитивно-образованных формул для формальных грамматик 2 и 3 классов по Хомскому: регулярной правосторонней детерминированной грамматике и LL(1)-грамматике. Для этих грамматик доказана эквивалентность доказательства существования транслирующей программы преобразованиями распознающей позитивно-образованной формулы и грамматического вывода.

Применение логического подхода к задаче разбора и преобразования входной цепочки позволяет унифицировать различные модели трансляции. Как известно [3], они различаются наличием или отсутствием символа конца входной цепочки и различной структурой распознающих автоматов: от детерминированного конечного автомата до нескольких разновидностей автоматов с магазинной памятью, имеющих разное количество стеков, различные интерпретации элементов таблиц переходов и т.п. Правило ω конструктивного фрагмента исчисления ПОФ является основой такой унификации. Это продемонстрировано на примере двух грамматик – регулярных грамматик и LL(1)-грамматик.

Алгоритмы построения распознающих формул, а также эквивалентность грамматического вывода и доказательства распознающих ПОФ составляют одно из защищаемых положений диссертационной работы.

Этот результат опубликован в [17-20].

Глава 3. Компонент синтеза транслирующих программ

Доказательство транслирующей позитивно-образованной формулы, по сути, является доказательством существования транслирующей программы. Синтез транслирующей программы осуществляется в случае успешного доказательства ее существования.

Так как входная цепочка входит в состав транслирующей ПОФ, то транслирующая программа представляет собой последовательность команд безусловного запуска транслирующих подпрограмм.

Для решения прикладных задач в объектной среде с применением синтеза транслирующих программ на основе позитивно-образованных формул транслирующие позитивно-образованные формулы должны быть инкапсулированы в соответствующий класс. Этот класс или его экземпляры должны обеспечивать разработку и отладку транслирующей ПОФ, применение транслирующей ПОФ в составе прикладной программы с целью синтеза и исполнения транслирующих программ.

Данная глава посвящена инструментальному средству конструирования и применения в составе прикладных программ транслирующих позитивно-образованных формул. Инструментальное средство выполнено в виде компонента. Под компонентом понимается объект, доступный прикладной программе при ее исполнении, а системе визуального объектно-ориентированного программирования – при проектировании прикладной программы [73 с. 24].

Глава состоит из трех пунктов:

- В п. 3.1 разработаны условия и схема применения компонента. Компонент синтеза транслирующих программ (КСТП) предназначен для конструирования и испытания на примерах транслирующих ПОФ при проектировании прикладной программы, доказательства существования транслирующей программы для

заданной входной цепочки языка, а также синтеза и исполнения транслирующей программы в режиме исполнения прикладной программы. Компонент может быть внедрен в инструментальную среду разработки на платформе .Net и использован в прикладной программе на любом из доступных программных языков среды разработки на платформе .Net [53, 62, 90].

- Пункт 3.2 посвящен организации данных и диалоговому интерфейсу КСТП. Основу КСТП составляют объектные представления транслирующей ПОФ и формальной грамматики. Элементами объектного представления транслирующей ПОФ являются экземпляры классов, характеризующих следующие сущности: транслирующая ПОФ в целом, входная и выходная цепочки, транслирующая программа, транслирующие подпрограммы с командами запуска, атомы, переменные вывода и формальные параметры атомов. Объектное представление формальной грамматики составляют класс контекстно-свободных грамматик и его наследники – классы регулярных и LL(1)-грамматик.
- Пункт 3.3 содержит выводы по данной главе.

3.1. Условия и общая схема применения компонента

Компонент синтеза транслирующих программ (КСТП) предназначен для выполнения функций трансляции в составе пользовательских приложений. КСТП обеспечивает:

- Конструирование транслирующих ПОФ. Транслирующие позитивно-образованные формулы конструируются в диалоговом режиме с помощью специального редактора. Особенностью конструирования ПОФ является возможность автоматического построения ее части на основе заранее подготовленной регулярной или LL(1)-грамматики по алгоритмам, описанным в п.2.1 и п.2.2.

- Тестирование транслирующих позитивно-образованных формул. Транслирующие ПОФ испытываются на примерах входных цепочек в диалоговом режиме. При обнаружении ошибок вводятся необходимые корректировки в транслирующую ПОФ или транслирующие подпрограммы. Испытание транслирующей ПОФ включает ее доказательство, синтез и исполнение транслирующей программы.
- Доказательство существования транслирующей программы для заданной входной цепочки языка. Доказательство транслирующей программы инициируется методами КСТП во время исполнения пользовательского приложения. Результатом доказательства является конъюнкт синтеза.
- Синтез транслирующей программы. Синтез транслирующей программы инициируется методами КСТП. Транслирующая программа синтезируется после завершения или во время доказательства ее существования. Синтез заключается в обработке конъюнкта синтеза с целью формирования программы, составленной из команд запуска транслирующих подпрограмм, из их шаблонов, переменных вывода и текущего входного символа. Транслирующая программа представляет собой последовательность команд безусловного запуска транслирующих подпрограмм.
- Исполнение транслирующей программы. Исполнение транслирующей программы инициируется методами КСТП. Транслирующая программа может быть исполнена после завершения или по мере ее синтеза.

Общая схема применения КСТП в пользовательском приложении представлена на рис. 3.1. Согласно схеме, экземпляр компонента встраивается в пользовательское приложение, разрабатываемое в одной из инструментальных сред разработки платформы .NET, дополняя приложение своими свойствами и

функциями. Работа с компонентом происходит в двух режимах – конструирования и использования транслирующей ПОФ.

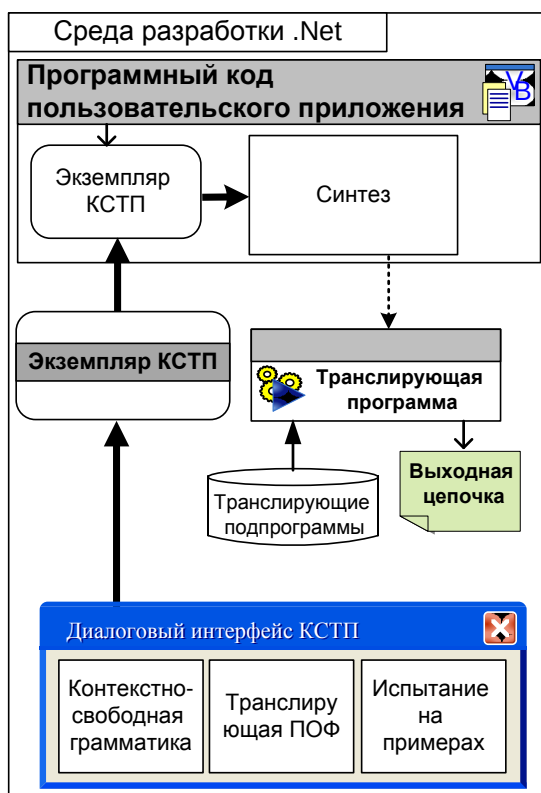


Рис. 3.1. Схема применения КСТП

В режиме конструирования с помощью диалоговых средств компонента задается часть входных данных (например, правила грамматики), строится соответствующая часть транслирующей ПОФ и проверяется ее работа на примерах тестовых цепочек

В режиме использования с помощью методов и свойств компонента задается его поведение в программном проекте прикладной задачи. При исполнении проекта, компонент по входной цепочке достраивает транслирующую ПОФ, осуществляет ее доказательство, по результатам доказательства синтезирует транслирующую программу и исполняет ее. Результат исполнения транслирующей программы КСТП записывает в выходную цепочку.

Применение компонента для выполнения синтаксически-управляемой трансляции (лексического и синтаксического анализа) в проекте учебного

транслятора основана на схеме, предложенной в [3 с. 168]. Используемая схема приведена на рис. 3.2.

Для решения задачи лексического анализа используется набор экземпляров КСТП. Каждый экземпляр в режиме конструирования настраивается на распознавание определенного типа лексем. В режиме исполнения с помощью набора КСТП выполняется лексический разбор входной строки и синтезируется код лексического анализатора входной строки в составе ПОФ.

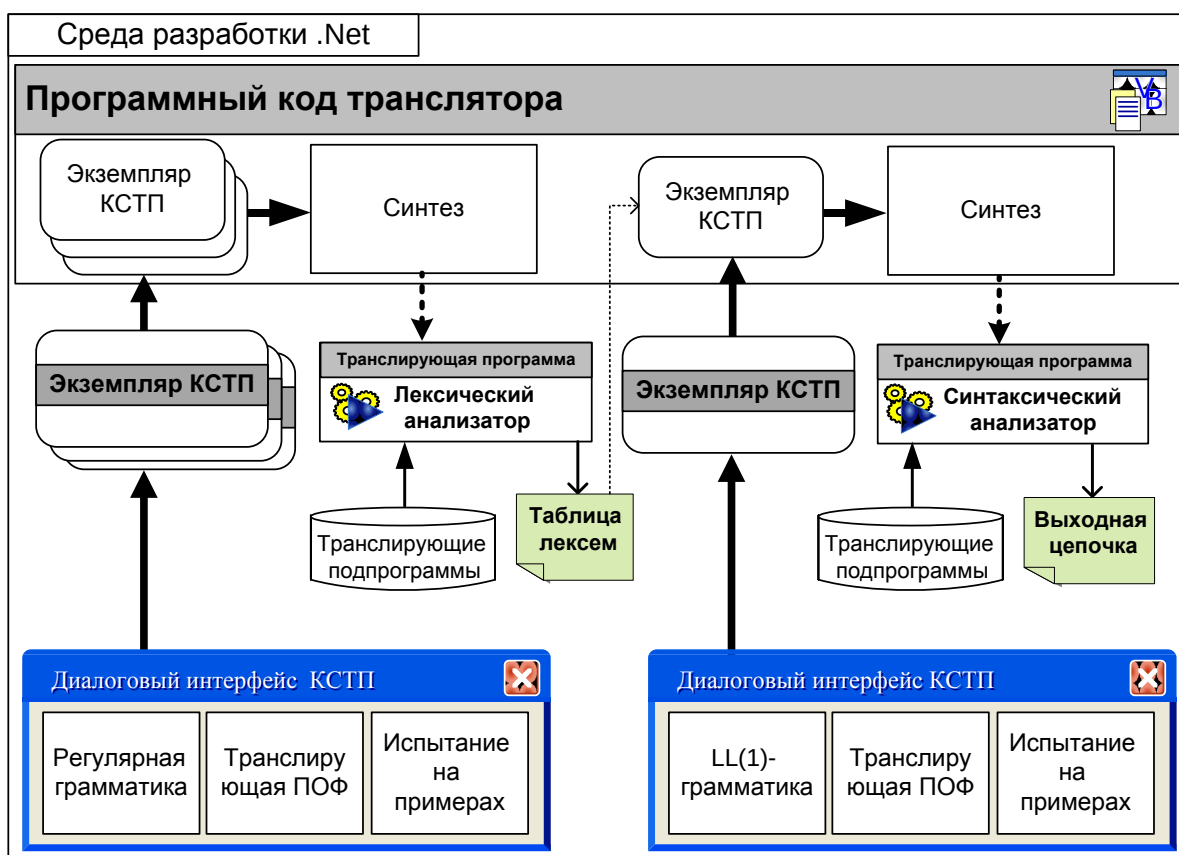


Рис. 3.2. Схема применения КСТП для выполнения синтаксически-управляемой трансляции

Для решения задачи синтаксического анализа используется один экземпляр КСТП. КСТП в режиме конструирования настраивается на выделение синтаксических конструкций во входной строке. В режиме исполнения с помощью компонента выполняется синтаксический разбор таблицы лексем, и синтезируется код синтаксического анализатора.

Результат исполнения кода лексического и синтаксического анализаторов экземпляры КСТП записывают в таблицу лексем и выходную строку соответственно.

3.2. Организация данных и диалоговый интерфейс компонента

Входными данными КСТП являются КС-грамматика, входная цепочка и шаблоны команд запуска транслирующих подпрограмм. По входным данным строится транслирующая позитивно-образованная формула.

Основу КСТП составляют объектные представления транслирующей ПОФ и формальной грамматики. Элементами объектного представления транслирующей ПОФ являются экземпляры классов, характеризующих следующие сущности:

- Транслирующая ПОФ в целом;
- Входная и выходная цепочки;
- Транслирующая программа;
- Транслирующие подпрограммы, команды их запуска и другие данные, ассоциированные с транслирующими подпрограммами.
- Атомы, переменные вывода, а также формальные параметры атомов.

Представленные выше экземпляры классов разносятся по коллекциям (см. например [1, 62]). Коллекции формируются, в частности, из атомов, составляющих конъюнкт, или из аргументов атома. Транслирующая ПОФ в целом представлена следующими коллекциями:

- Переменных вывода при корневом кванторе существования ПОФ.
- Атомов в составе конъюнктов вывода и синтеза в базе U_B .
- Вопросов в составе ПОФ.

Каждый вопрос, в свою очередь, это совокупность коллекций:

- Параметров вопроса.
- Атомов в составе предусловия вопроса.

- Атомов в составе постусловия вопроса.

Объектное представление формальной грамматики составляет класс контекстно-свободных грамматик и его наследники – классы регулярных и LL(1)-грамматик (рис. 3.3). Классы регулярных и LL(1)-грамматик отличаются друг от друга функциональными членами, предназначенными для контроля принадлежности конкретной грамматики соответствующему классу.

Практически все перечисленные выше элементы объектной модели компонента доступны для диалогового обзора и редактирования.

Диалоговый режим при применении КСТП обеспечивается штатными средствами инструментальной среды и собственными диалоговыми формами. На рис. 3.4 и 3.5 приведены примеры диалога с применением штатных средств инструментальной среды.

