Міністерство освіти і науки України
Запорізький державний університет

До захисту допущений
Зав. кафедрою
ММ та ІТ

(підпис)

Борю Сергій Юрійович
(прізвище, ім'я, по батькові)

(дата)

ДИПЛОМНА РОБОТА

НА ТЕМУ: **Построение тематического классификатора для Web-портала научных публикаций в области компьютерных наук**

Виконав Маляр Євгеній Миколайович

ст. групи 8228-2 Є. М. Маляр
(шифр) (підпис і дата) (ім'я, по батькові, прізвище)

Керівник доцент С. В. Курапов
(посада) (підпис і дата) (ім'я, по батькові, прізвище)

Консультант доцент В. А. Єрмолаєв
(посада) (підпис і дата) (ім'я, по батькові, прізвище)

Нормоконтролер Н. Б. Александрова
(підпис) (ім'я, по батькові, прізвище)

Запоріжжя
2002

ЗАДАНИЕ

РЕФЕРАТ

Дипломная работа: 69 с., 5 рис., 10 источников.

Объект исследования – ACM классификация в области компьютерных наук.

Цель работы - разработка математической модели и алгоритмов для построения классификации научных публикаций в области компьютерных наук.

Задачами работы: визуализация классификации научных публикаций ACM в области компьютерных наук и интеллектуальный поиск с использованием данной классификации.

Методы исследования - теоретические методы формальных языков и грамматик; представление языков с помощью грамматик.

Данная дипломная работа посвящена актуальной теме исследований в области математического моделирования сложных интеллектуальной системы. Данная система представляет собой классификатор научных публикаций в области компьютерных наук и возможность интеллектуального поиска.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

АСМ – Assosiation for Computer Machinery, Ассоциация Компьютерных Наук

ББК – Большой библиотечный каталог

УДК – Универсальный десятичный каталог

ДИПС – Документальная информационная поисковая система

ИПС – Информационная поисковая система

ЕЯ – Естественный язык

СУБД – Система управления базами данных

ИПЯ – Информационно поисковый язык

БНФ – нормальная форма Бэкуса

WWW – World Wide Web – Глобальная информационно-поисковая система

Web – портал – комплекс программ для визуализации, навигации и поиска документов, размещенных в WWW

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Проблема поиска информации в наше время становится все более сложной и трудноразрешимой. Особенно сложно искать нужную информацию в сети Internet. На сегодняшний момент существует множество электронных изданий, которые предоставляют публикации в области компьютерных наук в электронном виде. И наиболее сложная проблема заключается в поиске нужной публикации, что часто бывает трудноразрешимой проблемой.

Задача тематической классификации научных публикаций все чаще затрагивается компаниями, занимающимися электронными изданиями. Действительно, актуальность этой проблемы очень велика. Создание универсальной системы тематической классификации в области компьютерных наук, является основополагающим фактором в процессе развития, изучения и использования таких публикаций. Современный уровень развития науки и техники позволяет решить эту проблему наиболее эффективным образом, а именно - с использованием передовых компьютерных и информационных технологий.

В наше время существует множество классификаций электронных публикаций: NEC ResearchIndex [1], DBLP[2], AgentLink [3]. Каждая из них имеет свой собственный тематический каталог. Однако, наличие нескольких пересекающихся классификаторов приводит к проблеме универсальности, т.е. отсутствие единой всеобщей классификации электронных ресурсов в области компьютерных наук. Именно аспект отсутствия универсальности создает проблему поиска нужной информации. Универсальная классификация электронных публикаций должна приниматься всеми и быть очень детализированной.

В то же время, стандарты классификации бумажных изданий, такие как Большой библиотечный каталог (ББК), ISBN, в силу своей универсальности, не являются детализированными. Более того, такие классификации не поддаются

быстрому обновлению. Кроме того, существует большое количество тематических каталогов, таких как каталоги товаров и услуг, каталоги фирм по видам деятельности, и т.п., для которых также требуются средства визуализации, навигации и поиска.

Актуальность работы: на данный момент существует множество тематических каталогов, разработанных независимо, на базе разных языков представления. В то же время для работы с такими каталогами нужен стандартный набор средств визуализации, навигации, редактирования и поиска.

В данный момент уже разработаны разнообразные подходы к визуализации, навигации и поиску в документальных информационно-поисковых системах (ДИПС). Однако любая ДИПС ориентирована на представление информации о предметной области в некотором классификационном языке, тем самым, область применения этой ДИПС ограничивается конкретным классификационным языком.

Новизна данной работы состоит в том, что была предпринята попытка разработать механизмы обработки произвольных тематических каталогов унифицированным образом, на базе методов визуализации, навигации, редактирования и поиска, используемых в документальных ИПС.

В данной работе используется классификация электронных публикаций ACM, которая в наибольшей степени, по отношению к другим, тематическим каталогам, удовлетворяет требованиям общности и детализированности для классификации публикаций в области компьютерных наук.

Задачами дипломной работы являются: разработка алгоритмов и программ для отображения ACM классификации в виде древовидной структуры и интеллектуального поиска с использованием классификации.

1. ПРОБЛЕМА КЛАССИФИКАЦИИ ЭЛЕКТРОННЫХ ПУБЛИКАЦИЙ

Классические модели и методы в теории БД изначально ориентировались на организацию хранения и обработки детально структурированных данных. Чаще всего эти данные представляли собой числовые значения, описывающие те или иные характеристики информационных объектов.

Однако на практике оказалось, что чаще информация представлена не в виде структурированных массивов данных, а в виде простых текстовых документов. Вследствие этого документальные БД (иногда их еще называют полнотекстовыми) сразу выделялись в особый тип баз данных. Исторически сложилось так, что за системами, ориентированными на работу с текстовыми документами, укоренился термин информационно-поисковые системы (ИПС). Хотя, если быть точнее, их следует называть документальными ИПС (ДИПС), поскольку традиционные СУБД также являются ИПС, только фактографическими (ФИПС).

В отличие от традиционных БД, ориентированных на полное и точное представление данных достаточно простой смысловой структуры, документальные БД ориентированы на частичное, приближенное представление данных, имеющих значительно более сложную смысловую структуру, представленных на входе в форме текста.

Основной функцией любой ДИПС является информационное обеспечение потребителей на основе выдачи ответов на их запросы. Осуществление выдачи системой требуемых данных реализуется с помощь главной операции ДИПС - проведения информационного поиска. Информационный поиск является процедурой отыскания документов, содержащих ответ на заданные потребителем вопросы (определения основных понятий приведены из работы [4]).

Заметим, что в отличие от ФИПС, которые в ответ на запрос потребителя осуществляют выдачу конкретных сведений (фактов), ДИПС в результате проведения информационного поиска предоставляют потребителю совокупность документов, смысловое содержание которых соответствует его запросу.

Информационный поиск в системе проводится на основе поступившего от потребителя запроса на отыскание необходимой ему информации. Потребность человека в определенной информации в процессе его практической деятельности носит название информационной потребности. Под действием получаемой информации информационная потребность людей постоянно изменяется и трансформируется. Вследствие этого ее невозможно однозначно выразить и описать. Однако информационная потребность может быть представлена в виде некоторой последовательности ее частных значений в фиксированные моменты времени. Такое частное значение информационной потребности потребителя в определенные моменты времени, выраженное на естественном языке (ЕЯ), и представляет собой информационный запрос, с которым пользователь обращается к системе.

Однако запрос может быть неправильно сформулирован потребителем и не отражать его истинной информационной потребности в момент обращения к системе. Таким образом, при проведении информационного поиска в системе фактически рассматривается не информационная потребность пользователя, а только информационный запрос, в ответ на который и выдаются те или иные документы системы. Следовательно, реакцию системы необходимо рассматривать не только по отношению к информационной потребности, но по отношению к информационному запросу.

Для выражения данных отношений в теории ДИПС введены два фундаментальных понятия: пертинентность и релевантность. Под *пертинентностью* понимается соответствие смыслового содержания документа информационной потребности потребителя. Документы, содержание которых удовлетворяет информационной потребности, называют пертинентными. *Релевантность* представляет собой соответствие содержания документа информационному запросу в том виде, в каком он сформулирован, а документы, содержание которых отвечает запросу потребителя, носят название релевантных.

Автоматизация процесса информационного поиска потребовала формализации представления основного смыслового содержания информационного запроса и документов в виде соответственно поискового предписания (ПП) и поисковых образов

документов (ПОД). Для записи ПП и ПОД применяются специальные языки, называемые информационно-поисковыми (или просто информационными).

В процессе проведения информационного поиска в ДИПС определяется степень соответствия содержания документов и запроса пользователя путем сопоставления ПОД с ПП. А на основе такого сопоставления принимается решение о выдаче документа (он признается релевантным) или его невыдаче (он считается нерелевантным).

Решение о выдаче или невыдаче документа в ответ на запрос принимается на основе некоторого набора правил, по которому данной ДИПС определяется степень смысловой близости между ПОД и ПП. Такой набор правил получил название *критерия смыслового соответствия* (КСС). Критерий может задаваться явно или неявно. На самом деле КСС базируется не на ранее введенном понятии релевантности, а на понятии *формальной релевантности* - соответствии содержания ПОД и ПП. *Фактическая релевантность,* понимаемая как смысловое соответствие содержания документа информационному запросу, может быть установлена только человеком в процессе осмысления содержания документа и запроса.