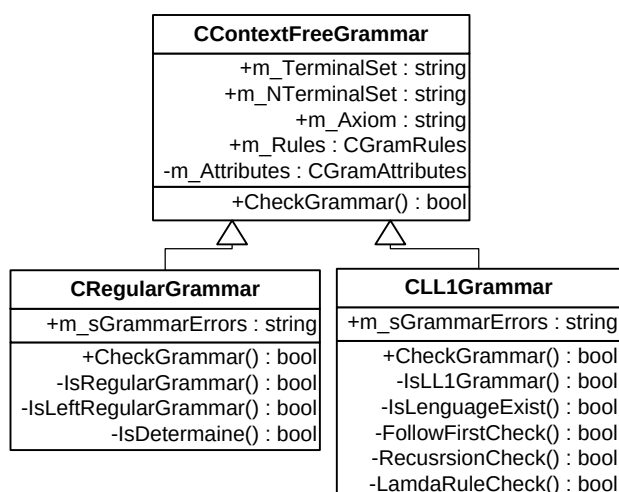


Рис. 3.3. UML-диаграмма классов грамматик в КСТП

Собственные диалоговые средства КСТП составляют четыре диалоговых формы:

1. Форма редактирования грамматик (рис.3.6). На форме вводятся множество терминальных символов, множество нетерминальных символов, аксиома и правила грамматики. Результат редактирования проверяется на принадлежность к классам

регулярной правосторонней детерминированной грамматике или LL(1)-грамматике.

2. Форма редактирования команд запуска транслирующих подпрограмм (рис.3.7). На форме вводятся шаблоны команды запуска транслирующей подпрограммы и параметры запуска. Также на форме вводится ссылка на подготовленный к исполнению код класса транслирующих подпрограмм. Класс наследуется от интерфейса КСТП *ISubRutines*, методы данного класса и являются реализацией транслирующих подпрограммами.
3. Форма непосредственного редактирования объектного представления транслирующей ПОФ (рис. 3.8). На форме редактируются коллекции, составляющие транслирующую ПОФ – конъюнкт вывода, конъюнкт синтеза и вопросы ПОФ. Кроме этого на этой форме вводится входная строка для испытания транслирующей ПОФ при проектировании прикладного приложения.
4. Форма сохранения результатов настроек транслирующей ПОФ в файл. Настройки сохраняются в виде XML-документа для последующей их загрузки в другой экземпляр КСТП.

Управление экземпляром КСТП при исполнении прикладной программы обеспечивается обращением к его функциональным членам. Краткое описание этих функциональных членов приведено в табл.3.1.

3.3. Выводы

Реализация транслирующей ПОФ и процесса ее построения в виде экземпляра КСТП позволяет унифицировать решение задач трансляции на основе универсального логического описания формальных языков. Это позволяет с помощью нескольких экземпляров КСТП решать задачи трансляции для двухуровневых (лексика и синтаксис) формальных языков.

С помощью КСТП задачи трансляции решаются на более высоком технологическом уровне по сравнению с компиляторами компиляторов.

Ниже представлен сравнительный анализ КСТП с другими средствами разработки трансляторов.

	КСТП	Prolog	YACC
Решение задач лексического анализа	+	+	-
Решение задачи синтаксического анализа	+	+	+
Работа с LL-грамматиками	+	+	-
Работа с LR-грамматиками	+ ¹¹	+	+
Интеграция в прикладную программу	В виде программного компонента	Вызов внешнего интерпретатора Prolog	В виде сгенерированного кода
Размер исходного кода транслятора для задачи распознавания арифметических выражений ¹²	~24 вопроса	~35 предложений	~40 строк
Отладка транслятора	+	-	-
Автоматическое исправление ошибок при трансляции	+	+	-
Графический интерфейс	+	-	-

Вариант КСТП разработан в среде MS Visual Studio .Net на языке C# [29]. КСТП можно применять в любой инструментальной среде визуального объектно-ориентированного программирования для платформы .Net. КСТП испытан в таких диалоговых средах как Visual Studio .Net 2005, Sharp Develop 2.2 , Code Gear RAD Studio 2007.

Разработка компонента синтеза транслирующих программ составляет одно из защищаемых положений диссертационной работы. Данный результат опубликован в [17, 39-41, 56].

¹¹Севостьянова Д.В., Курганский В.И. Восходящий грамматический разбор на основе позитивно-образованных формул // Вестник Иркутского государственного университета. Материалы научной студенческой конференции, 2009 г. – С. 163-165

¹²Jacques Cohen. Parsing and compiling using Prolog / Jacques Cohen, Timothy J. Hickey. – ACM Transactions on Programming Languages and Systems (TOPLAS). Volume 9 Issue 2, April 1987: –pp.125-163

Основные функциональные члены КСТП Таблица 3.1.

№	Название	Тип возвращаемого значения	Описание
1	Свойство Designer	Форма	Вызов диалога редактирования транслирующей ПОФ.
2	Свойство InputString	Коллекция	Предоставляет доступ для работы с входной строкой.
3	Метод Prove	Boolean	Осуществляет доказательство ПОФ. В случае успешного доказательства возвращает True.
4	Свойство SynExecAfterProve	Boolean	Задаёт способ синтеза и исполнения транслирующей программы – пошаговый или после завершения доказательства ПОФ.
5	Метод Synthesize	Boolean	Осуществляет синтез программы. В случае успеха возвращает True, иначе False.
6	Метод Execute	Boolean	Осуществляет исполнение синтезированной программы. В случае успеха возвращает True.
7	Свойство PCFQuery	Коллекция	Предоставляет доступ к вопросам ПОФ.
8	Свойство PCFKsyn	Коллекция	Предоставляет доступ к конъюнкту синтеза.
9	Свойство PCFKprove	Коллекция	Предоставляет доступ к конъюнкту вывода.
10	Метод GetOutputString	Коллекция	Возвращает выходную цепочку в виде коллекции. Цепочка строится при исполнении синтезированной транслирующей программы.
18	Метод BuildPCFbyGrm	Boolean	Осуществляет построение ПОФ по грамматике. В случае успеха возвращает True, иначе False.
19	Свойство RoutinesTpl	Коллекция	Предоставляет интерфейс для задания транслирующих подпрограмм.
20	Свойство Atoms	Коллекция	Предоставляет интерфейс для работы с атомами. Основными методами являются добавить, удалить и изменить.
21	Свойство SpecialTrSymbols	Коллекция	Предоставляет интерфейс для работы с символами конца строки. Основными методами являются добавить, удалить и изменить.
22	Свойство SettingsName	Строка	Предоставляет доступ для сохранения (загрузки) настроек КСТП в виде (из) XML-документа.
23	Свойство Description	Строка	Предоставляет доступ для редактирования описания к КСТП.
24	Свойство Descriptor	Строка	Предоставляет доступ для редактирования дескриптора КСТП.
25	Событие ProveStart		Событие порождается до начала доказательства транслирующей ПОФ
26	Событие ProveStep		Событие порождается при выполнении очередного шага доказательства ПОФ
27	Событие ProveEnd		Событие порождается после окончания доказательства транслирующей ПОФ.

Основные функциональные члены КСТП

Продолжение таблицы 3.1.

№	Название	Тип возвращае мого значения	Описание
28	Событие ProveAbduk		Событие порождается при выполнении очередного шага доказательства транслирующей ПОФ если не возможно выполнить не один вопрос.
29	Метод GetResult	Коллекция	Возвращает результат исполнения транслирующей программы.
30.	Свойство Colletion_SR	Коллекция	Элементы коллекции содержат пары <указатель, функция>, где указатель – значение типа <i>delegate</i> для запуска функции, а функция – метод, реализующий транслирующую подпрограмму.

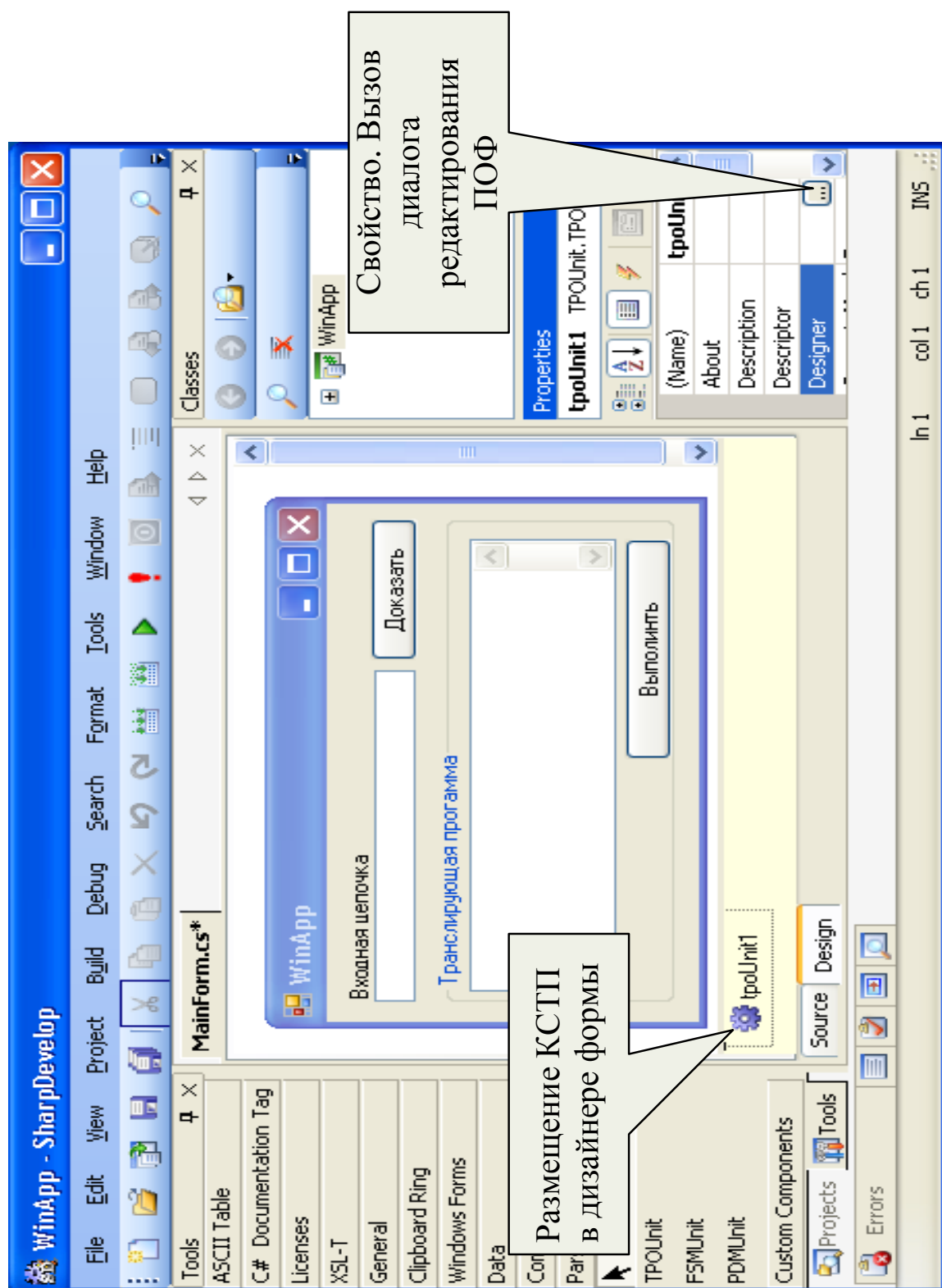


Рис.3.4. Пример управления проектом приложения с применением экземпляров КСТП

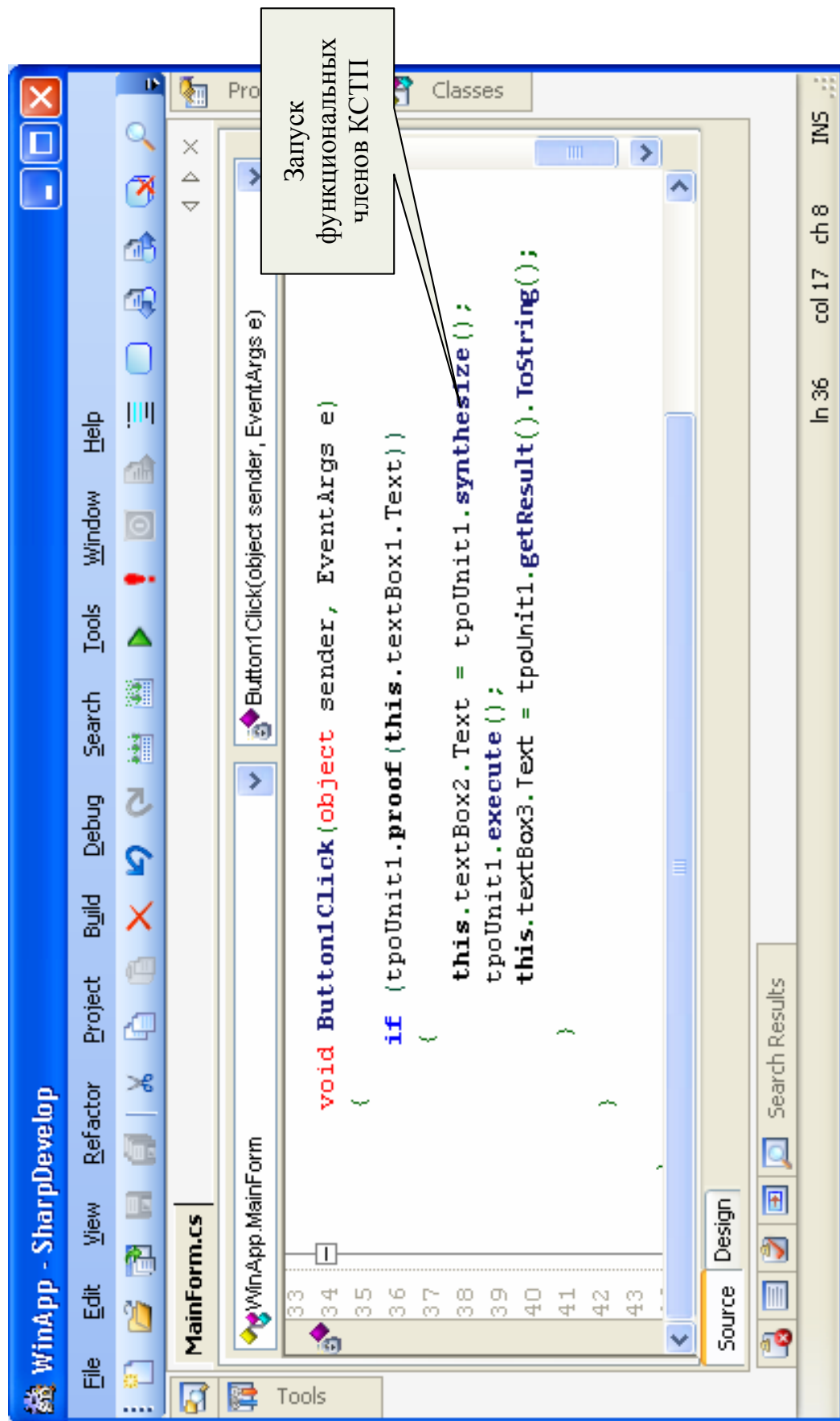
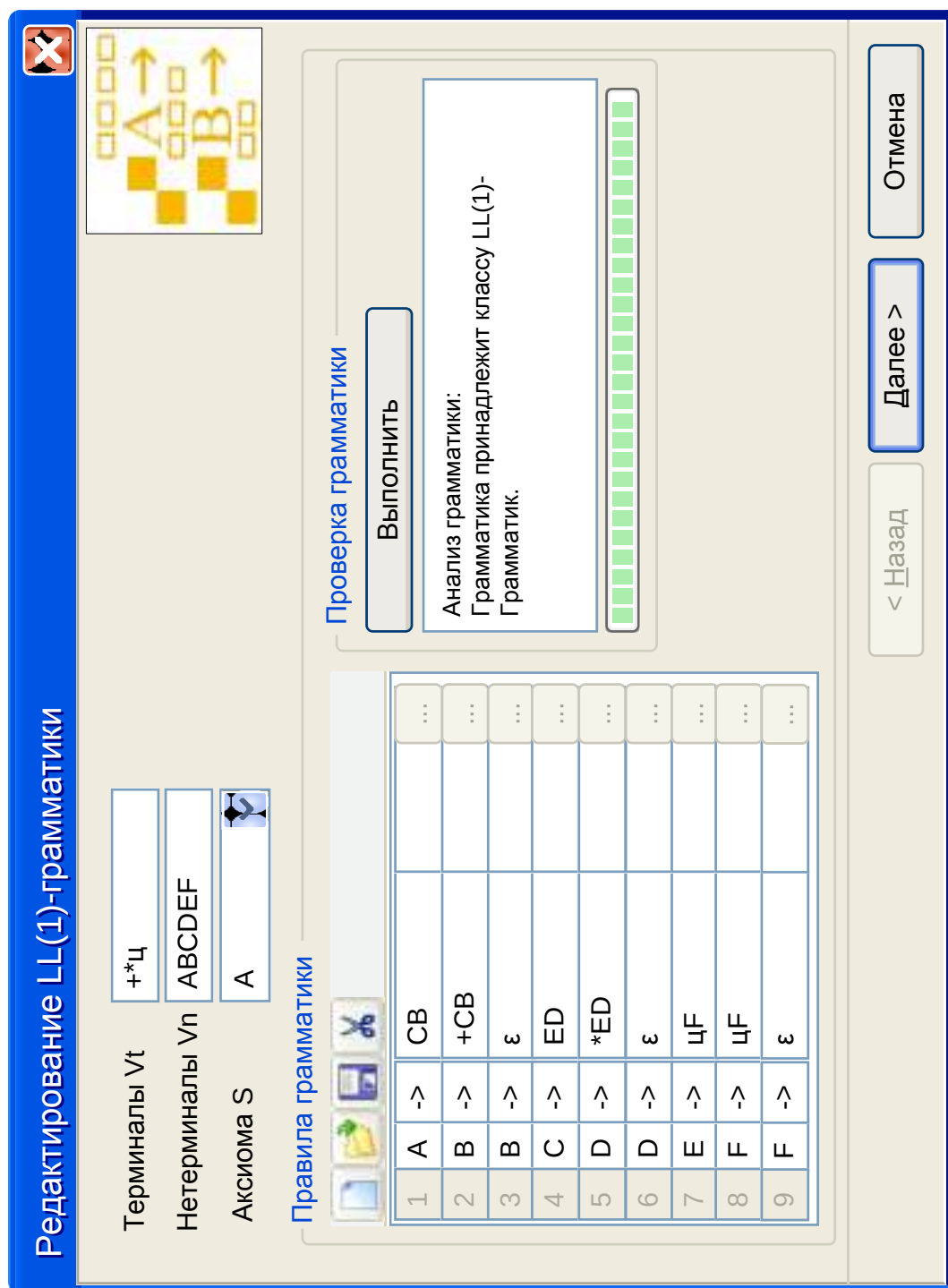


Рис.3.5. Редактирование текста учебного транслятора штатными средствами инструментальной среды



Терминалы Vt

$$\begin{array}{c} \text{U} \\ * \\ + \end{array}$$

Нетерминалы V_n

ABCDEF

Аксиома S

A

Правила грамматики

1	A	$\rightarrow$	CB	...
2	B	$\rightarrow$	+CB	...
3	B	$\rightarrow$	ε	...
4	C	$\rightarrow$	ED	...
5	D	$\rightarrow$	*ED	...
6	D	$\rightarrow$	ε	...
7	E	$\rightarrow$	$\neg F$	...
8	F	$\rightarrow$	$\neg F$	...
9	F	$\rightarrow$	ε	...

Проверка грамматики

Выполнить

Анализ грамматики:
Грамматика принадлежит классу LL(1)-
Грамматик.

10

◀ Назад

Далее >

Отмена

Рис.3.6. Форма для редактирования КС-грамматики

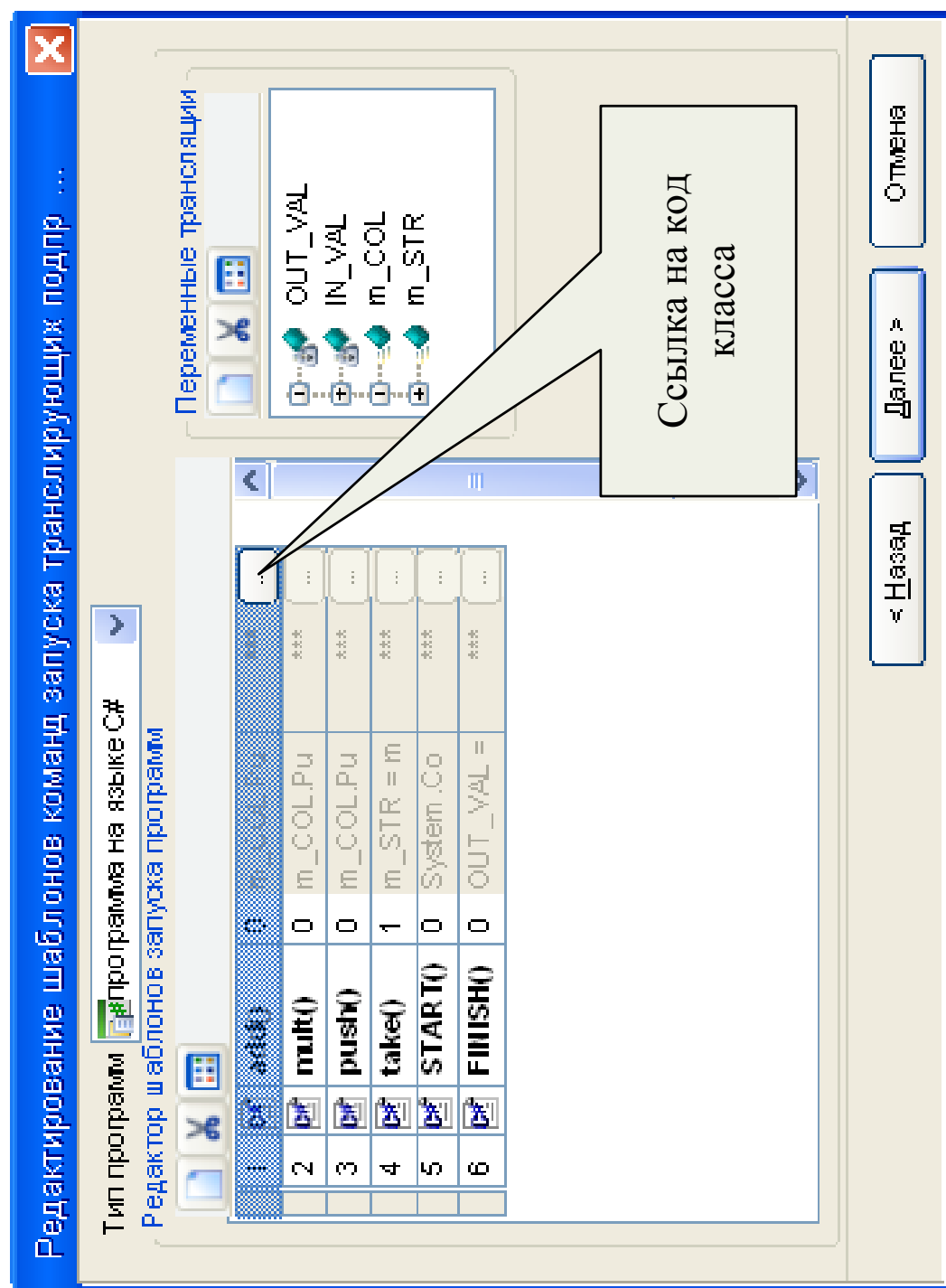


Рис.3.7. Форма для редактирования шаблонов команд запуска транлирующих подпрограмм

Глава 4. Примеры применения компонента синтеза транслирующих программ

Большинство современных методов и нотаций программирования концентрируются на поиске и композиции функциональных компонентов, которые обычно выражаются в виде объектов, модулей, процедур [73, с. 25]. Однако некоторые возможности, такие как обработка ошибок, трансляция, синхронизация, устойчивость и безопасность не всегда могут быть естественно и ясно выражены при использовании современных (например, объектно-ориентированных) нотаций и языков. Вместо этого, они выражаются небольшими фрагментами кода, разнесенными по нескольким функциональным компонентам.

Компонент синтеза транслирующих программ позволяет организовать использование имеющихся фрагментов транслирующего кода на основе целостной логической модели при решении задач трансляции формальных языков. Логическая модель обеспечивает ясное и целостное описание процесса трансляции и указание на то, какие подпрограммы следует использовать при трансляции в том или ином случае.

Компонент предназначен для построения транслирующих систем на основе пары математических моделей – формальных грамматик и позитивно-образованных формул. Применение компонента предполагает обработку входной цепочки путем доказательства существования транслирующей программы, синтеза и исполнения транслирующей программы.

Данная глава посвящена применениям КСТП при решении двух практических задач.

В п. 4.1 рассмотрен результат решения одной из задач – это технология конструирования учебных трансляторов. Учебный транслятор представляет собой GUI-приложение для платформы .Net. Собственно транслятор представляет собой несколько экземпляров КСТП (по одному для каждого типа лексем и один для синтаксиса входного языка) и программный код объемом

около 100 строк исходного текста для управления процессом трансляции. Все шаги этого процесса (разбор входных цепочек, синтез и исполнение транслирующих программ) обеспечиваются обращением к функциональным членам соответствующих экземпляров КСТП.

В п. 4.2 представлено решение второй задачи – контроля допустимости цепочек входных воздействий в составе диалогового средства таможенного контроля Терминал. Для контроля цепочек входных воздействий разработана логико-лингвистическая модель, которая представляет собой комплект транслирующих позитивно-образованных формул. Транслирующие ПОФ разработаны для ряда элементов управления и каждой диалоговой формы и реализованы в виде экземпляров КСТП. Особенность применения КСТП в данном случае заключается в том, что элементами алфавитов формальных языков, которым принадлежат правильные цепочки входных воздействий, являются кортежами. Кортеж это четверка, состоящая из идентификатора элемента диалога, идентификатора события, идентификатора параметра и значения параметра. Доказательство существования, синтез и исполнение транслирующих программ осуществляются в пошаговом режиме по мере формирования входных цепочек.

4.1. Технология решения учебных задач трансляции

Для разработки трансляторов и прикладных систем лингвистического обеспечения можно использовать обычные инструментальные средства, применяемые для создания программного обеспечения, такие как интегрированные среды разработки и программирования [например, 52]. Но создание трансляторов с использованием только этих средств очень трудоемкий процесс. Поэтому, для того чтобы сократить время, затрачиваемое на разработку и повысить качество программы, в дополнение к этим средствам могут использоваться специальные инструментальные средства разработки трансляторов.

Идея использования дополнительных инструментальных средств для разработки трансляторов, по-видимому, впервые предложена Тони Брукером в 1960 году при разработке языка Atlas Autocode [4]. Идея, предложенная Брукером, заключалась в генерации таблицы переходов автомата по формальной грамматике.

Современные средства разработки трансляторов значительно расширили идею Брукера. Выделяют следующие типы современных средств разработки трансляторов:

1. **Генераторы лексических анализаторов.** Данные генераторы строят лексические анализаторы, на основе правил регулярной грамматики. Сгенерированный лексический анализатор представляет конечный автомат [66, 69]. Как правило, такие системы генерируют программный код, который может быть интегрирован в код, построенный генераторами синтаксических анализаторов. Примерами таких систем является lex, flex, JLex, YooLex [85].
2. **Генераторы синтаксических анализаторов.** Данные системы генерируют код синтаксических анализаторов, на основе правил формальной грамматики записанных, например, в виде БНФ [30]. В зависимости от типа формальной грамматики выделяют следующие разновидности синтаксических анализаторов:
 - Генераторы анализаторов восходящего разбора. Эти генераторы основаны на LR-грамматиках. Основой LR-анализатора является таблица переходов, для построения которой, как правило, используется метод LALR [8]. Примерами таких генераторов являются YACC, Bison, Gold, GDK, PLY [85].
 - Генераторы анализаторов нисходящего разбора. Основу генератора составляет предикативный анализ. Основой предикативного анализа также является однозначная таблица переходов [67]. Таблица переходов строится на основе LL-грамматики по

известному алгоритму [3]. Примерами таких генераторов являются LISA, Coco/R, ppCC, JavaCC, SLK, Tap [85, 102].