В состав типичной ДИПС входят, как правило, четыре основные подсистемы (рис. 1):

1. Подсистема ввода и регистрации.
2. Подсистема обработки.
3. Подсистема хранения.
4. Подсистема поиска.

Запрос

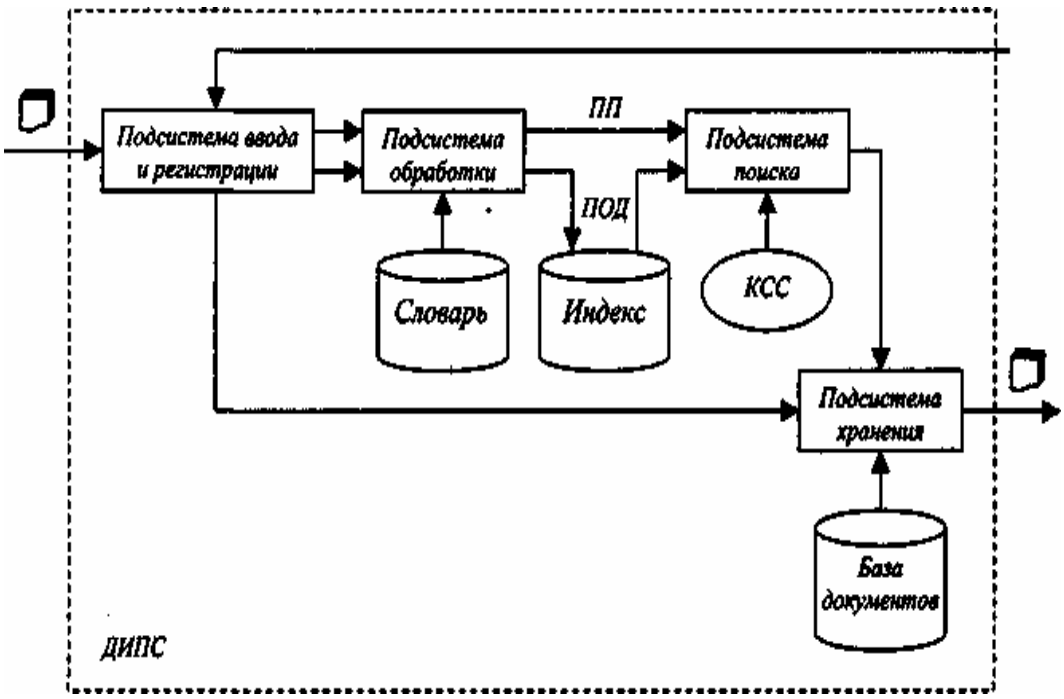

Рис. 1. Общая функциональная структура ДИПС

Текстовые документы, поступающие на вход системы, могут быть представлены как в бумажном, так и в электронном виде (в одном из многочисленных форматов). Поэтому подсистема ввода и регистрации решает следующие основные задачи:

- создание электронных копий бумажных документов (например, сканирование с последующим распознаванием текста или ввод с клавиатуры);
- обеспечение подключения к каналам доставки электронных документов;
- распознавание, а при необходимости и преобразование формата электронных документов;
- присвоение электронным документам уникальных идентификаторов (регистрация), а также ведение таблицы синхронизации имен (при необходимости сохранения прежних имен).

Все поступающие документы без внесения в них каких-либо изменений направляются в подсистему хранения для сохранения в базе документов. База документов может представлять собой простую совокупность файлов, распределенную по каталогам жесткого диска. Однако такой тип представления базы документов характеризуется двумя недостатками:

- неэффективным использованием дискового пространства;
- низкой скоростью доступа при большом количестве файлов.

Поэтому для хранения документов применяют средства сжатия и быстрого поиска информации. В этом случае подсистема хранения представляет собой совокупность стандартных или специализированных средств архивации, СУБД и т.п., обеспечивающих возможность доступа к данным по предъявляемому идентификатору.

Далее документы поступают на вход подсистемы обработки, задачей которой является формирование для каждого документа ПОД, в который заносится информация, необходимая для последующего поиска документа.

ПОД сохраняются в индексе. Логически индекс представляет собой таблицу, строки которой соответствуют документам, а столбцы — информационным признакам, на основе которых строится ПОД. В ячейках таблицы могут храниться либо 1, либо О-в зависимости от наличия или отсутствия данного признака в данном документе.

Очевидно, что такая таблица будет сильно разреженной, и хранить все значения не имеет смысла. Поэтому на практике используют свертку таблицы по строкам или столбцам. Такую форму хранения называют прямой или инверсной соответственно. Поскольку при свертке таблицы структура индекса усложняется, для его поддержания могут использоваться средства СУБД.

При поступлении на вход системы запроса пользователя он преобразуется в ПП и передается в подсистему поиска, задачей которой является отыскание в индексе ПОД, удовлетворяющих ПП с точки зрения КСС. Идентификаторы релевантных документов подаются с выхода подсистемы поиска на вход подсистемы хранения, которая осуществляет выдачу пользователю самих релевантных документов.

Как известно, естественный язык (ЕЯ) является универсальной знаковой системой, служащей для обмена информацией между людьми. Поскольку документы, поступающие на вход ДИПС, записаны на ЕЯ, справедливо было бы задаться вопросом , а нельзя ли использовать ЕЯ в качестве основного средства преставления информации во время всего цикла функционирования ДИПС? Ответ будет положительным, если речь идет о тех ИПС, в которых соответствие между запросом и документом устанавливает человек. Однако в современных ЭВМ эта операция

выполняется компьютером, что практически исключает применение ЕЯ в качестве основного средства представления информации. Это объясняется существенными недостатками ЕЯ с точки зрения машинной технологии обработки информации, основные из которых рассмотрены ниже.

Многообразие средств передачи смысла. Несмотря на то, что основным средством передачи смысла сообщения является лексика естественного языка, и сообщениях на ЕЯ функцию передачи смысла выполняет и ряд других элементов:

- контекст;
- парадигматические отношения между словами;
- текстуальные отношения между словами;
- ссылки на слова (словосочетания, фразы и т.д.), ранее упоминавшиеся в тексте сообщения.

Семантическая неоднозначность. Сообщения, записанные на естественном зыке, могут быть семантически неоднозначными. Семантическая неоднозначность возникает в основном из-за синонимии и многозначности слов естественного языка.

Синонимия представляет собой тождественность или близость по значению слов, выражающих одно и то же понятие, которые отличаются одно от другого ни оттенками значений, или стилистической окраской, или одновременно обоими названными признаками. Синонимами естественного языка являются как отдельные слова, так и словосочетания.

Многозначность характеризует возможность неоднозначного понимания ее смысла отдельных слов естественного языка. Многозначность слов представлена двумя разновидностями - полисемией и омонимией. Полисемия - это совпадение названий различных предметов, имеющих между собой какие-либо общие свойства или признаки. К типичным общим свойствам, служащим базой полисемии, следует отнести сходство предметов, их смежность (пространственную, временную и т.д.), а также одинаковое функциональное

назначение. Примерами полисемии являются: "команда" (воинское подразделение) - "команда" (экипаж судна) - "команда" (спортивная).

Омонимия - это совпадение названий различных предметов, не имеющих между собой каких-либо общих свойств. Например: "лук" (оружие) - "лук" (растение); "ключ" (родник) - "ключ" (дверной). Омонимичные слова, совпадающие между собой как по написанию, так и по звучанию, следует отличать от омографов - слов, обозначающих различные предметы, одинаковые по написанию, но разные по звучанию, например: «замок»(дверной) – «замок» (дворец).

1.1 Необходимость классификации электронных публикаций

Проблема классификации электронных публикаций являлась и является острой и нерешенной проблемой. Сложность такой проблемы заключается в том, чтобы создать универсальный классификатор публикаций, который определенным образом классифицировал бы публикации, записывая их в тот или иной пункт этого каталога. Т.е. мы приходим к идее создания универсального классификатора и построения универсального тематического каталога. Но такая задача оказывается тоже трудноразрешимой. Универсальность каталога и классификатора означают то, чтобы всеми принимался данный каталог и данный классификатор, что является весьма сложной задачей. Рассматривая более глубоко данную проблему нужно сказать, что именно степень детализации и уточнения влияет на универсальность такого каталога.

1.2 Подходы к классификации публикаций

Невозможность использования ЕЯ в качестве основного средства представления информации в ДИПС приводит к необходимости применения искусственных языковых средств.

Информационно-поисковым языком (ИПЯ) называется специализированный искусственный язык, предназначенный для описания основного смыслового содержания поступающих в систему сообщений, с целью обеспечения возможности последующего их поиска [4].

ИПЯ создается на базе ЕЯ, однако отличается от него компактностью, наличием четких грамматических правил и отсутствием семантической неоднозначности.

ИПЯ принято разбивать на два основных типа:

- классификационные языки
- дескрипторные языки

Принципиальная разница между данными типами языков заключена в процедуре построения предложений (фраз) языка. В ряде языков в их лексический состав наряду со словами, выражающими простые понятия, заранее включены также словосочетания и фразы, выражающие сложные понятия. Для записи смыслового содержания сообщений в таких ИПЯ используются только отдельные элементы из этого набора, в том числе и готовые сложные понятия. Фактически построение сложных синтаксических конструкций заменяется выбором соответствующего сложного понятия (в виде словосочетания или фразы) из готового набора. Например:

Политика.Внутренняя.Федеральная
Политика.Внутренняя.Региональная

Политика.Внешняя...