3. **Средства синтаксически-управляемой трансляции.** Такие системы позволяют создавать наборы программ, которые осуществляют обход дерева синтаксического разбора и генерируют промежуточный код. Код генерируется из атрибутов заложенных в грамматику [3 с. 280]. В качестве примера таких систем можно привести Kimwitu, The Memphis Tree Builder & TreeWalker Tool, ParseUnit, DESCARTES [41, 84, 85].
4. **Автоматические генераторы кода.** Данные системы получают набор правил, определяющие способ трансляции каждой операции промежуточного языка в машинный язык. Основу этих генераторов составляют эвристические алгоритмы и требования к объектному коду. Примерами таких систем являются Twig, BEG, IBURG [3].
5. **Генераторы анализаторов потока данных и оптимизаторов.** В идеале компилятор должен давать столь же хороший код, как и код написанный вручную. Реальность же такова, что это достигается только в ограниченном количестве случаев, к тому же с огромным трудом. Однако зачастую код, который дает алгоритм компиляции, может быть сделан более быстрым или более компактным. Такое улучшение достигается преобразованием программы с помощью специальных систем. Эти системы позволяют создавать программы, осуществляющие анализ потока данных, необходимых для получения оптимизированного кода. Примерами таких систем являются Berkeley ANalysis Engine, Oprimix, Pag [3, 85].
6. **Пакеты разработки компиляторов.** Такие пакеты, как правило, включают в себя почти все перечисленные выше инструментальные средства разработки компиляторов. Особенность данных пакетов является интеграция и использования генераторов в виде единого модуля.

Примерами таких систем являются метасистема-SATURN, COCKTAIL, ELI, GENTLE, PCCTS [49, 85].

7. **Системы создания сред разработки.** Данные системы кроме компилятора позволяют построить редактор, ориентированный на определенный язык, отладчик, редактор структур и диалоговых интерфейсов. В качестве примера таких систем можно привести The Synthesizer Generator, Centaur, Synthesizer Generator, Eclipse [85].

Математические модели трансляции являются наиболее характерным материалом в части изучения и практического применения современного математического аппарата при решении нетривиальных задач. Практическое освоение математических моделей трансляции требует построения и исследования математических моделей формальных языков, преобразования этих моделей в модели процессов трансляции и, наконец, разработки транслирующей программы. Ситуация усугубляется тем, что строить и исследовать модели учебных трансляторов, используя только карандаш и бумагу, довольно трудно, так как даже для простейших задач, имеющих хоть какой то практический смысл, объем рутинных вычислений довольно велик. Построить транслирующую программу человеку, имеющему опыт написания программ объемом несколько десятков команд, даже на основе хорошо проработанной математической модели представляется трудновыполнимой задачей. Вывод один — необходимы средства автоматизации решения учебных задач, не только соответствующие современному уровню в научно-техническом плане, но и обеспечивающие достижение учебных целей. Популярные средства автоматизации разработки транслирующих программ (например, [47 с.74, 76, 81, 85, 95, 98, 99, 100]) ориентированы на решение практических задач, но не обеспечивают достижения учебных целей в должном объеме. Большое количество Web-ресурсов, посвященных вопросам разработки трансляторов, и их содержание говорят о популярности вопросов математического

моделирования языков и процессов трансляции в профессиональной, студенческой и аспирантской среде.

Компонент синтеза транслирующих программ пригоден для учебных целей и позволяет строить учебные трансляторы. В нашем случае учебный транслятор – это GUI-приложение для платформы .Net. Пример диалоговой формы учебного транслятора приведен на рис. 4.1. Транслятор представляет собой несколько экземпляров КСТП (по одному для каждого типа лексем и один для синтаксиса входного языка), набор штатных визуальных объектов (форма и элементы управления) и программный код объемом около 100 строк исходного текста для управления процессом трансляции.

Все шаги процесса трансляции (разбор входных цепочек, синтез и исполнение транслирующих программ) обеспечиваются обращением к функциональным членам соответствующих экземпляров КСТП.

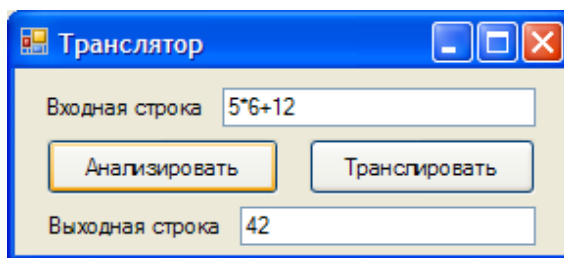


Рис. 4.1. Диалоговая форма учебного транслятора

Конструирование транслирующих позитивно-образованных формул также обеспечивается соответствующими экземплярами КСТП при проектировании учебного транслятора в инструментальной среде Microsoft Visual Studio.Net.

Все остальные действия, связанные с разработкой учебного транслятора, обеспечиваются штатными средствами среды разработки Visual Studio .Net [52].

Описание технологии решения учебных задач с использованием компонента представлено на рис. 3.2. Технология внедрена в Институте математики, экономики и информатики Иркутского государственного университета [39, 41]. Набор компонентов, обеспечивающих данную

технологию, включает КСТП и зарегистрирован Иркутским государственным университетом в Федеральной службе по интеллектуальной собственности, патентам и товарным знакам [56].

Далее в п. 4.1.1 рассмотрим требования к учебным трансляторам и их структуру, а в п. 4.1.2 – пример учебного транслятора.

4.1.1. Учебные трансляторы. Требования и структура

Рассмотрим технологию решения учебных задач трансляции более подробно. Учебная задача трансляции состоит из двух подзадач – задачи лексического анализа и задачи синтаксического анализа.

Решение задачи лексического анализа предполагает настройку нескольких экземпляров КСТП на выполнение разбора входной строки по типам лексем. Настройка состоит из следующих шагов:

1. Разработка регулярной грамматики для распознавания заданного типа лексем (например, целочисленная константа, идентификатор и др.).
2. Построение набора транслирующих подпрограмм (например, построение коллекции лексем или исправление ошибок во входной строке).
3. Построение транслирующей ПОФ из заданной грамматики по приведенному в гл. 2 алгоритму.
4. Тестирование транслирующей ПОФ на примерах правильных и неправильных цепочек символов входной строки
5. Программная реализация лексического разбора входной строки с помощью методов, свойств и событий КСТП;

Таким образом, лексический разбор входной строки выполняется несколькими экземплярами КСТП. Результатом выполнения лексического разбора является коллекция (таблица) лексем с двумя полями – исходное представление и дескриптор лексемы. Например, в результате выполнения

лексического разбора арифметического выражения вида $12.5 + iVar * 4$ получится коллекция следующего вида:

№	Исходное представление	Дескриптор
1	12.5	I
2	+	+
3	iVar	V
4	*	*
5	4	I

Необходимость такой коллекции обусловлена тем, что синтаксический анализатор работает с дескрипторами лексем, а транслирующая программа – с их исходным представлением.

Решение задачи синтаксического анализа заключается в настройке одного экземпляра КСТП на синтаксический разбор последовательности лексем. Настройка состоит из следующих шагов:

1. Разработка LL(1)-грамматики для распознавания синтаксических конструкций во входной строке (например, арифметических выражений и др.).
2. Построение набора транслирующих подпрограмм (пример комплект транслирующих подпрограмм приведен в таблице 1.1 гл. 1 настоящей работы)
3. Построение транслирующей ПОФ по приведенному в гл. 2 алгоритму.
4. Тестирование транслирующей ПОФ на примерах правильных и неправильных цепочек символов входной строки.
5. Программная реализация синтаксического разбора последовательности лексем с применением методов, свойств и событий КСТП;

Результатом синтаксического анализа последовательности лексем является выходная строка. Причем вид выходной строки зависит от транслирующих подпрограмм. Так, для арифметических выражений это может быть вычисленное значение, постфиксная или префиксная запись выражения.

4.1.2. Пример учебного транслятора

Рассмотрим пример транслятора, вычисляющего десятично-дробные арифметические выражения. Транслятор разрабатывается в Visual Studio .Net. Для управления транслятором используется диалоговая форма (рис.4.1), ее визуальное объектное представление на этапе проектирования приведено на рис. 4.2.

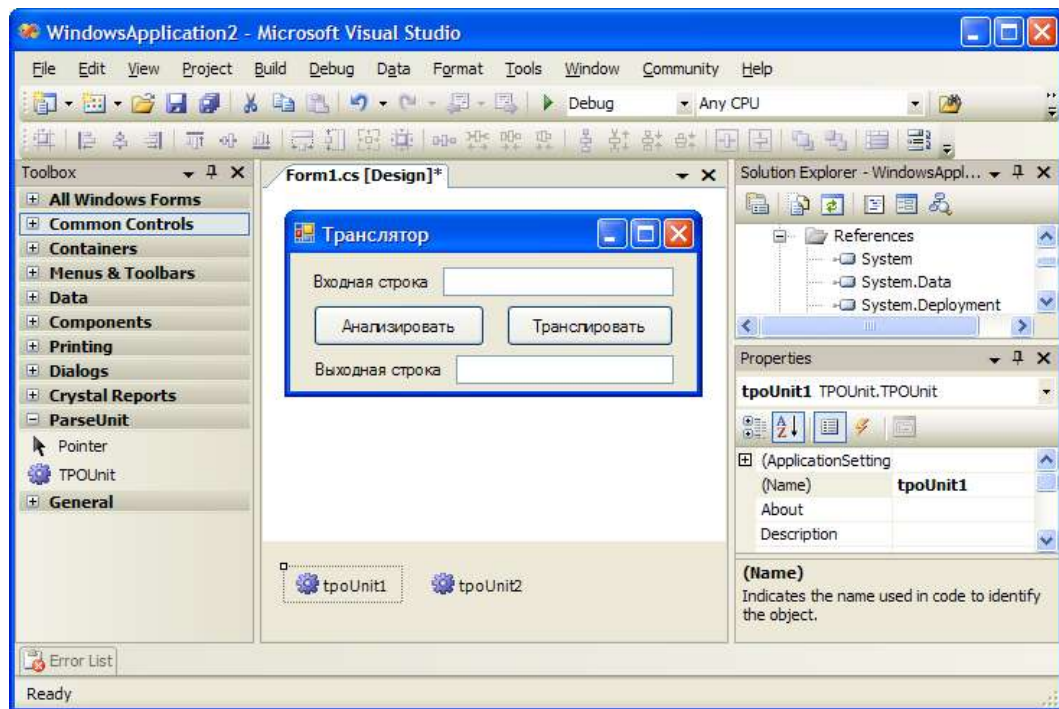


Рис. 4.2. Объектное представление диалоговой формы учебного транслятора

Диалоговое окно транслятора и результат его работы представлен на рис. 4.1. Учебный транслятор обеспечивает лексический анализ входной цепочки литер из текстового элемента управления, помеченного элементом управления типа *Label* (Входная строка), размещение результата лексического анализа в коллекции лексем *mColl*, синтаксический анализ этой коллекции лексем и размещение результата трансляции в текстовом элементе управления, помеченного элементом управления типа *Label* (Выходная строка).

В программном проекте использовано два экземпляра КСТП – *tpoUnit1* и *tpoUnit2*. Первый компонент обеспечивает лексический анализ числовых констант с плавающей точкой на основе ПОФ приведенной в пункте 2.1

(формула 2.1.1) настоящей работы. Лексемы +, *, # распознаются с помощью метода *Substring* для строковых величин [62, с. 124].

Экземпляр КСТП *tpoUnit2* обеспечивает синтаксический анализ коллекции *mColl*. LL(1)-грамматика, составляющая основу этого экземпляра КСТП, и соответствующая транслирующая ПОФ приведены в пункте 2.2 (формула 2.2.1) настоящей работы. Класс транслирующих подпрограмм составляют шесть функций, все они реализуют операции работы со стеком. Класс транслирующих подпрограмм имеет вид:

```
class SyntaxSubRoutines : CTPOUnit.ISubRoutines
{
    private Stack iSteck;
    private int iInternalU;
    public int START()
    {
        iSteck = new Stack();
        iInternalU = 0;
        return iInternalU;
    }
    public void FINISH(ref ArrayList OUT_VAL)
    {
        iSteck = null;
        OUT_VAL.Add(m_COL.Pop().ToString());
    }
    public int Plus1()
    {
        iInternalU++;
        return iInternalU;
    }
    public void push(ref ArrayList IN_VAL)
    {
        iSteck.Push(IN_VAL[iInternalU]);
    }
    public void add()
    {
        iSteck.Push((int)iSteck.Pop() + (int)iSteck.Pop());
    }
    public void mult()
    {
        iSteck.Push((int)iSteck.Pop() * (int)iSteck.Pop());
    }
}
```

Программный код лексического и синтаксического анализаторов включен в модуль формы и приведен ниже.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
```

```

using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Collections;

namespace WindowsApplication2
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