Таким образом, с помощью таких языков производится классификация соотношений, т.е. отнесение их к классам, обозначенным лексическими единицами ИПЯ. Поэтому такие языки получили название классификационных. Частным случаем классификационного ИПЯ является рубрикатор, лексическими единицами которого являются названия

тематических рубрик. В целом к рубрикаторам некоторой предметной области понимается ориентированный граф, состоящий из независимых деревьев. Листья деревьев будем называть рубриками - объектами, инкапсулирующие знания о конкретных фрагментах данной предметной области. Все нелистовые вершины являются классификационными обобщениями листовых вершин и используется лишь про ведении информационного поиска.

Обычно рубрикатор формируется группой экспертов, на основании их знаний о предметной области с учетом информационных потребностей пользователей. Следует подчеркнуть одну особенность классификационных языков. Поскольку сложные понятия задаются заранее, до начала процедуры записи сообщений с помощью ИПЯ, образующие их слова также заранее связаны (скоординированы) определенными связями. Поэтому такие языки носят название предкоординируемых.

Другой тип языков составляют дескрипторные ИПЯ, в которых ЛЕ заранее не связаны никакими текстуальными отношениями. Сложные синтаксические конструкции - предложения или фразы - создаются в этих языках путем объединения (координации) ЛЕ во время процедуры представления смыслового содержания документов системы. Готовых предложений или фраз в таких языках нет, поэтому отсутствуют ограничения на составление сложных понятий. Фактически из небольшого числа ЛЕ данные языки позволяют строить предложения, выражающие практически любой смысл. Такие ИПЯ носят также название посткоординируемых, поскольку координация между словами предложения возникает во время его записи.

Различают дескрипторные ИПЯ с грамматикой и без грамматики. Первые характеризуются наличием ряда жестких правил формирования синтаксических конструкций. Например, при использовании дескрипторного ИПЯ с позиционной грамматикой, в котором при описании действий принято на первом месте записывать наименование действия, далее субъекта, а затем объекта этого действия, фраза: "Иванов владеет автомобилем" может выглядеть так: "владеть Иванов автомобиль". В дескрипторных ИПЯ без грамматики такие правила

отсутствуют, и порядок следования ЛЕ в ПОД или ПП не играет роли. Т.е. приведенный выше пример может быть одинаково представлен последовательностями "владеть Иванов автомобиль", "Иванов владеть автомобиль" и т.п. Кроме того, различают дескрипторные ИПЯ с контролируемой и со свободной лексикой. Лексический состав первых строго ограничен и зафиксирован в словаре ИПЯ, в то время как на лексический состав вторых не налагается никаких ограничений, и он может постоянно пополняться за счет включения новых ЛЕ.

1.3Известные классификации публикаций

На сегодняшний момент существует множество классификаций публикаций, например классификаций NEC ResearchIndex [1], DBLP[2], AgentLink [3]. Каждая из этих классификаций имеет свой тематический каталог, который в той или иной мере является детализированным и полным. В зависимости от этого классификация считается более или менее универсальной.

Особое внимание следует обратить на библиотечные каталоги. Всем издаваемым печатным изданиям (книги, сборники статей, справочники и т.д.) ставится в соответствие некоторый код, определяющий принадлежность содержимого печатного издания к некоторому разделу каталога. Примерами таких каталогов являются, очевидно, Большой Библиотечный Каталог (ББК), ISBN, Универсальный Десятичный Каталог (УДК).

Однако, использование каталогов для классификации электронных публикаций затруднено по таким причинам:

А) Каталоги ББК и ISBN являются каталогами общего назначения, т.е. это универсальные каталоги, которые используются для классификации всевозможной литературы. Каталоги ББК и ISBN являются универсальными каталогами и не имеют высокой степени детализации, т.к. слишком сложно разработать каталог для классификации всевозможной литературы с высокой степенью детализации ее разделов.

Б) Каталоги ББК и ISBN имеют множество ссылок из одного раздела на другой, тем самым заменяя высокую степень детализации классификации ссылками на другие разделы.

В данной работе в качестве тематического каталога для публикаций в области компьютерных наук используется тематический каталог, разработанный Ассоциацией Компьютерных Наук, Association for Computer Machinery (ACM). ACM – международная организация, объединяющая исследовательские группы из учебных заведений, промышленности, с целью обмена знаниями, публикаций результатов интересных исследований, проведения конференций и т.д., и обладающая большим влиянием в данной области.

Разработанная в ACM классификация публикаций в области компьютерных наук является одной из самых подробных контролируемых классификаций в этой области. Дерево каталога ACM классификации на данный момент насчитывает 1286 разделов. Дерево каталога ACM классификации представлено на рисунке 1.

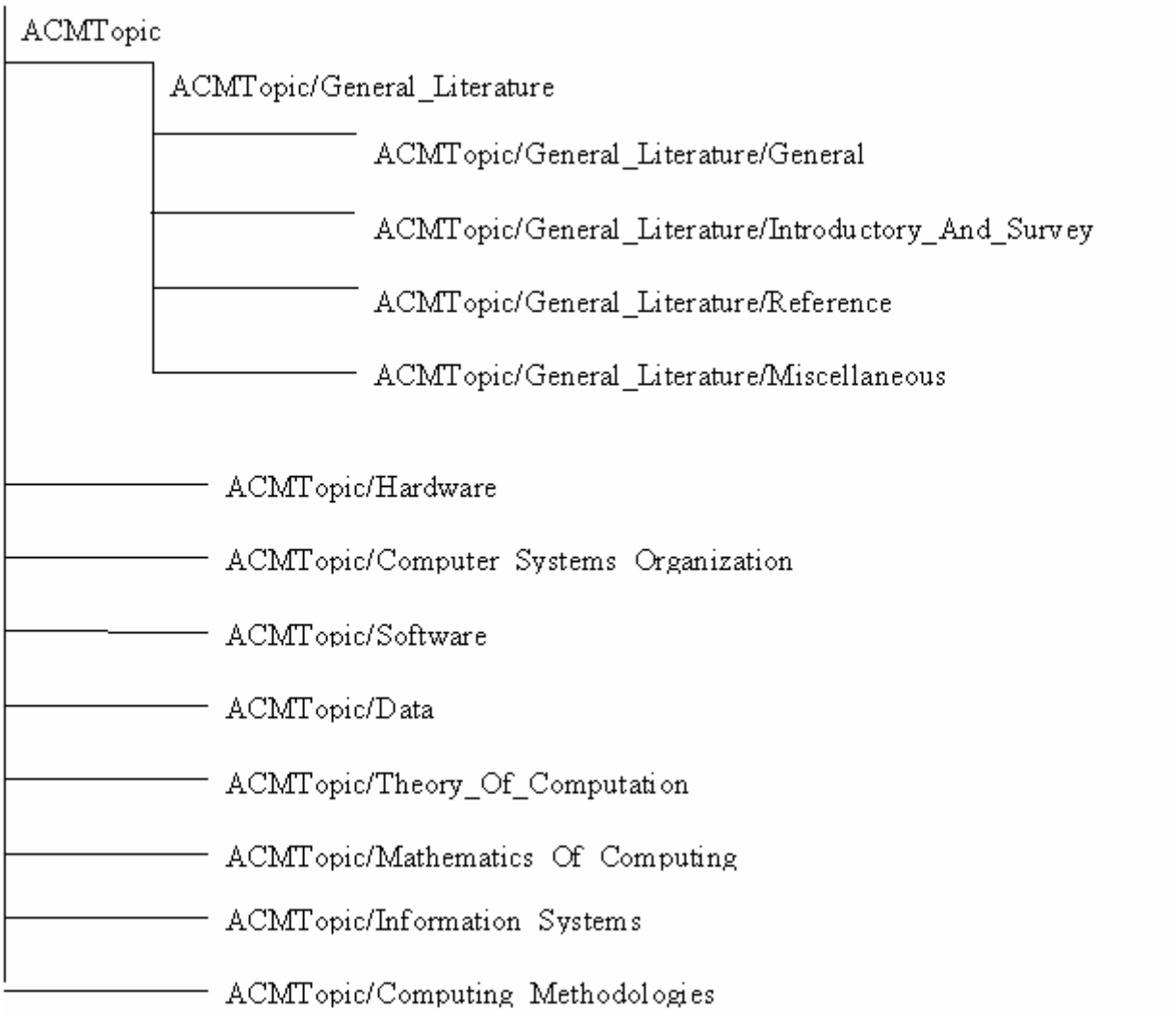

Рисунок 1 – Пример ACM классификации

1.4 Модели поиска текстовой информации

Модель поиска текстовой информации характеризуется четырьмя параметрами:

- представлением документов и запросов;
- критерием смыслового соответствия;
- методами ранжирования результатов запроса;
- механизмами обратной связи, обеспечивающими оценку релевантности пользователем.

Рассмотрим наиболее распространенные модели поиска с позиции первых трех параметров.

Булева модель представляет документы с помощью набора терминов, присутствующих в индексе, каждый из которых рассматривается как булева переменная. При наличии термина в документе соответствующая переменная принимает значение True. Присваивание терминам весовых коэффициентов не допускается. Запросы формулируются как произвольные булевы выражения, связывающие термины с помощью стандартных логических операций: AND, OR или NOT. Мерой соответствия запроса документу служит значение статуса выборки (RSV, retrieval status value). В булевой модели RSV равно либо 1, если для данного документа вычисление выражения запроса дает True, либо 0 в противном случае. Все документы с RSV = 1 считаются релевантными запросу.

Такая модель проста в реализации и применяется во многих коммерческих системах. Она позволяет пользователям вводить в свои запросы произвольные сложные выражения. Однако эффективность поиска обычно невысока. К тому же, ранжировать результаты невозможно, так как все найденные документы имеют одинаковые RSV, а терминам нельзя присвоить весовые коэффициенты. Нередко результаты выглядят противоестественно. Например, если пользователь указал в запросе десять терминов, связанных логической операцией AND, документ, содержащий девять таких терминов, в выборку не попадет. Для повышения эффективности поиска в ИПС часто применяется обратная связь с пользователем. Как правило, система просит пользователя указать релевантность или нерелевантность нескольких документов, включенных в начало списка вывода. Поскольку результаты не ранжируются, выбор документов для подобной экспертной оценки релевантности затруднен.

Модель нечетких множеств [5] основывается на теории нечетких множеств, допускающей (в отличие от обычной теории множеств) частичную принадлежность элемента тому или иному множеству. Здесь логические операции переопределены таким образом, чтобы учесть возможность неполной принадлежности множеству, а обработка запросов пользователя выполняется

аналогично булевой модели. Тем не менее ИПС на основе подобной модели оказывается практически столь же не способной классифицировать полученные результаты, что и системы, базирующиеся на булевой модели.

Строгая булева модель и модель, использующая методы теории нечетких множеств, требуют меньших объемов вычислений (при индексировании и оценке соответствия документов запросу), чем другие модели. Они менее сложны алгоритмически и предъявляют не очень жесткие требования к другим ресурсам, таким как дисковое пространство для хранения представлений документов.

Пространственно-векторная модель [5] основана на предположении, что совокупность документов можно представить набором векторов в пространстве, определяемом базисом, из n нормализованных векторов терминов. Значение первого компонента вектора представляющего документ отражает вес термина в нем. Запрос пользователя также представляется n-мерным вектором. Показатель RSV, определяющий соответствие документа запросу, задается скалярным произведением векторов запроса и документа. Чем больше RSV, тем выше релевантность документа запросу.

Достоинство подобной модели в ее простоте. Она позволяет легко реализовать обратную связь для оценки релевантности пользователем. В то же время приходится жертвовать выразительностью спецификации запроса, присущей булевой модели.

Вероятностные модели [5]. В пространственно-векторной модели подразумевается, что векторы терминов, ортогональны и существующие взаимосвязи между терминами не должны приниматься во внимание. Кроме того, в такой модели не специфицируется степень соответствия "запрос - документ" и она оценивается достаточно произвольно. Вероятностная модель учитывает все взаимозависимости и связи терминов, а также определяет такие основные параметры, как веса терминов запросов и форма соответствия "запрос - документ". Данная модель базируется на двух главных параметрах: Pr(rel) и Pr(nonrel), на вероятности релевантности и нерелевантности документа запросу

пользователя, которые вычисляются на основе вероятностных весовых коэффициентов терминов и фактического присутствия терминов в документе. Подразумевается, что релевантность является бинарным свойством, и поэтому Pr(rel) = 1 Pr(nonrel)=0. Кроме того, в этой модели применяются два стоимостных параметра: al и a2. Они характеризуют соответственно потери, связанные с включением в результат нерелевантного документа и пропуском релевантного документа.

Данная модель требует определения вероятностей вхождения термина в побайтные и нерелевантные части совокупности документов, оценить которые довольно сложно. Между тем она выполняет важную функцию, объясняя процесс поиска и предлагая теоретическое обоснование методов, применяется ранее эмпирически (например, введение некоторых систем определения весовых коэффициентов терминов).

1.5 Выводы и постановка задачи

В данном разделе сделано введение в проблему использования тематических каталогов для классификации публикаций. Рассмотрены примеры классификаций электронных изданий. Также рассмотрены задачи построения тематического каталога публикаций в области компьютерных наук и функции интеллектуального поиска.

К основным задачам, решаемым с использованием тематических каталогов публикаций, являются:

- использование тематического каталога для поиска подходящих публикаций;
- использование тематического каталога для классификации новых публикаций;

Решение таких задач требует таких практических подходов:

- визуализация тематического каталога в виде иерархии концепций, для навигации, анализа и редактирования;

- экспорт каталогов из одного языкового представления в другое;
- поиск публикаций, удовлетворяющих заданным описаниям разделов тематического каталога;

Разработка единого, удовлетворяющего всех, тематического каталога, очевидно, сопряжена с большими трудностями. Основная причина затруднений вызвана несоответствиями в представлениях разных групп ученых, разных научных школ, в трактовке тех или иных понятий и их определений. Поэтому, требование существования одного глобального каталога, представляющего тематику некоторой области, заменяется требованием существования многих частично пересекающихся каталогов, удовлетворяющих отдельные группы экспертов. В то же время, разработка единых каталогов терминов в некоторых небольших областях, приветствуется, и удачные реализации таких каталогов действительно используются большим количеством экспертов.

Задача поиска публикаций, удовлетворяющих заданным описаниям разделов, является логическим продолжением задачи представления тематического каталога. Она включает в себя использование уже построенного дерева концепций. В задаче поиска дерево концепций используется для реализации функции интеллектуального поиска. Функция интеллектуального поиска заключается в решении задачи полного перебора. В критерии поиска вводятся ключевые слова по которым пользователь хочет вести поиск и выбираются разделы, в которых, по мнению пользователя должна содержаться нужная информация. Тем самым область поиска сужается в десятки раз и уменьшается вероятность нахождения нерелеваных ссылок. Важным аспектом является то, что при поиске не производится полный перебор, а выполняется частичный.

ПОСТАНОВКА ЗАДАЧИ:

Дан некоторый классификационный язык, в терминах которого рубрицируются публикации в области компьютерных наук наук [4]. На основании этого языка создан тематический каталог для рубрицирования публикаций.

1. Разработать алгоритмы и программы для визуализации тематического каталога со средствами навигации и редактирования в виде иерархии разделов.
2. Разработать алгоритмы и программы поиска аннотированных публикаций, удовлетворяющих описаниям некоторых разделов каталога.

Для реализации задачи 1 предполагается использовать методы анализа формальных языков с помощью грамматик и порождающих алгебр, элементы общей алгебры.

Для реализации задачи 2 предполагается использовать булеву модель представления документов и запросов в ДИПС.

2. ФОРМАЛЬНЫЕ ПОДХОДЫ К ВИЗУАЛИЗАЦИИ ТЕМАТИЧЕСКОГО КАТАЛОГА И К ЗАДАЧЕ ИНФОРМАЦИОННОГО ПОИСКА

В данной главе будут рассмотрены формальные подходы для решения задач, поставленных в п.1.5. В п.2.1 рассматривается формальный метод для решения задачи визуализации тематического каталога, с использованием аппарата общей алгебры. В п.2.2 формулируется булева модель поиска по тематическому каталогу.

2.1 Формальный подход к решению задачи визуализации тематического каталога

В настоящее время большое распространение получил язык eXtensible Markup Language (XML) [6], как язык разметки текстов, позволяющий расширение основных разметочных конструкций (тэгов), конструкциями, соответствующие понятиям, присущим только конкретной области. Такие дополнительные тэги определяются с помощью специальной конструкции XML - пространства имен (XML namespace), и используются наравне со стандартными тэгами разметки документов.

Существуют общепринятые расширения языка XML для разметки текстов на основании знаний о предметной области. В качестве таких языков используются специализированные языки, такие как RDF[7], XML[6], которые предназначены для разметки текстов на основании знаний о предметной области, с большей или меньшей степенью детализации. Каждый из перечисленных языков реализуется как надстройка над языком XML, и определяется с помощью соответствующего пространства имен (<rdf>, <rdfs>, <daml>, <oil>).

Язык классификации публикаций ACM также является надстройкой над языком XML, т.к. использует синтаксис XML, но расширяет множество разметочных конструкций конструкциями классификационного языка. Более того, поскольку для рубрицирования текстов необходимо использовать знания

о предметной области, язык классификации публикаций использует также пространства имен языков RDF, RDFS, DAML.

Таким образом, в терминах документальных информационно-поисковых систем, для классификации документов публикаций в области компьютерных наук используется классификационный язык. Его особенность состоит в том, что его единственной конструкцией является название раздела – рубрики, к которой может быть отнесен текст публикации.

С другой стороны, можно рассматривать представление данного классификационного языка как текст на другом, вспомогательном языке, в нашем случае, на расширении языка XML для классификации публикаций.

Язык классификации публикаций ACM представлен в виде пространства имен с идентификатором acm (см. Приложение А).

Для построения корректного перевода классификационного языка ACM в язык представления древовидных структур необходимо использовать средства формального представления и анализа языков, а значит аппарат грамматик и общей алгебры.

2.1.1 Определение грамматики, порождающей алгебры, перевода для формальных языков

Любой формальный язык представляет собой множество цепочек в некотором конечном алфавите. В лингвистике вместо термина «алфавит» употребляется термин «словарь», так как элементы, из которых он составлен, представляют собой слова, или, точнее, словоформы. В то же время цепочка над словарем рассматривается как словосочетание или предложение. К формальным языкам относятся, в частности, искусственные языки для общения человека с машиной — языки программирования. Для задания описания формального языка необходимо, во-первых, указать алфавит, т. е. совокупность объектов, называемых символами (или буквами), каждый из которых можно воспроизводить в неограниченном количестве экземпляров (подобно обычным печатным буквам или цифрам), и, во-вторых, задать формальную грамматику

языка, т. е. перечислить правила, по которым из символов строятся их последовательности, принадлежащие определяемому языку,— правильные цепочки.