            TPOUnit.Lexemes mColl = new TPOUnit.Lexemes();
            // обработка нажатия кнопки «Анализировать»
            private void button1_Click(object sender, EventArgs e)
                // Лексический анализ
            {
                string iSemi = "+*#", iLex="";
                for (int i = 0; i < textBox1.Text.Length; i++)
                {
                    if (iSemi.Contains(textBox1.Text.Substring(i, 1)))
                    {
                        tpoUnit1.InputString = iLex;
                        if (tpoUnit1.Prove())
                            mColl.addLexeme(iLex, tpoUnit1.Descriptor);
                        mColl.addLexeme(textBox1.Text.Substring(i, 1), textBox1.Text.Substring(i, 1));
                    }
                    else
                        MessageBox.Show("Ошибка при разборе: " + iLex);
                    iLex = "";
                }
                iLex += textBox1.Text.Substring(i, 1);
            }
        }
        // обработка нажатия кнопки «Транслировать»
        private void button2_Click(object sender, EventArgs e)
            // Синтаксический анализ
        {
            SyntaxSubRoutines iSRoutines;
            tpoUnit2. RoutinesTmp1 = iSRoutines; //ссылка на класс транслирующих подпрограмм
            tpoUnit2.InputString = mColl;
            if (tpoUnit2.Prove() && tpoUnit2.Synthesize() && tpoUnit2.Execute())
                textBox2.Text = tpoUnit2.GetResult().ToString();
            else
                MessageBox.Show("Входная строка отвергнута!", "Синтаксический анализ")
        }
    }
}

```

4.2. Контроль цепочек входных воздействий в диалоговой системе таможенного контроля Терминал

Диалог в современных прикладных диалоговых системах зачастую представляет собой сложные цепочки взаимосвязанных входных воздействий, которые порождают различные виды событий. Штатные средства программирования обработки событий в современных системах объектно-ориентированного программирования не обеспечивают анализа контекста входного воздействия (см. например, [6]).

Первые попытки формально специфицировать входные воздействия в диалоговых системах появились в 80-х годах [86]. В работе [78] рассмотрен обзор различных моделей спецификации входных воздействий. Наибольшую популярность получила модель, основанная на формальных грамматиках и предложенная Рейснером [97]. Основная идея данной модели основана на трех тезисах [77]:

1. Входные воздействия в диалоговой системе представляются сообщениями, которые инициализируют процедуры обработки событий (например, сообщение нажатия кнопки *BM_CLICK*);
2. Представление последовательности входных воздействий в виде строки (например *{BM_CLICK}{WM_SETFOCUS}{WM_SETTEXT}*);
3. Описание правильных (неправильных) последовательностей входных воздействий с помощью формальных грамматик.

В данном пункте предложена разновидность модели контроля входных воздействий с помощью регулярных и LL(1)- грамматик. Отличие заключается в использовании транслирующих ПОФ для моделирования разбора последовательности входных воздействий. Применение ПОФ позволяет решать более широкий спектр задач за счет возможности синтеза различных “транслирующих” программ. Использование методик искусственного интеллекта при контроле входных воздействий впервые предложена Янгом [106].

Рассмотрим решение задачи контроля допустимости цепочек входных воздействий в составе диалогового средства таможенного контроля (СТК) Терминал. Это средство представляет собой мобильное вычислительное устройство, оснащенное программой сбора учетных данных и результатов контрольных измерений лесоматериалов необработанных.

В соответствии с решением коллегии ФТС России средство таможенного контроля Терминал принято к экспериментальному опробованию в составе новой технологии контроля лесоматериалов необработанных¹³.

Далее в пункте 4.2.1 приведено назначение и структура диалоговой СТК Терминал. В пункте 4.2.2 рассмотрена логико-лингвистическая модель СТК Терминал. Пункт 4.2.3 посвящен реализации модели.

4.2.1. Назначение и структура диалоговой системы

СТК Терминал предназначено для сбора учетных и контрольных данных о деятельности лесного склада. СТК представляет собой специальное вычислительное устройство – портативный терминал сбора данных (например, Minic4x или TDS Recon). Устройство базируется на специальной версии операционной системы Windows. Особенностью данных устройств является отсутствие приемлемого манипулятора курсором, поэтому управление вводом осуществляется с помощью кнопок на клавиатуре корпуса устройства. Основная программная часть СТК Терминала разрабатывалась на платформе .Net Compact Framework в среде разработки MS Visual Studio .Net 2003 на языке C# [55], дополнительные программные библиотеки были разработаны отдельно в MS Visual eMbedded Tools 3.0 на языке C++ [2].

Диалог СТК Терминал состоит из 5 основных и 18 вспомогательных диалоговых форм. Рассмотрим основные формы подробно.

¹³ О распространении системы электронного поштучного учета. Письмо ФТС России от 11 марта 2008 №01-11/9055

1. Форма «Бревно». Данная форма предназначена для ввода характеристик бревен. Форма используется в четырех различных контекстах – для ввода данных о приемке, для ввода контрольных характеристик при контроле на складе, для ввода контрольных характеристик при контроле транспортного средства и для отображения данных об отгруженных в транспортное средство бревенах. Форму составляют 4 кнопки, 4 поля для ввода и 4 выпадающих списка (рис. 4.4).
2. Форма «Штабель». Данная форма предназначена для ввода характеристик штабеля. Форма используется в трех контекстах – для ввода данных о приемке, для ввода данных об отгрузке и для контроля транспортного средства. Форму составляют 4 кнопки, 11 полей для ввода и 1 выпадающий список (рис. 4.5).
3. Форма «Акт». Данная форма предназначена для ввода данных об актах приемки и отгрузки. Форма используется в двух контекстах – для создания акта приемки и для создания акта отгрузки. Форму составляют 5 кнопок, 2 поля для ввода и 1 список (рис. 4.6).
4. Форма «Акт контроля на складе». Данная форма предназначена для ввода данных об акте контроля бревен на лесном складе. Форму составляют 2 кнопки и 2 поля для ввода (рис. 4.7).
5. Форма «Акт контроля транспортного средства». Данная форма предназначена для ввода данных об акте контроля бревен и штабелей на транспортном средстве. Форму составляют 4 кнопки и 3 поля для ввода (рис. 4.7).

№ бирки		Порода		ГОСТ	
A00001		Сосна		9463-88	
Уч.д.	д	Сорт	В, м3		
24	Д	2	0.21		
Длина		ТН ВЭД			
4.00		4403203100			

Рис. 4.4. СТК Терминал. Диалоговая форма «Бревно»

№ш	Длина	Ширина	Высота
1	4.00	2.03	2.06

ТП1пол	ТП2пол	ТПвнут	ТПшапк	Т.мусор
0.01	0.03	0.00	0.02	0.22

Толщ.от	Толщ.до	П. К.	В.расч.	V, скл.
20	40	0.56	1.76	14.29

Рис. 4.5. СТК Терминал. Диалоговая форма «Штабель»

Бревна - F1	+ F2 - F3	Тр.средства
Штабели	A324BT	
Мест: 19	E123ET 38 RUS	
Wm3:3.382	K831EO 38 RUS	

Рис. 4.6. СТК Терминал. Диалоговая форма «Акт»

Дата 26.06.06 09:50 № акта 097

Контролер Администратор

Контроль бревен на складе- F1

Отчет о контроле на складе - F2

Рис. 4.7. СТК Терминал. Диалоговая форма «Акт контроля на складе»

Дата 26.06.06 09:50 № акта 097

Тр.средство

Контроль бревен- F1

Контроль штабелей - F2

Отчет о контроле - F3

Рис. 4.8. СТК Терминал. Диалоговая форма «Акт контроля транспортного средства»

Вспомогательные диалоговые формы предоставляют доступ к различным справочникам, спискам и сервисным функциям СТК Терминал.

4.2.2. Логико-лингвистическая модель диалога

Логико-лингвистическая модель состоит из комплекта транслирующих ПОФ. Основу каждой транслирующей ПОФ составляет регулярная или LL(1)-

грамматика. Особенностью применения грамматики является таблица обозначений. Таблица обозначений необходима для сопоставления входных воздействий терминальным символам грамматики.

Пример таблицы обозначений:

№	Входное воздействие			Терм. символ	Описание
	ID объекта	ID события	Значение события ¹⁴		
1	IDC_NUMB	WM_KEYDOWN	0-9	ц	Ввод в поле «Номер акта»
2	DLG_ORG	WM_ACTIVATE		а	Открытие диалога «Организации»
...	...	...	...	...	...

Как видно из таблицы входное воздействие составляют три элемента: идентификатор объекта, идентификатор события и значение события. Под первым номером в таблице описано воздействие, которое происходит при вводе цифр в элемент управления IDC_NUMB, ему сопоставлен терминальный символ 'ц'. Некоторые параметры входного воздействия для вырожденных случаев могут опускаться. Так, под вторым номером в таблице описано воздействие с двумя параметрами и ему сопоставлен терминальный символ 'а'.

Таким образом, терминальные символы языка входных воздействий в диалоговой системе – это обозначения кортежей вида *<идентификатор объекта-получателя, идентификатор сообщения (событие), идентификатор параметра, значение параметра>*.

Нетерминальные символы в грамматике обозначены цепочками прописных символов (например, *START*, *END* и т.д.).

Суть логико-лингвистической модели диалога СТК Терминал – контроль правильных цепочек входных воздействий. Правильные цепочки входных воздействий в СТК Терминал описаны для основных диалогов и элементов управления с помощью регулярной или LL(1)- грамматик, а их контроль реализован в виде экземпляров КСТП. На рисунке 4.9. представлено два уровня логико-лингвистической модели диалога. Рассмотрим их подробно:

¹⁴ Под значением события понимается значение одного из параметров функции обработки событий. Во всех случаях обрабатываются значения одного из параметров.

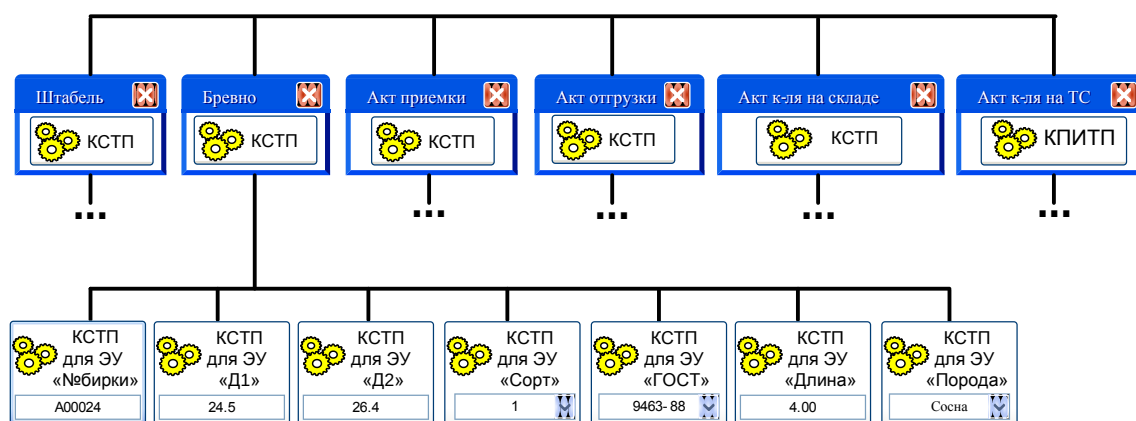


Рис. 4.9. СТК Терминал. Логико-лингвистическая модель

Первый уровень – уровень формы. КСТП на данном уровне контролирует входные воздействия к элементам управления. Грамматики первого уровня самая сложная и включает большое число терминальных и нетерминальных символов. Например, на рис. 4.10 представлена упрощенная схема вызова формы «Бревно». Символы А, В, С, ... на дугах схемы – терминальные символы. Таблица обозначений терминальных символов в данном случае имеет вид:

№	Входное воздействие			Терм. символ	Описание
	ID объекта	ID события	Значение события		
1	IDC_ADDLOG	WM_SETFOCUS		A	Получение фокуса кнопкой «Добавить»
2	IDC_NUMB	WM_SETFOCUS		B	Получение фокуса элементом «№ бирки»
3	IDC_LOGTYPE	WM_SETFOCUS		C	Получение фокуса элементом «Порода»
4	IDC_D1	WM_SETFOCUS		D	Получение фокуса элементом «Дмин»
5	IDC_D2	WM_SETFOCUS		E	Получение фокуса элементом «Дмакс»
6	IDC_GOST	WM_SETFOCUS		F	Получение фокуса элементом «ГОСТ»
7	IDC_SORT	WM_SETFOCUS		G	Получение фокуса элементом «Сорт»
8	IDC_LEN	WM_SETFOCUS		H	Получение фокуса элементом «Длина»
9	IDC_TNVED	WM_SETFOCUS		I	Получение фокуса элементом «ТНВЭД»
10	IDC_PREV	WM_SETFOCUS		G	Получение фокуса кнопкой «Пред, <F1>»
11	IDC_NEXT	WM_SETFOCUS		K	Получение фокуса кнопкой «След, F3>»
12	DLG_LOG	WM_CLOSE		L	Закрытие диалога «Бревно»
13	IDC_ITOG	WM_SETFOCUS		M	Получение фокуса элементом «Итого».

Логико-лингвистическая модель приведенной схемы вызовов для формы «Бревно» может быть описана грамматикой $G = \langle V_T, V_N, P, S \rangle$. Где:

$V_T = \{A, B, C, D, E, F, G, K, L, M\}$ – множество терминальных символов;

$V_N = \left\{ \begin{array}{l} \text{START, ADD, NUMB, TYPE,} \\ \text{D1, D2, GOST, SORT, LEN,} \\ \text{TNVED, PREV, NEXT, EXIT} \end{array} \right\}$ – множество нетерминальных символов;

$S = START$ – аксиома;

Множество правил P состоит из продукций:

$\langle START \rangle := A \langle ADD \rangle$	$\langle TNVED \rangle := A \langle ADD \rangle$
$\langle START \rangle := L \langle EXIT \rangle$	$\langle TNVED \rangle := G \langle PRED \rangle$
$\langle ADD \rangle := B \langle NUMB \rangle$	$\langle TNVED \rangle := L \langle EXIT \rangle$
$\langle NUMB \rangle := C \langle TYPE \rangle$	$\langle TNVED \rangle := K \langle NEXT \rangle$
$\langle TYPE \rangle := D \langle D1 \rangle$	$\langle NEXT \rangle := K \langle PRED \rangle$
$\langle D1 \rangle := E \langle D2 \rangle$	$\langle NEXT \rangle := M \langle START \rangle$
$\langle D2 \rangle := F \langle GOST \rangle$	$\langle PRED \rangle := M \langle ADD \rangle$
$\langle GOST \rangle := G \langle SORT \rangle$	$\langle PRED \rangle := M \langle START \rangle$
$\langle SORT \rangle := H \langle LEN \rangle$	$\langle EXIT \rangle := L$
$\langle LEN \rangle := I \langle TNVED \rangle$	

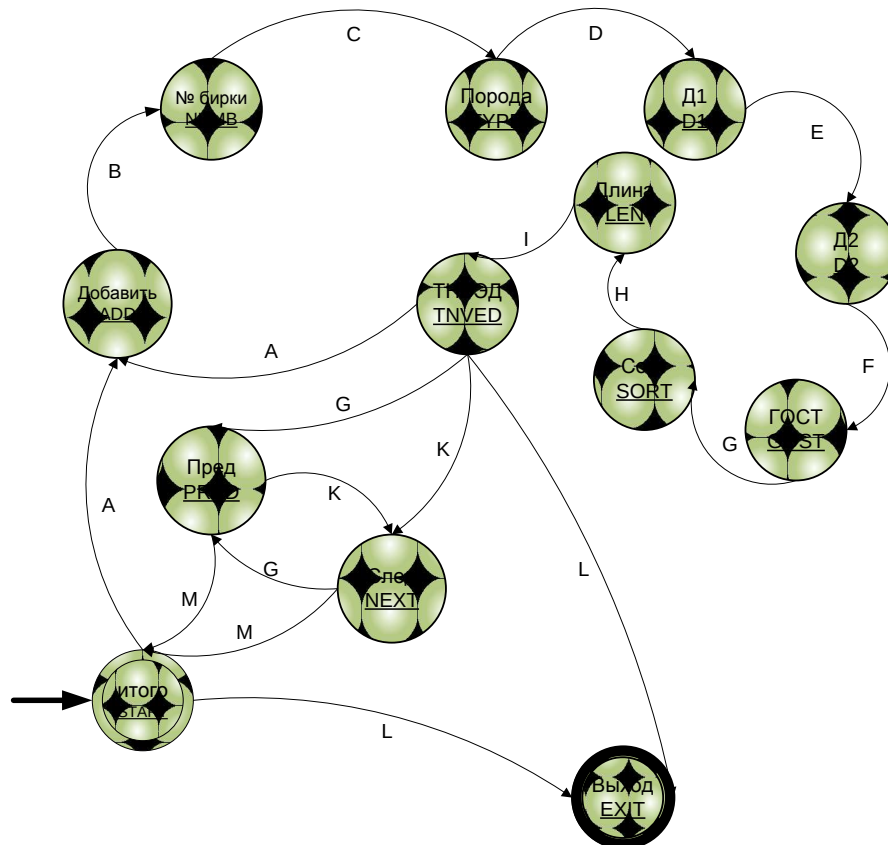


Рис. 4.10. Логико-лингвистическая модель диалога. Форма «Бревно».

Второй уровень – уровень элементов управления. КСТП на данном уровне реализует обработку входных воздействий для конкретного текстового поля или выпадающего списка. Например, для поля «Номер бирки» КСТП реализует форматно-логический контроль введенных данных в соответствии с грамматикой $G = \langle V_T, V_N, P, S \rangle$. Где $V_T = \{b, u\}$, $V_N = \{A, B, C, D, E, F\}$, $S = A$, множеством правил P состоит из продукций:

<A>:=б	<D>:=ц<E>
:=ц<C>	<E>:=ц<F>
<C>:=ц<D>	<F>:=ц

Таблица обозначений состоит из двух записей и имеет следующий вид:

№	Входное воздействие			Терм. символ	Описание
	ID объекта	ID события	Значение события		
1	IDC_NUMB	WM_KEYDOWN	0-9	ц	Ввод алфавитных символов
2	IDC_NUMB	WM_KEYDOWN	A-Za-z	б	Ввод цифровых символов

Особенностью функционирования логико-лингвистической модели является то, что входные воздействия обрабатываются соответствующими экземплярами КСТП по мере их возникновения. Обработка заключается в вызове метода *Prove* компонента. В качестве аргумента методу *Prove* передается терминальный символ. Терминальный символ получается из входного воздействия с помощью таблицы обозначений. Таблица обозначений представляет собой в программном коде последовательность условных операторов. Далее контролируется правильность распознаваемой конструкции. Транслирующая программа синтезируется в случае получения недопустимого входного воздействия. Назначение транслирующей программы – вывести диагностическое сообщение и инициализировать элементы логико-лингвистической модели для продолжения обработки.

Пример. Рассмотрим реализацию обработки входных воздействий для описанной грамматики элемента управления «Номер бирки». Транслирующая ПОФ без учета входной строки для данного элемента управления имеет вид:

$$\begin{array}{l}
\forall s : P(\delta, s), Q(A) \exists : R(\delta, s), Q(B) \\
\forall s : P(u, s), Q(B) \exists : R(u, s), Q(C) \\
\forall s : P(u, s), Q(C) \exists : R(u, s), Q(D) \\
\forall s : P(u, s), Q(D) \exists : R(u, s), Q(E) \\
\forall s : P(u, s), Q(E) \exists : R(u, s), Q(F) \\
\forall s : P(u, s), PT(\#), Q(F) \exists : RT(\#)[FINISH()] \\
\forall : RT(\#) \exists : F \\
\exists u : u = 1, P(, u), Q(A)[START()] \left\{ \begin{array}{l}
\forall s : P(u, s), Q(A) \exists : END()[error1()] \\
\forall s : P(\delta, s), Q(B) \exists : END()[error1()] \\
\forall s : P(\delta, s), Q(C) \exists : END()[error1()] \\
\forall s : P(\delta, s), Q(D) \exists : END()[error1()] \\
\forall s : P(\delta, s), Q(E) \exists : END()[error1()] \\
\forall s : P(\delta, s), Q(F) \exists : END()[error2()] \\
\forall s : P(u, s), Q(F) \exists : END()[error2()] \\
\forall : END() \exists : STOP()
\end{array} \right.
\end{array}$$

Транслирующие подпрограммы START, FINISH, error1, error2 описываются в отдельном классе. В данном случае они реализуют обработку недопустимых значений и подсветку элемента. Ниже приведен класс, реализующий транслирующие подпрограммы:

```

class IDC_NUMB_SubRoutines : CTPOUnit.ISubRoutines
{
    public void START(ref Control nCtrl)
    {
        nCtrl.Font.Bold = true;
    }
    public void FINISH(ref Control nCtrl)
    {
        nCtrl.Font.Bold = false;
    }
    public int error1(ref Control nCtrl)
    {
        MessageBox.Show("Введено недопустимое значение!");
        nCtrl.Text = "";
    }
    public void error2(ref Control nCtrl)
    {
        MessageBox.Show("Введен лишний символ!");
    }
}

```

В системе программирования на платформе .Net входные воздействия можно обрабатывать стандартной процедурой *PreFilterMessage* в классе, наследуемом от интерфейса *IMessageFilter*. Для элемента управления «Номер бирки» данный класс имеет вид:

```
public class MyMessageFilter : IMessageFilter
{
    public bool PreFilterMessage(ref Message m)
    {
        Control control = Control.FromChildHandle(m.HWnd);
        if (control.Name == "IDC_NUMB")
        {
            if (m.Msg == CTPOUnit.WM_KEYDOWN) //нажатие кнопки
            {
                if (pMsg-> wParam > 65 /*A*/ && pMsg-> wParam < 128 /*z*/)
                {
                    //запуск КСТП
                    return !m_CTPOUnit.Prove('6');
                }
                else if (pMsg-> wParam > 48 /*0*/ && pMsg-> wParam < 57 /*9*/ )
                {
                    //запуск КСТП
                    return !m_CTPOUnit.Prove('ч');
                }
                else return true; //если недоустимый символ, то отклоняем его
            }
            else if (m.Msg == CTPOUnit.WM_KILLFOCUS) //потеря фокуса
            {
                //Если ПОФ доказана разрешить смену фокуса
                return !m_CTPOUnit.Prove();
            }
            else if (m.Msg == CTPOUnit.WM_SETFOCUS) //получение фоукса
            {
                m_CTPOUnit.Prove(control.Text);
            }
            else return false;
        }
    }
}
```

Обработка входных воздействий с помощью КСТП для элемента управления «Номер бирки» заключается в проверке вводимых данных на соответствие формату, подсветке элемента управления и выводе сообщений об ошибках.

4.2.3. Реализация модели

Логико-лингвистическая модель для СТК Терминал состоит из 49 экземпляров КСТП настроенных для обработки воздействий диалоговых форм и элементов управлений. Применение экземпляра КСТП заключается в запуске

метода *Prove*, анализе результатов контроля и исполнении синтезированной транслирующей программы.

Интеграция КСТП в диалоговые формы и в элементы управления заключается в следующих действиях:

1. Реализация класса транслирующих подпрограмм. Класс транслирующих подпрограмм наследуется от интерфейса *ISubRutines*. Данный интерфейс предоставляется КСТП и содержит обязательные спецификации процедур *START* и *FINISH*.
2. Настройка КСТП на грамматику и класс транслирующих подпрограмм. Часть настроек КСТП задается в диалоговом режиме (например, грамматика), а часть программируется в коде инициализации диалоговых форм или элементов управления (например, ссылки на класс транслирующих подпрограмм и начальное состояние КСТП).
3. Описание фильтров для обработки входных воздействий. Фильтр для обработки входных воздействий представляет собой класс-наследник стандартного интерфейса .NET *IMessageFilter*. Данный интерфейс содержит процедуру обработки входных воздействий *PreFilterMessage*. В процедуре *PreFilterMessage* описывается таблица обозначений, происходит перехват входных воздействий и вызов метода *Prove* КСТП.

4.3. Выводы

В данной главе рассмотрено применение КСТП при решении двух практических задач: разработка учебных трансляторов и контроль входных воздействий в составе диалогового средства контроля входных воздействий Терминал. Применение КСТП при решении приведенных задач позволило подойти к процессу разработки на качественно новом уровне.

Постановка и схема решения задачи контроля диалога человека и объектной программы составляет одно из защищаемых положений диссертационной работы. Данный результат опубликован в [12-20]. Получен акт о применении программного средства для контроля входных воздействий в составе СТК Терминал (см. приложение 2 к настоящей работе).

Заключение

Выделение частного случая конструктивного фрагмента исчисления позитивно-образованных формул для задач трансляции позволяет унифицировать применение различных грамматик и автоматов для решения этих задач. Основа унификации – структура транслирующей позитивно-образованной формулы, правило ω и возможность его адаптации применительно к конкретным грамматикам и стратегиям анализа входных цепочек за счет новой стратегии вывода, а также идея синтеза транслирующей программы из заданного набора транслирующих подпрограмм. В диссертационной работе возможность унификации продемонстрирована для двух классов формальных грамматик – регулярных грамматик и LL(1)-грамматик.

Инструментарий для разработки и применения транслирующих позитивно-образованных формул при решении прикладных задач оформлен в виде компонента синтеза транслирующих программ (КСТП). Компонент применен при решении двух практических задач – разработке технологии конструирования учебных трансляторов и контроля последовательностей цепочек входных воздействий в специализированной диалоговой системе.

Рассмотрим некоторые варианты использования КСТП в прикладных программах:

1. **Учебные трансляторы.** КСТП согласно схеме, приведенной в [3 с. 168], можно использовать для создания несложных трансляторов. Настройка КСТП происходит в диалоге и не требует дополнительного кода по обработке входной строки, в результате программный код учебного транслятора будет небольшим. Использование КСТП позволит учащимся быстро овладеть не одной-двумя, а более широким спектром формальных грамматик, даже не разрабатывая учебных трансляторов.

- 2. Обработка входных воздействий в диалоговой системе.** Диалог в современных прикладных диалоговых системах, как правило, представляет собой сложные цепочки взаимосвязанных входных воздействий. Штатные средства программирования обработки входных воздействий в современных системах объектно-ориентированного программирования не обеспечивают анализа контекста, в котором имеет место то или иное воздействие (см. например, [6]). В диалоговых системах входные воздействия представляют собой последовательности событий. С помощью КСТП можно разбирать последовательности таких событий, представляя их в виде входной цепочки. При выявлении некорректной последовательности событий (входной цепочки) КСТП может синтезировать сообщение об ошибке или код обработки данного сообщения.
- 3. Обработка запросов в Web-приложениях.** По аналогии с обработкой входных воздействий в диалоговой системе, КСТП можно использовать для обработки запросов пользователя на стороне сервера в Web-приложениях. Результатом обработки запросов, например, может быть набор ссылок или программный код, обрабатывающий запросы.
- 4. Генерация справочного материала.** Генерация текстов на естественном языке из заданных фрагментов является актуальной задачей [51, 60, 87, 88]. С помощью КСТП можно построить систему генерации контекстно-зависимой справки. Например, справку можно генерировать во время обработки входных воздействий пользователя в диалоговой системе. При этом транслирующие подпрограммы будут представлять собой фрагменты справочного текста. Доказательство транслирующей ПОФ позволит синтезировать справку, содержание и

структура которой будут зависеть от последовательности предшествующих действий пользователя.

Приведенную идею синтеза программ на основе моделей трансляции и автоматического синтеза программ предполагается развивать в двух направлениях:

- Разработка алгоритмов построения транслирующих позитивно-образованных формул для других типов КС-грамматик (например, LR-грамматик, недетерминированных грамматик [79] и грамматик с количеством символов предпросмотра больше 1 [101]).
- Модификация алгоритма синтеза транслирующей программы. Представляется возможным и полезным разработать алгоритм синтеза ветвящейся и циклической транслирующей программы. Это позволит упростить технологию работы с транслирующей ПОФ за счет вывода из ее состава входной цепочки.

В рамках диссертации получены и выносятся на защиту следующие результаты:

1. Подмножество языка ПОФ, стратегия вывода τ , постановка и принципиальная схема решения задачи трансляции доказательством существования и синтезом транслирующей программы на основе исчисления ПОФ.

2. Алгоритмы построения распознающих позитивно-образованных формул по заданным грамматикам: регулярной правосторонней детерминированной грамматике и LL(1)-грамматике. Для этих грамматик доказана эквивалентность доказательства теоремы существования транслирующей программы преобразованиями распознающей позитивно-образованной формулы и грамматического вывода.

3. Инструментальное средство – компонент синтеза транслирующих программ – для конструирования транслирующих позитивно-образованных формул, а также применения этих формул в составе прикладных программ.

4. Постановка и схема решения задачи контроля диалога человека и объектной программы на основе формальной спецификации требований к диалогу двухуровневой иерархией формальных языков, каждый из которых задается ПОФ. Задача контроля диалога решена применением КСТП.

Литература

1. Агуров П.В. С#. Сборник рецептов. СПб.: БХВ-Петербург, 2007. С. 432.
2. Астраханцев Ф.П. Разработка приложений для мобильных устройств на основе технологий Microsoft // Компьютерные инструменты в образовании– СПб.: Изд-во ЦПО "Информатизация образования", 2005. № 7. С. 59-65.
3. Ахо А., Сети Р., Ульман Дж. Компиляторы: принципы, технологии и инструменты: Пер. с англ. М.: Изд-дом «Вильямс», 2001. С. 768.
4. Ахо А., Ульман Дж. Теория синтаксического анализа, перевода и компиляции. Т. 1: Синтаксический анализ. М.:Мир. 1978. С. 612.
5. Ахо А., Ульман Дж. Теория синтаксического анализа, перевода и компиляции. Т. 2: Компиляция. М.:Мир. 1978. С.487.
6. Бабаев А.А. Автоматный интерфейс. Новый метод создания логики интерфейса. RSDN Magazine №5. 2005. URL: <http://www.rsdn.ru/article/UI/AutoUI.xml> (дата обращения: 12 мая 2006)
7. Бейзер Б. Тестирование черного ящика. Технологии функционального тестирования программного обеспечения и систем. СПб.: Питер. 2004. С. 318.
8. Бильгаева Н.Ц. Теория алгоритмов, формальных языков, грамматик и автоматов: Учебное пособие. Улан-Удэ: Изд-во ВСГТУ. 2000. С. 51.
9. Бутаков М.И. Компонент построения и исполнения транслирующих программ на основе позитивно-образованных формул // Материалы IX школы-семинара молодых ученых «Математическое моделирование и информационные технологии» 22-27 октября 2007 года. – Иркутск: Изд-во Инст. дин. сист. и теор. управ-я СО РАН. 2007. С.36-39.
10. Бутаков М.И. Компонент построения и исполнения транслирующих программ на основе позитивно-образованных формул. // Прикладные информационные технологии и системы. – Иркутск: изд-во Иркут. гос. ун-та, 2009. С.4-27.