Заметим, что каждый символ алфавита рассматривается как неделимый в том смысле, что при построении цепочек никогда не используются его графические элементы (части символа) и всякая последовательность символов однозначно представляет некоторую цепочку. Практически это требование достигается, например, путем установления пробела (промежутка стандартной длины) между символами, который превышает длину любого из промежутков, встречающихся внутри символов алфавита.

Правила формальной грамматики можно рассматривать как продукции (правила вывода) — элементарные операции, которые, будучи применены в определенной последовательности к исходной цепочке (аксиоме), порождают лишь правильные цепочки. Сама последовательность правил, использованных в процессе порождения некоторой цепочки, является ее выводом. Определенный таким образом язык представляет собой формальную систему. Известными примерами формальных систем служат логические исчисления (исчисление высказываний, исчисление предикатов), которые подробно изучаются в соответствующих разделах математической логики.

По способу задания правильных цепочек формальные грамматики разделяются на порождающие и распознающие. К порождающим относятся грамматики, по которым можно построить любую правильную цепочку с указанием ее структуры и нельзя построить ни одной неправильной цепочки. Распознающая грамматика — это грамматика, позволяющая установить, правильна ли произвольно выбранная цепочка и, если она правильна, выяснить ее строение.

Формальные грамматики широко применяются в лингвистике и программировании в связи с изучением естественных языков и языков программирования.

В приложении Б рассмотрен формальный аппарат для построения

перевода, сохраняющего смысл языковых конструкций, из одного формального языка в другой.

2.1.2 Построение перевода классификационного языка в язык представления древовидных структур

Представим грамматику входного языка классификации публикаций ACM, L_{acm} в нормальной форме Бэкуса (БНФ).

<*acmClassification*>::=<*acmClassificationName*><*acmTopic*>+

<*acmClassificationName*>::=<*acmLiteral*>

<*acmTopic*> ::=”<**acm:Topic rdf:ID=””**<*acmTopicPath*>**””>”**

”**<rdfs:label>”**<*acmTopicName*>**”</rdfs:label>”**

[<*acmComment*>]

<*acmSeeAlso*>+

<*acmSubTopit*>+

[<*acmSuperTopic*>+] **“/acm:Topic”**

<*acmTopicPath*>::=<*acmTopicName*><*acmDashedTopicList*>

<*acmDashedTopicList*> ::=”/”<*acmTopicName*><*acmDashedTopicList* >

<*acmComment*> ::=”**<rdfs:comment>”**<*string*>**”</rdfs:comment>”**

<*acmSeeAlso*> ::=**“<rdfs:seeAlso rdfs:resource=**

““#”<*acmTopicPath*>””**/>”**

<*acmSubTopic*> ::=**“<acm:SubTopic rdfs:resource=””#”**

<*acmTopicPath*>””**/>”+**

<*acmSuperTopic*>::=**“<acm:SuperTopic**

rdfs:resource=””#”<*acmTopicPath*>**””/>”**

<*acmTopicName*>::=<*acmLiteral*>

<*acmLiteral*> ::=<*string*>

Для облегчения анализа данной грамматики преобразуем нотации БНФ < *nonTerminalList* >+ и
{*nonTerminalList*} в отдельные нетерминальные символы типа
<*nonTerminalList*>. Получим такие дополнительные нетерминалы:

<*acmTopicList*> ::=<*acmTopic*><*acmTopicList*>
<*acmSeeAlsoList*> ::=<*acmSeeAlso*><*acmSeeAlsoList*>
<*acmSubTopicList*> ::=<*acmSubTopic*><*acmSubTopicList*>
<*acmSuperTopicList*> ::=<*acmSuperTopic*><*acmSuperTopicList*>

Формально, порождающая грамматика языка L_{acm} имеет вид:

$G_{acm} = (A_{acm}, V_{n\_acm}, \sigma_{acm}, P_{acm})$,

где множество терминалов целевого языка

A_{acm} = {**<acm:Topic rdf:ID=", ">,</acm:Topic>,**
<rdfs:label>,</rdfs:label>, /, <rdfs:comment>, </rdfs:comment>,
<rdfs:SeeAlso rdfs:resource=#">, "/>,<acm:SubTopic rdfs:resource=#",
<acm:SubTopic rdfs:resource=#"};

множество нетерминалов целевого языка

V_{n_acm} = {<*acmClassification*>, <*acmClassificationName*>,
<*acmTopicPath*>, <*acmComment*>, <*acmSeeAlso*>, <*acmSubTopic*>,
<*acmSuperTopic*>, <*acmLiteral*>, <*string*>, <*acmTopicList*>,
<*acmSeeAlsoList*>, <*acmSubTopicList*>, <*acmSuperTopicList*>,
<*acmDashedTopicList*>};

аксиома целевого языка

σ_{acm} =<*classification*>

схема грамматики, определяющая множество правил порождения, представлена выше.

Порождающая алгебра $MA_{acm} =< M(G_{acm}), \Omega_{acm} >$ будет контекстно-свободной, т.к. в левых частях правил из P стоят только нетерминальные символы, а во-вторых многоосновной.

$M(G_{acm}) = A_{string}$

Множество операций Ω_{acm}.

$\omega_{acmClassification} : A_{acmClassificationName} \times A_{acmTopicList} \to A_{acmClassification}$

$\omega_{acmClassificationName} : A_{acmLiteral} \to A_{acmClassificationName}$

$\omega_{acmTopicList} : A_{acmTopic} \times A_{acmTopicList} \to A_{acmTopicList}$

$\omega_{acmTopic} : A_{acmTopicName} \times A_{acmTopicPath} \times A_{acmComment} \times A_{acmSeeAlso} \times A_{acmSubTopic} \times A_{acmSuperTopic} \to A_{acmTopic}$

$\omega_{acmTopicPath} : A_{acmTopicName} \times A_{acmDashedTopicList} \to A_{acmTopicPath}$

$\omega_{acmDashedTopic} : A_{acmTopicName} \times A_{acmDashedTopicList} \to A_{acmDashedTopicList}$

$\omega_{acmComment} : A_{String} \to A_{acmComment}$

$\omega_{acmSeeAlsoList} : A_{acmSeeAlso} \times A_{acmSeeAlsoList} \to A_{acmSeeAlsoList}$

$\omega_{acmSeeAlso} : A_{acmTopicName} \to A_{acmSeeAlso}$

$\omega_{acmSubTopicList} : A_{acmSubTopic} \times A_{acmSubTopicList} \to A_{acmSubTopic}$

$\omega_{acmSubTopic} : A_{acmTopicPath} \to A_{acmSubTopic}$

$\omega_{acmSuperTopicList} : A_{acmSuperTopic} \times A_{acmSuperTopicList} \to A_{acmSuperTopicList}$

$\omega_{acmSuperTopic} : A_{acmSuperTopicName} \to A_{acmSuperTopic}$

$\omega_{acmTopicName} : A_{string} \to A_{acmTopicName}$

$\omega_{acmLiteral} : A_{string} \to A_{literal}$

Носителем алгеры является набор множеств

$M(G_{acm}) = \{A_{acmClassification}, A_{acmClassificationName}, A_{acmTopicList}, A_{acmTopic}, A_{acmTopicPath}, A_{acmDashedTopicList}, A_{acmComment}, A_{acmLabel}, A_{acmSeeAlsoList}, A_{acmSeeAlso}, A_{acmSubTopicList}, A_{acmSubTopic}, A_{acmSuperTopicList}, A_{acmSuperTopic}, A_{acmTopicName}, A_{literal}\}$

Грамматика целевого языка L_{tree}, предназначенного для представления тематического каталога в виде иерархии рубрик (ориентированного графа), в нормальной форме Бэкуса имеет вид:

<treeclassification>::= <treetopic>+

<treetopic> ::= **<k>**<treecode>**</k>**
<treetopicName><treetopicPath><treeTopicSuper>
[<treeSeeAlso>+][<treeTopicComment>]

<treecode> ::=<integer>

<treetopicName> ::= **”<n>”**<string>**”</n>”**

<treetopicPath> ::= <treeTopicName>”/”+ <treetopicName>

<treetopicSuper> ::= **”<superclass>”**<treecode>**”</superclass>”**

<treetopicComment> ::=**“<com>”**<string>**”</com>”**

<treetopicSeeAlso> ::=**”<s_a>”**<treecode>**”</s_a>”**

Для облегчения анализа данной грамматики преобразуем нотации БНФ < *nonTerminalList* >+ и {*nonTerminalList*} в отдельные нетерминальные символы типа <*nonTerminalList*>. Получим такие дополнительные нетерминалы:

<treetopicList> ::=< treetopic><treetopicList>
<treetopicSeeAlsoList> ::=< treetopicSeeAlso><treetopicSeeAlsoList>

Формально, порождающая грамматика языка L_{tree} имеет вид:

$G_{tree} = (A_{tree}, V_{n\_tree}, \sigma_{tree}, P_{tree})$,

где множество терминалов целевого языка

A_{tree} = **{<k>, </k>, <n>, </n>, <superclass>, </superclass>, <com>, </com>, <s_a>, </s_a>,** /};

множество нетерминалов целевого языка

V_{n_tree} = {<treeClassification>, <treetopic>, <treetopicList>, <treecode>, <treetopicName>, <treetopicPath>, <treetopicSuper>, <treetopicSeeAlso>, <treetopicComment>, <treetopicSeeAlsoList>};

аксиома целевого языка

σ_{tree}=<classification>

$MA_{tree} =< M(G_{tree}), \Omega_{tree} >$

$M(G_{tree}) = A_{string}$

Множество операций Ω_{tree}

$V_{treeClassification} : A_{treeClassificationName} \times A_{treeTopicList} \to A_{treeClassification}$

$V_{treeClassificationName} : A_{treeLiteral} \to A_{treeClassificationName}$

$V_{treeTopicList} : A_{treeTopic} \times A_{treeTopicList} \to A_{treeTopicList}$

$V_{treeTopic} : A_{treeTopicName} \times A_{treeCode} \times A_{treeTopicPath} \times A_{treeTopicSuper} \times A_{treeSeeAlso} \to A_{treeTopic}$

$V_{treeTopicPath} : A_{treeTopicName} \times A_{treeDashedTopicList} \to A_{treeTopicPath}$

$V_{treeDashedTopic} : A_{treeTopicName} \times A_{treeDashedTopicList} \to A_{treeDashedTopicList}$

$V_{treeCode} : A_{treeInteger} \to A_{treeCode}$

$V_{treeTopicName} : A_{treeString} \to A_{treeTopicName}$

$V_{treeTopicSuper} : A_{treeCode} \to A_{treeTopicSuper}$

$V_{treeComment} : A_{treeString} \to A_{treeComment}$

$V_{treeSeeAlso} : A_{treeCode} \to A_{treeComment}$

$V_{treeLiteral} : A_{treeString} \to A_{treeLiterat}$

Построим перевод языка L_{acm} в L_{tree}.

Синтаксическая алгебра языка L_{acm} имеет вид:

$MA_{acm}^{synt} =< A_{string}, \Omega_{acm} >$

Синтаксическая алгебра языка L_{tree} имеет вид:

$MA_{tree}^{synt} =< \{B_{\text{int}eger}, B_{string}\}, \Omega_{tree} >$

Сравним синтаксические алгебры для данных языков L_{acm} и L_{tree}. Для языка

L_{acm} система свободных образующих состоит из единственного множества A_{string}, а для языка L_{tree} система свободных образующих состоит из множеств $B_{\text{int}eger}, B_{string}$.

Таким образом, для данных языков L_{acm} и L_{tree}, ни системы свободных образующих, ни множества операций соответствующих алгебр не совпадают. Поэтому, отображение между порождающими алгебрами этих двух языков будем искать в виде квазигомоморфизма, предварительно выполним выравнивание систем свободных образующих таким образом:

1) будем полагать, что в алгебре S_{acm} система свободных образующих состоит из множеств $A_{acmString}$, $A_{acmInteger}$, причем $A_{acmInteger} \equiv \varnothing$. Везде, где необходимо, элементу $a \in A_{string}$ будем сопоставлять пару (a,b), где $b \in A_{integer}$, т.е. $b=0$
2) будем предполагать, что $A_{acmString} \subseteq B_{treeString}$, $A_{acmInteger} = \varnothing \subseteq B_{treeInteger}$

Обозначим $\begin{cases} A_{string} = A_{acmstring} \cup B_{treeString} \\ A_{\text{int}eger} = A_{acmInteger} \cup B_{treeInteger} \end{cases}$

Соотношения между порождающими алгебрами обоих языков, их синтаксическими алгебрами, показано на рисунке 2:

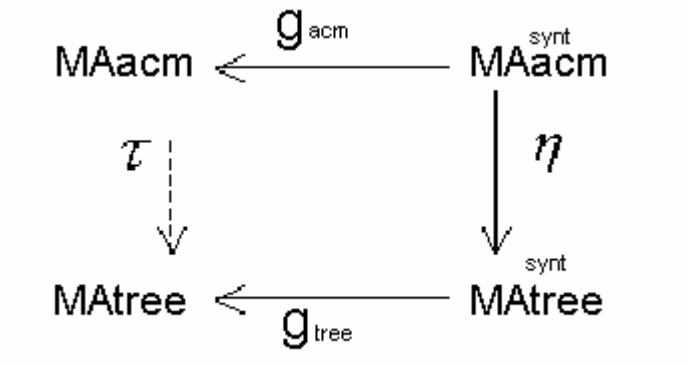

Рисунок 2 – Диаграмма соотношения между порождающими и синтаксическими алгебрами языков L_{acm} и L_{tree}.

Искомый перевод τ между двумя порождающими алгебрами $A_{acm\ u}\, A_{tree}$ существует, т.к. существует квазигомоморфизм η между синтаксическими алгебрами $MA_{acm}^{synt}\ u\ MA_{tree}^{synt}$

Построим квазигомоморфизм η.

Квазигомоморфизм η задается такой системой равенств, сопоставляющих операциям из синтаксической алгебры MA_{acm}^{synt} производные операции из синтаксической алгебры MA_{tree}^{synt}.

η_1: $\omega_{acmLiteral}(s) = V_{treeLiteral}(S)$, $s \in A_{string}$

η_2: $\omega_{acmInteger}(i) = V_{treeInteger}(i)$, $i \in A_{integer}$

η_3: $\omega_{acmTopicName}(\omega_{acmLiteral}(s)) = V_{treeLiteral}(s)$, $s \in A_{string}$

η_4: $\omega_{acmComment}(s) = V_{treeComment}(s)$, $s \in A_{string}$

η_5: $\omega_{acmDashedTopicList}(\omega_{acmTopicName}(\omega_{acmLiteral}, (s_1)), \omega_{acmDashedTopicList}(\omega_{acmTopicName}(\omega_{acmLiteral}(s_2))), \ldots) = V_{treeDashedTopicList}(V_{treeTopicName}(s_1), V_{treeDashedTopicList}(V_{treeTopicName}(s_2), \ldots))$ $s_1, s_2, \ldots \in A_{string}$

η_6: $\omega_{acmTopicName}(\omega_{acmTopicName}(\omega_{acmLiteral}(s_1)), \omega_{acmDashedTopicList}(\omega_{acmTopicName}(\omega_{acmLiteral}(s_2)), \ldots) = V_{treeTopicPath}(V_{treeTopicName}(s_1), V_{treeDashedTopicList}(V_{treeTopicName}(s_2), \ldots))$ $s_1, s_2, \ldots \in A_{string}$

Пополним синтаксическую алгебру S_{tree} операцией соответствия между элементами множества $A_{treeTopicPath}$ и элементами множества $A_{treeCode}$

V_1: $A_{treeTopicPath} \rightarrow A_{treeCode}$:

$\forall a \in A_{treeTopicPath}\ \exists! c \in A_{treeCode}$

Операции V_1 нет соответствия в A_{tree}

η_7: $\omega_{acmSeeAlso}(\omega_{acmTopicPath}(\omega_{acmTopicName}(\omega_{acmLiteral}(s)),\ldots) = V_{treeSeeAlso}(V_1(V_{treeTopicPath}(s), \ldots)))$

η_8: $\omega_{acmSuperTopic}$ [$\omega_{acmTopicPath}$($\omega_{acmTopicName}$($\omega_{acmLiteral}$(s_1)),… NULL]… $\omega_{acmDashedTopicList}$ ($\omega_{acmTopicName}$($\omega_{acmLiteral}$ (s_k)), NULL))] = $V_{treeTopicSuper}$ (V_1($V_{treeTopicPath}$($V_{treeTopicName}$(s_1),… $V_{treeDashedTopicList}$ ($V_{treeTopicName}$ (s_{k-1}), … $\underbrace{))))))))}_{к-1 скобка}$

η_9: $\omega_{acmSubTopic}$ ($\omega_{acmTopicPath}$($\omega_{acmTopicName}$($\omega_{acmLiteral}$(s_1)),… $\omega_{acmDashedTopicList}$ ($\omega_{acmTopicName}$($\omega_{acmLiteral}$ (s_k)), NULL)) = $V_{treeSubTopic}$ ($V_{treeTopicPath}$($V_{treeTopicName}$(s_1),… $V_{treeDashedTopicList}$ ($V_{treeTopicName}$ (s_k))))

Выравниваем сигнатуры порожденных и синтаксических алгебр, добавляя к $M(G_{tree})$ множество $A_{treeSubTopic} \equiv \varnothing$ и операцию $V_{treeSubTopic}$: $A_{treeTopicPath} \rightarrow V_{treeSubTopic}$

η_{10}: $\omega_{acmTopic}$ ($\omega_{acmTopicPath}$(s_1,…,s_{k1})), $\omega_{acmTopicName}$(…s_{k1+1})), $\omega_{acmComment}$(s_{k1+2}), $\omega_{acmSeeAlso}$(…s_{k1+3},…,s_{k2}), $\omega_{acmSubTopic}$(…, s_{k2+1},…,s_{k3}), $\omega_{acmSuperTopic}$(…s_{k3+1}, …,s_{k4}) = $V_{treeTopic}$ ($V_{treeTopicName}$(s_{k1+1})$V_{treeTopicPath}$ (…s_1…s_{k1}), V_1($V_{treeTopicPath}$(…s_1…s_{k1})), $V_{treeComment}$(s_{k1+2}), $V_{treeSeeAlso}$(…, s_{k1+3}, …,s_{k2}), $V_{treeTopicSuper}$(…, s_{k3+1},…, s_{k4}))

η_{11}: $\omega_{acmClassificationName}$ ($\omega_{acmLiteral}$(s)) = $V_{treeClassificationName}$ ($V_{treeLiteral}$(s))

η_{12}: $\omega_{acmClassificationName}$ ($\omega_{acmClassificationName}$, $\omega_{acmTopicList}$) = $V_{treeClassification}$ ($V_{treeClassificationName}$, $V_{treeTopicList}$)

На основании приведенного квазигомоморфизма в главе 3 построены алгоритмы перевода языка классификации ACM в язык представления древовидных структур.