- 11.Бутаков М.И., Курганский В.И. Контроль диалога объектных программ на основе позитивно-образованных формул // Вопросы современной науки и практики. – Университет им. В.И. Вернадского, 2010, № 4-6(29). С. 106-115.
- 12.Бутаков М.И. Об одной модели диалога в объектных системах // Материалы VIII школы-семинара молодых ученых «Математическое моделирование и информационные технологии: управление, искусственный интеллект, прикладное программное обеспечение, технологии программирования». – Иркутск: Изд-во Инст. дин. сист. и теор. управ-я СО РАН. 2006. С. 37-38.
- 13.Бутаков М.И. Учебный стенд по методам синтаксически управляемой трансляции // Вестник Иркутского университета. Специальный выпуск: Материалы научно-теоретической конференции молодых ученых, посвященной 60-летию Великой Победы. – Иркутск: Изд-во Иркут. гос. Ун-та. 2005. С.49-51.
- 14.Бутаков М.И. Инструментальное средство синтеза и исполнения транслирующих программ на основе позитивно-образованных формул // Инфокоммуникационные и вычислительные технологии и системы: Материалы семинара молодых ученых в рамках III международной конференции, Улан-Удэ:Изд.-полигр.комплекс ВСГАКИ. 2010. С. 39-42.
- 15.Бутаков М.И. Инструментальное средство синтеза и исполнения транслирующих программ на основе позитивно-образованных формул // 4-я Всероссийская мультikonференция по проблемам управления: Материалы 4-й Всероссийской мультikonференции. Т.1. – Таганрог: Изд-во ТТИ ЮФУ. 2011. С. 27-29.
- 16.Бутаков М.И., Контроль входных воздействий в специализированной системе. // Информационные и математические технологии в науке и управление. Труды XIV Байкальской Всероссийской Конференции

«Информационные и математические технологии в науке и управлении».
Часть II. – Иркутск: ИСЭМ СО РАН. 2009. С. 105-115.

17. Бутаков М.И., Курганская О.В. Лексический анализ: обучение моделированию и программированию // Информационные технологии моделирования и управления, 2008, №2 (45). С.128-135.
18. Бутаков М.И. , Курганская О.В. Решение учебных задач трансляции на основе позитивно-образованных формул // Системы управления и информационные технологии. 2008. № 3.1 (33). С. 124-128.
19. Бутаков М.И., Курганский В.И. О решении задачи трансляции планированием вычислений на основе позитивно-образованных формул // Системы управления и информационные технологии. 2007. № 3.1 (29). С. 120-123.
20. Бутаков М.И., Курганский В.И. Об одной модели трансляции на основе позитивно-образованных формул // Информационные технологии моделирования и управления. 2007. № 6 (40). С. 663-672.
21. Васильев С.Н. Вывод теорем на основе логических уравнений и типизации переменных // Функции Ляпунова и их применение. Новосибирск: Наука, Сиб.отделение, 1987). С. 60-84.
22. Васильев С. Н., Жерлов А. К., Федосов Е. А., Федунцов Б. Е. Интеллектуальное управление динамическими системами. М.: Физматлит. 2000. 352 с.
23. Васильев С.Н., Черкашин Е.А. Интеллектуальное управление телескопом // Сибирский журнал индустриальной математики, т. 1, № 2. 1998. С. 81-98
24. Васильев С.Н., Гулямов Ш.Б. Жерлов А.К., Литвинов Ю.Ф. О логических средствах системы планирования вычислений «ПАСАД» // Алгоритмы. Автоматизация программирования (Сборник научных трудов). – Вып.66. – Ташкент: АН Уз.ССР. 1988. С.97-112.
25. Васильев С.Н., Жерлов А.К. Развитие логических методов автоматизации решения задач с приложениями к управлению динамическими системами и планированию действий [Электронный ресурс]: Труды международной

- конференции «Интеллектуальные системы и информационные технологии управления». Псков. 2000 <http://www.inftech.webservis.ru/> (11 фев. 2008)
- 26.Васильев С.Н., Жерлов А.К. Об исчислении типово-кванторных формул// ДАН. – Т.343, № 2. 1995. С. 583-585.
- 27.Васильев С.Н. Методы и программные средства синтеза математических теорем //Инструментальные системы и моделирование. – Новосибирск: Наука.Сиб.отд-ние.1988. С. 4-27.
- 28.Васильев С.Н. Интеллектуальный подход к автоматизации проектных расчетов сложных управляемых систем // Оптимизация, управление, интеллект. 2000.Вып 4. С. 111–126.
- 29.Вилле К. Представляем C#. – М.:ДМК Пресс. 2001. – 192 с.
- 30.Гросс М., Лантен А. Теория формальных грамматик. – М.: ИЗДАТЕЛЬСТВО «МИР», 1972. – 294 с.
- 31.Гуров В.С., Мазин М.А., Шалыто А.А. Текстовый язык автоматного программирования // Тезисы докладов международной научной конференции, посвященной памяти профессора А.М. Богомолова "КОМПЬЮТЕРНЫЕ НАУКИ И ТЕХНОЛОГИИ". Саратов: СГУ. 2007. С. 66-69.
- 32.Ершов А.П. Научные основы доказательного программирования: Научн. Сообщ. // Вестн. АН СССР. 1984. № 10. С. 9-19.
- 33.Канжелев С.Ю., Шалыто А.А. Автоматическая генерация автоматного кода // Информационно-управляющие системы.2006. № 6. С.35-42.
- 34.Карпов Ю.Г. Теория и технология программирования. Основы построения трансляторов. – СПб.: БХВ-Петербург, 2005. – 272 с.
- 35.Карпов Ю.Г. Теория автоматов. – СПб.: Питер, 2003. – 208 с.
- 36.Компаниец Р.И., Маньков Е. В., Филатов Н.Е. Системное программирование. Основы построения трансляторов // Учебное пособие

- для высших и средних учебных заведений. – СПб.: КОРОНА принт, 2000. – 256 с.
- 37.Кристиансен Т., Торкингтон Н. Perl. Сборник рецептов. Для профессионалов. 2-е изд. – СПб.: Питер, 2004. – 928 с.
- 38.Кудрявцев Е.М. AutoLISP. Программирование в AutoCAD 14. – М.: «ДМК», 1999. – 568 с.
- 39.Курганский В.И., Бутаков М.И. Лексический анализ на основе регулярных грамматик и конечных автоматов. – Иркутск: Иркут. гос. ун-т, 2006. – 20 с.
- 40.Курганский В.И., Бутаков М.И. Математические модели лексического и синтаксического анализа для студентов младших курсов. // Обозрение прикладной и промышленной математики. Том 13, выпуск 3. Москва: Изд-во «ОПиМП». 2006. С.77-78.
- 41.Курганский В.И., Бутаков М.И. Синтаксический анализ на основе LL(1)-грамматик и автоматов с магазинной памятью. – Иркутск : Иркут. гос. ун-т, 2006. – 32 с.
- 42.Айзерман М.А., Гусев Л.А., Розоноэр Л.И., Смирнова И.М., Таль А.А. Логика, Автоматы, Алгоритмы; под ред. Н.А. Королева. – М.: Физматгиз, 1963. – 556 с.
- 43.Марчук Ю.Н. Основы компьютерной лингвистики. Учебное пособие. М.:Изд-во МПУ «Народный учитель», 2000. – 226 с.
- 44.Минц Г.Е., Тыугу Э.Х. Полнота правил структурного синтеза // Докл. АН СССР. 1982. Т.265, № 6. – С.41-60.
- 45.Митичкин С.А. Практика программирование в среде 1С:Предприятие 7.7 – М.:Издательский Дом «КомБук», 2004 – 272 с.
- 46.Гайдышев И.П. Решение научных и инженерных задач средствами Excel, VBA и C/C++. – СПб.:Издательство БХВ-Петербург, 2004 – 512 с.

- 47.Мозговой М.В. Классика программирования: алгоритмы, языки, автоматы, компиляторы. Практический подход. – СПб.:Наука и Техника,2006. – 320 с.
- 48.Молчанов А.Ю. Системное программное обеспечение: Учебник для вузов. – СПб.:Питер, 2006. – 395с.
- 49.Опарин Г.А., Феоктистов Д.Г. САТУРН – метасистема для автоматизации конструирования и поддержки функционирования пакетов прикладных программ с автоматическим планированием вычислительного процесса // Пакеты прикладных программ. Итоги применения. Новосибирск: Наука. 1986. С.12-21.
- 50.Опарин Г.А. К теории планирования вычислительного процесса в пакетах прикладных программ // В. кн.: Пакеты прикладных программ. Методы и разработки. Новосибирск: Наука. 1981. С. 5-20.
- 51.Поспелов Д.А. Логические методы анализа и синтеза схем// Изд. 3-е, перераб. и доп. – М., «Энергия». 1974. – 368 с.
52. Терехов А.А., Вояковская Н. В., Булычев Д., Москаль А Разработка компиляторов на платформе .NET [Электронный ресурс]– СПб, 2001 – <http://www.ict.edu.ru/ft/005128/> (18 сент. 2008)
- 53.Рихтер Дж, Программирование на платформе Microsoft .Net Framework: Пер. с англ. – 2-е изд., испр. – М.: Издательско-торговый дом "Русская Редакция", 2003 - 512 с.
- 54.Румянцев М.И. К вопросу о построении лингвистической модели бизнес-процессов коммерческого банка // Информационные технологии моделирования и управления. 2007. №6 (40). С.663-672.
- 55.Иво Салмре. Программирование мобильных устройств на платформе .NET Compact Framework: Пер. с англ. – М.:Издательский дом "Вильямс", 2006. – 736 с.
- 56.Корольков Ю.Д., Курганский В.И., Бутаков М.И. Свидетельство об официальной регистрации программы ЭВМ № 2007611658, Российская

Федерация. Набор компонентов ParseUnit. Заявитель и правообладатель – ин-т математики, экономики и информатики Иркут. ун-та; заявл. 9.01.2007, опубл. 20.04.2007.

57. Севостьянова Д.В., Курганский В.И. Восходящий грамматический разбор на основе позитивно-образованных формул // Вестник Иркутского государственного университета. Материалы научной студенческой конференции. – Иркутск: Изд-во Иркут. гос. Ун-та. 2009. С. 163-165
58. Смит Родерик. Полный справочник по FreeBSD : Пер. с англ. – М.: Издательский дом «Вильямс», 2005. – 672 с.
59. Соколов А.П. Системы программирования: теория, методы, алгоритмы; Учеб. Пособие. – М.: Финансы и статистика, 2004. – 320 с.
60. Соколова Е.Г., Болдасов М.В. Автоматическая генерация текстов на ЕЯ (портрет направления) [Электронный ресурс]. – <http://www.dialog-21.ru/Archive/2004/Sokolova.htm> (декабрь 2008)
61. Стюарт Рассел, Питер Норвиг. Искусственный интеллект. Современный подход, 2-е изд.: Пер. с англ. – М.: Издательский дом «Вильямс», 2007. – 1408 с.
62. Троелсен. Э. С# и платформа .Net. Библиотека программиста. – СПб.: Питер, 2004. – 796 с.
63. Тыугу Э.Х. Концептуальное программирование. – М.: Наука, 1984. – 256 с.
64. Тэллес М., Хеих Ю. Наука отладки: Пер. с англ. – М.: КУДИЦ-ОБРАЗ, 2003. – 560 с.
65. Фаулер М. Предметно-ориентированные языки программирования. – М.: Издательский дом «Вильямс», 2011. – 576 с.
66. Хантер Р. Основные концепции компиляторов. :Пер. англ. – М.: Издательский дом «Вильямс», 2002. – 256 с.
67. Хантер Р. Проектирование и конструирование компиляторов: Пер. с англ. Предисл. В.М. Савинкова. – М.: Финансы и статистика, 1984 – 232 с.

68. Хомский Н.О. некоторых формальных свойствах грамматик // "Кибернетический сборник", вып. 5, ИЛ. 1962. С. 121-227.
69. Хопкрофт Д., Мотвани Д., Ульман Д. Введение в теорию автоматов, языков и вычислений, 2-е изд.: Пер. с англ. – М.: Издательский дом "Вильямс", 2002. – 528 с.
70. Чень Ч., Ли Р. Математическая логика и автоматическое доказательство теорем: Пер. с англ. Ли. Под ред. С.Ю. Маслова. – М.: Наука. Главная редакция физико-математической литературы, 1983. – 360 с.
71. Черкашин Е.А. Программная система КВАНТ/1 для автоматического доказательства теорем: автореф. дис. канд. техн. наук: 05.13.11. Иркутск, 1999. – 17 с.
72. Черкашин Е.А. Система АДТ КВАНТ/3 на разделяемых структурах данных // Материалы IX школы-семинара молодых ученых «Математическое моделирование и информационные технологии» 22-27 октября 2007 года. – Иркутск: Изд-во Инст. дин. сист. и теор. управ-я СО РАН, 2007. – С.173-177
73. Чернецки К., Айзенкер У. Порождающее программирование: методы, инструменты, применение. Для профессионалов. – СПб.: Питер, 2005. – 731 с.