2.2 Формальный подход к решению задачи интеллектуального поиска

Функция интеллектуального поиска является важной частью для решения задачи поиска. Поэтому можно назвать несколько методов ее решения, например: алгоритм полного перебора.

Алгоритм полного перебора для задачи классификации является заведомо правильным, но он требует большие затраты машинного и пользовательского времени. Т.е. проще говоря, пользователь, получивший список ссылок должен обработать их вручную, для выделения из всей массы ссылок нужные ему. При этом сам механизм поиска является заведомо неэффективным, что снижает вообще потребность в нем.

Наиболее распространенным является язык поиска, позволяющий составить логические выражения из набора терминов. При этом используются булевы операторы AND, OR, NOT. Тогда запрос может выглядеть следующим образом:

((информационная and система) or ИПС) not СУБД

Эта фраза означает: <*Найди все документы, которые содержат одновременно слова "информационная" и "система", либо слово "ИПС", но не содержат слова "СУБД"*>. . Такая схема достаточно проста и поэтому наиболее широко применяется в современных ИПС, однако еще 20 лет назад уже были хорошо известны ее недостатки.

Булев поиск плохо масштабирует выдачу. Оператор AND может очень сильно сократить число документов, выдаваемых на запрос. При этом все будет зависеть от того, насколько типичными для базы данных являются поисковые термины. Оператор OR, напротив, может привести к неоправданно широкому запросу, в котором полезная информация затеряется за информационным шумом. Для успешного применения этого ИПЯ следует хорошо знать лексику

системы и ее тематическую направленность. Как правило, для системы с таким поисковым языком создаются специальные документально - лексические базы данных со сложными словарями или тезаурусами, содержащими информацию о связи терминов словаря друг с другом.

Тем не менее, простота булевой модели поиска позволяет ее использование в произвольном тематическом каталоге. Запросы формулируются как произвольные булевы выражения, связывающие термины с помощью стандартных логических операций AND, OR или NOT.

Формально, задача булева поиска сводится к такой:

Пусть

$S=\{Si\}_{i=1,i}$ – множество публикаций в некоторой произвольной области.

$K=\{Kj\}_{j=1,j}$ – множество ключевых слов характеризующих эту произвольную область

$\forall Si \in S$, $i=1,I$ $\exists \{k_1,\ldots,k_{ni}\}$- множество ключевых слов характеризующих публикацию S_i

Отображение $f`_1$: $S \to K$ – ставит в соответствие каждой публикации $Si \in S$ множество ключевых слов Kj, $j=1,n_i$

Отображение $f`_2$: $K \to S$ – ставит в соответствие каждому ключевому слову $Kj \in K$ множество публикаций Si, $i=1,n_j$

Отображение $f`_3$: $K \times S \to \{0,1\}$ – ставит в соответствие каждой паре (K_j,S_i) значение 0 или 1 так, что:

$\forall k \in K$, $s \in S$

$f`_3(k,s)=0$, если $S \notin f_2(K)$

$f`_3(k,s)=1$, если $S \in f_2(K)$

Пусть также

$R=\{R_m\}_{m=1,M}$ – множество разделов тематического каталога для представления этой же предметной области.

$\forall R_m \in R$, $m=1,M$ $\exists \{k_1,\ldots,k_{nm}\}$- множество ключевых слов характеризующих раздел R_i

Отображение $f`_4$: $R \to K$ – ставит в соответствие каждому разделу $R_m \in R$ множество ключевых слов K_j, $j=1,n_m$

Отображение $f`_5$: $K \to R$ – ставит в соответствие каждому ключевому слову $K_j \in K$ множество разделов R_m, $m=1,n_j$

Отображение $f`_6$: $K \times R \to \{0,1\}$ – ставит в соответствие каждой паре (K_j,R_m) значение 0 или 1 так, что:

$\forall k \in K, r \in R$

$f`_6 (k,r)=0$, если $R \notin f_5(K)$

$f`_6 (k,r)=1$, если $R \in f_5(K)$

Отображение $f`_7$: $R \to S$ – ставит в соответствие каждому разделу $R_m \in R$ множество публикаций Si, $i=1,n_j$

Отображения $f_1 – f_7$ задают соотношения между тематическим каталогом, ключевыми словами, описывающими разделы этого каталога, и ключевыми словами, описывающими публикации.

Информационный запрос представляет собой набор слов, среди которых есть ключевые слова для данной предметной области.

Результат информационного запроса будем искать в виде множества публикаций и множества разделов, отобранных по ключевым словам, присутствующим в запросе.

Обозначим набор ключевых слов запроса $Q=\{k_1, \ldots, k_m\}$

Искомое множество публикаций $S=\{Si\}_{i=1,L}$ будет таким, что,

$\forall k \in Q, s \in S : f_3(k,s)=1$

Искомое множество разделов тематического каталога $R=\{R_m\}_{m=1,M}$ будет таким, что,

$\forall k \in Q, r \in R : f_6(k,r)=1$

2.3 Выводы

В данном разделе приводятся формальные подходы к задаче визуализации тематического каталога и к задаче информационного поиска. На основании определений грамматики формального языка, порождающей алгебры формального языка, строится перевод для формальных языков. Входной язык L_{acm} и выходной язык L_{tree} являются контекстно-свободными, для них были построены кс-грамматики, представленные в нормальной форме Бэкуса. Построены алгебры для данных грамматик и перевод для них. Приведена модель поиска – булева модель.

3. ИСПОЛЬЗОВАНИЕ ACM КЛАССИФИКАЦИИ В WEB – ПОРТАЛЕ ЭЛЕКТРОННЫХ ПУБЛИКАЦИЙ

3.1. Алгоритм преобразования тематического каталога

Пример части входной и выходной классификации приведен в приложении В.

Перевод между двумя языками L_{acm} и L_{tree} реализуется в виде алгоритма.

Словесное описание алгоритма:

1. На вход подается подаётся множество строк классификации ACM $S_1,\ldots,S_n$
2. Считывается S_i строка
3. Если S_i = <ACMTopic>, тогда i=i+1. Переходим к шагу 2.
4. Если S_i = </ACMTopic>, тогда i=i+1. Переходим к шагу 2.
5. Если S_i = <SubTopic>, тогда вызываем метод обработки тега <SubTopic>, i=i+1. Переходим к шагу 2.
6. Если S_i = <Comment>, тогда вызываем метод обработки тега <Comment>, i=i+1. Переходим к шагу 2.
7. Если S_i = <SuperTopic>, тогда вызываем метод обработки тега <SuperTopic>, i=i+1. Переходим к шагу 2.
8. Если S_i = <SeeAlso>, тогда вызываем метод обработки тега <SeeAlso>, i=i+1. Переходим к шагу 2.
9. Конец

Визуальное представление алгоритма: рисунок 3.

Программа, реализованная на базе данного алгоритма приведена в приложении Г.

3.2 Алгоритм поиска релевантных документов по тематическому каталогу

1. Каждой публикации ставится в соответствие набор ключевых слов, наиболее полно определяющих тему публикации.
2. Каждый раздел тематического каталога также обладает набором связанных с темой раздела ключевых слов.
3. Разделяют два вида поиска:
4. Первый способ поиска предполагает, что алгоритм поиска выполняет поиск статей, в анотациях которых есть все ключевые слова из запроса.
5. Второй способ поиска предполагает, что алгоритм поиска выполняет поиск статей, относящихся к таким разделам каталога, в списках ключевых слов для которых есть ключевые слова из запроса.

Словесное представление алгоритма:

1. Пусть $К_1$, …, $К_n$ – ключевые слова для поиска.
2. Строим множество всех публикаций, в аннотациях которых встречаются ключевые слова $К_1$, …, $К_n$

$$S^1 = \bigcap_{i=1}^{n} f_2(K_i)$$

3. Строим множество публикаций в разделах, список ключевых слов для которых содержит ключевые слова $К_1$, …, $К_n$

$$S^2 = \bigcap_{i=1}^{n} f_5(K_i)$$

4. Результирующее множество статей S образуется таким образом:

$$S = S^1, \ если \quad S^2 = \varnothing$$

5. $$S = S^2, \ если \quad S^1 = \varnothing$$
$$S = S^1 \cap S^2, \ если \quad S^1 \ и \ S^2 \neq \varnothing$$

6. Конец

Визуальное представление алгоритма: рисунок 4.