74. Шалыто А.А. Автоматно-ориентированное программирование. // Материалы IX Всероссийской конференции по проблемам науки и высшей школы "ФУНДАМЕНТАЛЬНЫЕ ИССЛЕДОВАНИЯ В ТЕХНИЧЕСКИХ УНИВЕРСИТЕТАХ". – СПб.: изд-во Политехнического университета. 2005. – С. 44-52.
75. Шамгунов Н.Н. Разработка методов проектирования и реализация поведения программных систем на основе автоматного подхода: автореф. дис. канд. техн. наук: 05.13.13. – Санкт-Петербург, 2004. – 18 с.

76. Rodger S.H. A collection of tools for making automata theory and formal languages come alive // ACM SIGCSE Bulletin. Volume 29 , Issue 1. 1997. p. 15-19.
- 77.. A Scalable Formal Method for Design and Automatic Checking of User Interfaces // ACM Transactions on Software Engineering and Methodology, Vol. 14, No. 2, 2005 p. 124-167.
78. Gregory Abowd, Jonathan Bowen. User Interface Languages: A Survey of Existing Methods // Technical Report PRG-TR-5-89, Programming Research Group, Oxford University, October 1989 – p. 65.
79. Aho A.V., Johnson S.C, Ullman J.D. Deterministic parsing of ambiguous grammars // Comm. ACM 18:8. 1975. p. 441-452.
80. Butyrin S.A., Makarov V.P., Mukumov R.R., Somov Ye.I., Vassilyev S.N An Expert System for Design of Spacecraft Attitude Control System // Artificial Intelligent in Engineering, Vol. 11., No 1. 1997.p. 49-59.
81. Anthony A. Aaby. Compiler Construction using Flex and Bison // Walla Walla College. Version of February 25, 2004. p. 96.
82. Cherry L.L. Writing tools // IEEE Trans. on Communications COM-30:1. 1982. p. 100-104.
83. Cormack G.V. Data compression using dynamic Markov modeling // Computer J., – Vol. 30, № 6, 1987. p. 541-550.
84. Bruno Ginoux. DESCARTES: An Automatic Programming System for Algorithmically Simple Programs // International Workshop on Software Specifications & Design. Proceedings of the 9th international workshop on Software specification and design, 1998, p. 106-109.
85. Free Compiler Construction Tools. [Электронный ресурс]. –: <http://thefreecountry.com/> (1 дек. 2007)
86. Harrison M.D., Duke D.J. A review of formalisms for describing interactive behaviour// In Software Engineering and Human-Computer Interaction,

- volume 896 of Lecture Notes in Computer Science, Springer-Verlag. 1995. p. 49-75.
- 87.Iván I. Garibay, Generating text with a theorem prover // ACM International Conference Proceeding Series; Vol. 5, Proceedings of the workshop on Student research table of contents, 2000. p. 13-18.
- 88.Iván I. Garibay. Automatic generation of natural language documentation from statecharts // Master's thesis, University of Central Florida. 2000. p. 13-18.
- 89.Jarvis J.F. Feature recognition in line drawings using regular expressions // Proc. 3rd Intl. Joint Conf. on Pattern Recognition, 1976. p. 189-192.
- 90.Jeff Prosise. Programming Microsoft .NET // MS Press, 2002. p. 816.
- 91.Jacques Cohen, Timothy J. Hickey. Parsing and compiling using Prolog // ACM Transactions on Programming Languages and Systems (TOPLAS). Volume 9 Issue 2, April. 1987: p.125-163.
- 92.John E. Hopcroft and Jeffrey D. Ullman. Introduction to automata theory, languages, and computation //Addison-Wesley Publishing, Reading Massachusett. -2nd ed. 2001. p. 521.
- 93.Manna, Z., Waldinger Z. Fundamental of Deductive Program Synthesis // IEEE Transaction on Software Engineering, Vol. 18(8), 1992. p. 674-704.
- 94.Manna, Z., Waldinger R. Towards automatic program synthesis // Comm. ACM, 14(3). 1971. p. 151-164.
- 95.Marjan Mernik, Nikolaj Korbar, Viljem Žumer LISA: a tool for automatic language implementation // ACM SIGPLAN Notices, Volume 30 , Issue 4. 1995. p. 71-79.
- 96.Oparin G., Novopashin A. Boolean Modeling of Action Planning in Distributed Computing Systems // Journal of Computer and Systems Sciences International. 2004. p.25-30.
- 97.Phyllis Reisner. Formal Grammar and Human Factors Design of an Interactive Graphics Sysyrem // IEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. SE-7, №.2, march. 1981. p. 229-240

98. Resler R.D., Deaver D.M. VCOCO: a visualisation tool for teaching compilers. Annual Joint Conference Integrating Technology into Computer Science Education // Proceedings of the 6th annual conference on the teaching of computing and the 3rd annual conference on the teaching of computing and the 3rd annual conference on Integrating technology into computer science education: Changing the delivery of computer science education, Dublin City Univ., Ireland, 1998. p. 199-202.
99. Sami Khuri, Yanti Sugono. Animating Parsing Algorithms // ACM SIGSE Bulletin, Volume 30, Issue 1. 1998. p. 232-236.
100. Stephen A. Blythe. James Michael C., Rodger Susan. H. LLparse and LRparse: visual and interactive tools for parsing // ACM SIGCSE Bulletin, Volume 26, Issue 1. 1994, p. 208-212.
101. Terence J., Russell W. LL and LR translators need $k > 1$ lookahead // ACM SIGPLAN Notices, Volume 31, Issue 2. 1996: p. 27-34.
102. Terry P.D., Compilers and Compiler Generators, an introduction with C++ [Электронный ресурс] Rhodes University, 1996-2000 – <http://www.scifac.ru.ac.za/compilers/> (2004, October 21).
103. Tyugu E., Algorithms and Architectures of Artificial Intelligence. IOS Press, 2007 – p.171.
104. Vassilyev S.N. Machine Synthesis of Mathematical Theorems // J. Of Logic Programming Vol. 9, № 2&3. 1990. p. 235-266.
105. Vassilyev S.N., Davydov A.V., Zherlov A.K. Intelligent control via new efficient logics // Proc. of the 17th IFAC World Congress. Seoul (Korea). 2008. p. 13713-13718.
106. Young R.M., Barnard R. M., Simon T. How would your favorite user model cope with these scenarios? // SIGCGI Bulletin, (20). 1989. p. 51-55.

Приложение 1. Множество направляющих символов для LL(1)-грамматик

Рассмотрим LL(1)-грамматику $G = \{V_T, V_N, P, S\}$. Введем в рассмотрение множество $DS(A, \alpha)$, для правила $(A \rightarrow \alpha) \in P$, $A \in V_N$, $\alpha \in (V_T \cup V_N)^*$. Формально это множество определяется как объединение двух множеств $First(\alpha)$ и $Follow(A)$ [42, 4]:

$$DS(A, \alpha) = First(\alpha) \cup Follow(A)$$

Множество $First(\alpha)$ называют множеством символов-предшественников для строки α и определяют так:

$$First(\alpha) = \left\{ a \mid a \in V_T, \alpha \in V^+, \alpha \xRightarrow{*} a\beta, \beta \in V^* \right\}$$

Из этой записи следует, что $First(\alpha)$ - это множество терминальных символов, которые получаются как головные символы строк, выводимых из непустой строки α .

Алгоритм вычисления множества $First(\alpha)$ имеет вид:

Повторять пока нет изменений:

1. Если $\alpha = t$, $t \in V_T$, тогда $First(\alpha) = \{t\}$;
2. Если $\alpha = K$, $K \in V_N$ и $(K \rightarrow \varepsilon) \in P$, тогда добавить ε в $First(\alpha)$;
3. Если $\alpha = Y_1, \dots, Y_k$, для $i = 1, \dots, k-1$ выполнить
 - а. Если $\varepsilon \in First(Y_1) \wedge \dots \wedge \varepsilon \in First(Y_i)$ и $a(\neq \varepsilon) \in First(Y_{i+1})$, тогда добавить a в $First(\alpha)$;
 - б. Если $\varepsilon \in First(Y_1) \wedge \dots \wedge \varepsilon \in First(Y_k)$, тогда добавить ε в $First(\alpha)$;

Множество $Follow(A)$ назовем множеством символов-последователей для нетерминала A и определим его так:

$$Follow(A) = \left\{ a \mid a \in V_T, \alpha \xRightarrow{*} \varepsilon, S \xRightarrow{*} xAay; S, P \in V_N; x, y \in V^*; S - \text{аксиома} \right\}$$

Эта запись означает, что $Follow(A)$ - это множество терминалов, которые могут появляться в сентенциальных формах непосредственно за нетерминалом A . Множество $Follow(A)$ принимается во внимание только в том случае, если существует вывод $\alpha \xRightarrow{*} \varepsilon$. Сентенциальная форма – это любое порождение аксиомы грамматики [92 с. 178].

Обозначим S – аксиома грамматики, $\#$ - символ конца строки. Тогда алгоритм вычисления всех множества $Follow$ для грамматики G имеет вид:

Повторять пока нет изменений:

1. Добавить $\#$ в $Follow(S)$
2. Если имеет место продукция $A \rightarrow \alpha B \beta$, $A, B \in V_N$, $\alpha, \beta \in V_T$, тогда добавить все из $First(\beta)$ (за исключением ε) в $Follow(B)$;
3. Если имеет место продукция $A \rightarrow \alpha \beta$ или продукция $A \rightarrow \alpha B \beta$, $A, B \in V_N$, $\alpha, \beta \in V_T$, где $\varepsilon \in First(\beta)$, тогда добавить все из $Follow(A)$ в $Follow(B)$;

Рассмотрим пример. Пусть дана грамматика $G = \{V_T, V_N, P, S\}$, где $V_T = \{a, b, c\}$, $V_N = \{A, B, C\}$, $S = A$, $P = \{A \rightarrow BC \ A \rightarrow Cb \ B \rightarrow aB \ B \rightarrow \varepsilon \ C \rightarrow c \ C \rightarrow bC\}$ вычислим множество направляющих символов:

$$DS(A, BC) = First(BC) \cup Follow(A) = First(BC) = \{a, b, c\}$$

$$DS(A, Cb) = First(Cb) \cup Follow(A) = First(Cb) = \{b, c\}$$

$$DS(B, aB) = First(aB) \cup Follow(B) = First(aB) = \{a\}$$

$$DS(B, \varepsilon) = First(\varepsilon) \cup Follow(B) = Follow(B) = \{b, c\}$$

$$DS(C, c) = First(c) \cup Follow(C) = First(c) = \{c\}$$

$$DS(C, bC) = First(bC) \cup Follow(C) = First(bC) = \{b\}$$

Приложение 2. Акт об использовании результатов работы



Федеральная таможенная служба
Центральное экспертно-криминалистическое
таможенное управление
**ЭКСПЕРТНО-КРИМИНАЛИСТИЧЕСКАЯ
СЛУЖБА**
-РЕГИОНАЛЬНЫЙ ФИЛИАЛ
г. Иркутск

664039, ул. Гоголя, 53
Тел.: 135-253, Факс: (8-395-2) 38-72-59
ОКПО 29449896, ОГРН 1037739150866
ИНН/КПП 7708197861/381202001

00.02.2009 № 05-18/154

Директору ИДСТУ СО РАН
ч.-к. РАН Бычкову И.В.

664033, Иркутск,
ул. Лермонтова, 134, а/я 292

АКТ

Об использовании результатов диссертационной работы Бутакова М. И.

Настоящим актом подтверждается, что разработанная в рамках диссертационной работы аспиранта ИДСТУ СО РАН Бутакова Михаила Игоревича логико-лингвистическая модель диалога используется в составе средства таможенного контроля (СТК) «Терминал». Программная реализация логико-лингвистической модели диалога выполнена с помощью компонента синтеза транслирующих программ.

Средство таможенного контроля Терминал предназначено для автоматизации учета и таможенного контроля лесоматериалов необработанных. Логико-лингвистическая модель диалога обеспечивает контроль правильности действий оператора и существенно повышает надежность и дружелюбность интерфейса СТК «Терминал».

СТК «Терминал» применялось при экспериментальном опробовании новой системы таможенного контроля лесоматериалов необработанных на основе их поштучного учета и маркировки средствами таможенной идентификации. Эксперимент организован Сибирским таможенным управлением и проводился более года. В нем участвовали Забайкальская, Иркутская и Наушкинская таможни, а также около 50 крупных лесоэкспортеров.

Решением выездного заседания коллегии ФТС РФ от 25.05.07 результаты эксперимента признаны удовлетворительными. Принято решение о распространении системы таможенного контроля лесоматериалов необработанных на основе их поштучного учета в других федеральных округах России.

Начальник ЭКС – филиала ЦЭКТУ
г. Иркутска
полковник таможенной службы, к.т.н., доцент

С.В. Трунов

002805