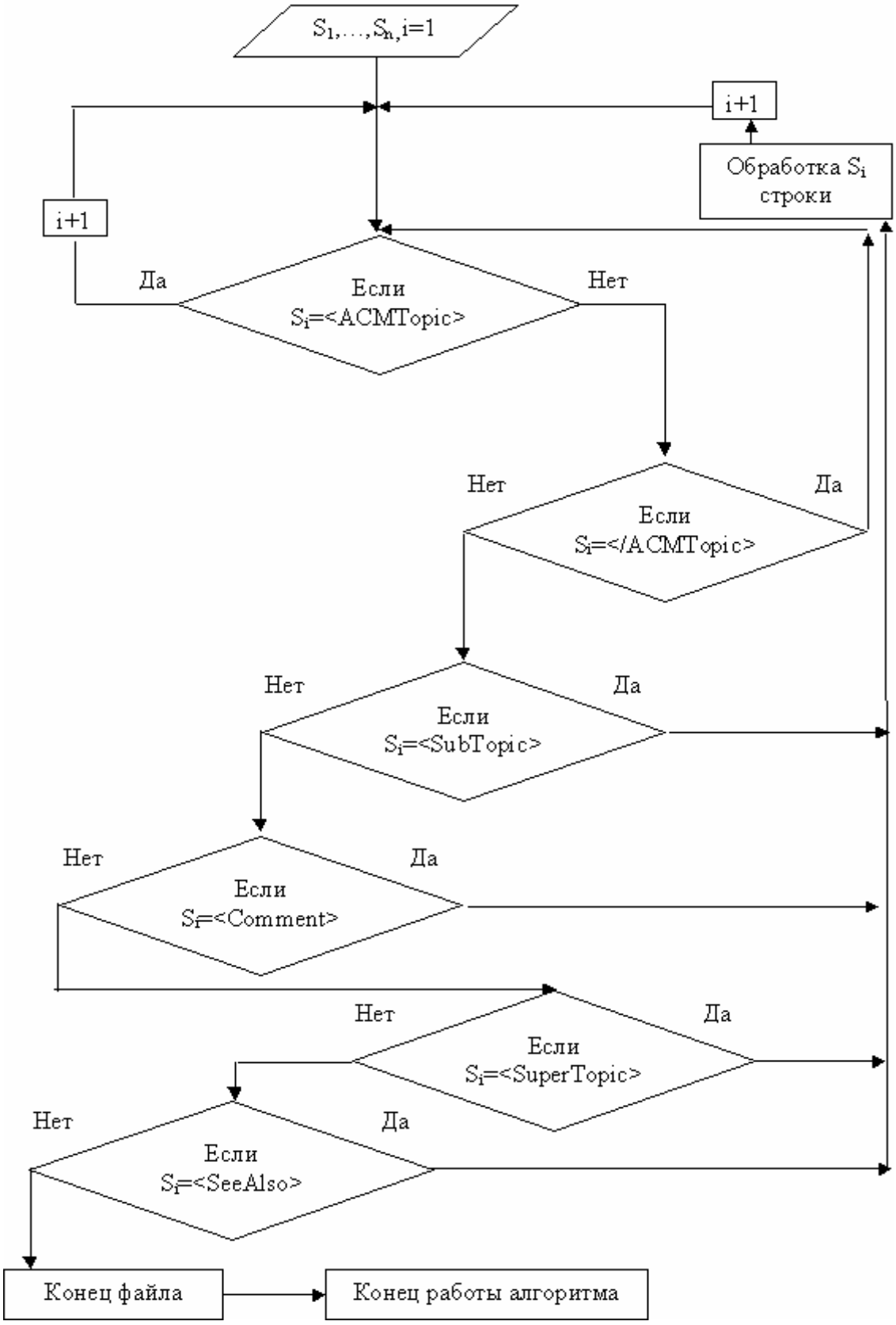

Рисунок 3 - Алгоритм перевода между двумя языками L_{acm} и L_{tree}

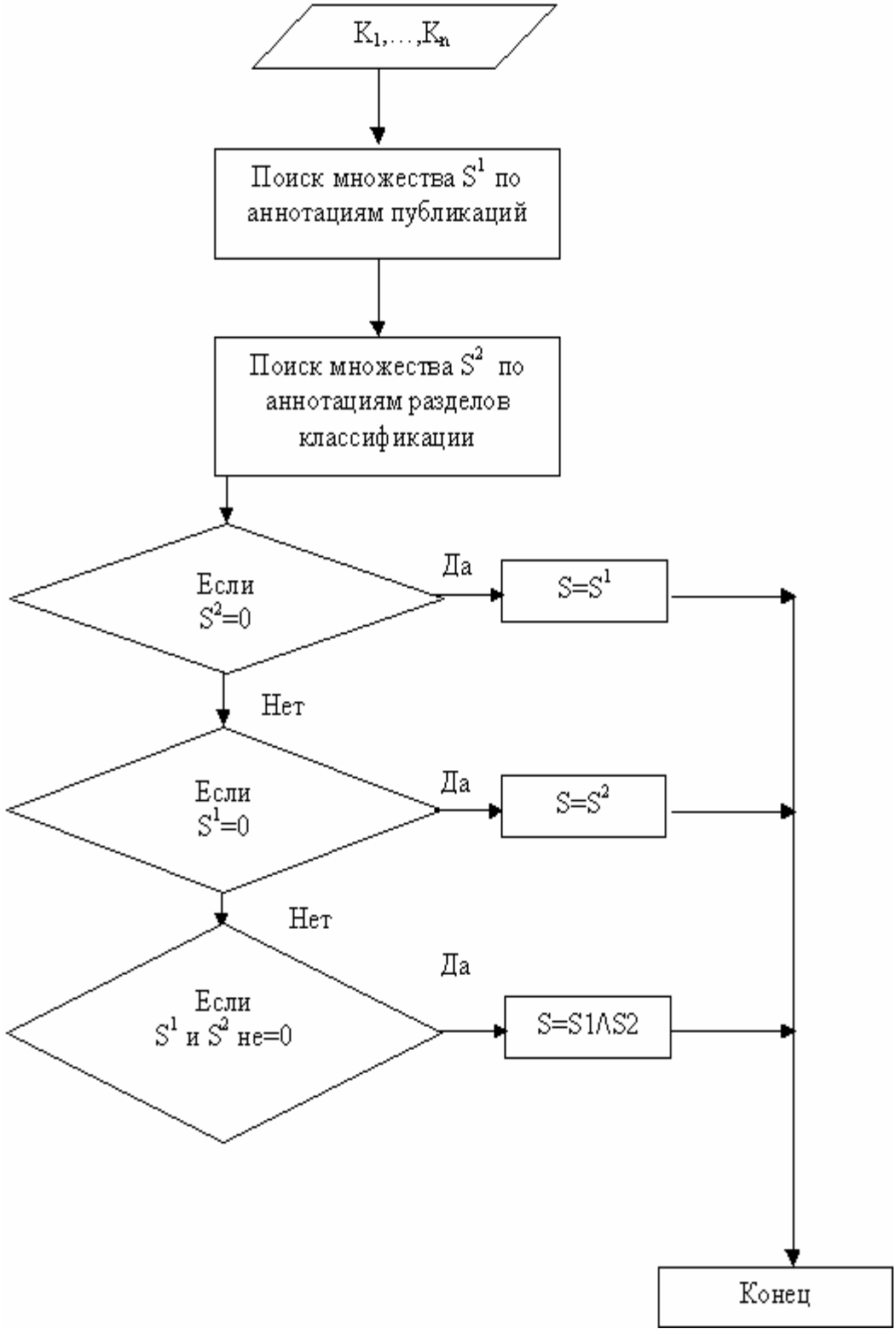

Рисунок 4 - Алгоритм поиска по ключевым словам разделов и аннотациям публикаций.

3.3 Выводы

В данной главе детально рассмотрены алгоритмы перевода входного языка, представленного в БНФ в язык представления древовидных структур и алгоритм интеллектуального поиска.

Рассматривая структуру поиска нужно сказать, что она включает в себя два вида поиска. Механизм первого поиска заключается в том, что автор публикации должен добавит список ключевых слов в ее аннотацию и тогда можно сравнивать список ключевых слов введенных пользователем, с списком ключевых слов в аннотациях публикаций. Механизм второго поиска заключается в том, что выдается множество тех публикаций разделов, ключевые слова которых совпадают с ключевыми словами, введенными пользователем.

ЗАКЛЮЧЕНИЕ

В данной дипломной работе сделано введение в проблему использования тематических каталогов для классификации публикаций. Рассмотрены примеры классификаций электронных изданий. Также рассмотрены задачи построения тематического каталога публикаций в области компьютерных наук и функции интеллектуального поиска.

К основным задачам, решаемым с использованием тематических каталогов публикаций, относятся:

- использование тематического каталога для поиска подходящих публикаций;
- использование тематического каталога для классификации новых публикаций;

Решение таких задач требует таких практических подходов:

- визуализация тематического каталога в виде иерархии концепций, для навигации, анализа и редактирования;
- экспорт каталогов из одного языкового представления в другое;
- поиск публикаций, удовлетворяющих заданным описаниям разделов тематического каталога;

В работе приводятся формальные подходы к задаче визуализации тематического каталога и к задаче информационного поиска. На основании определений грамматики формального языка, порождающей алгебры формального языка, строится перевод для формальных языков. Входной язык L_{acm} и выходной язык L_{tree} являются контекстно-свободными, для них были построены кс-грамматики, представленные в нормальной форме Бэкуса. Построены алгебры для данных грамматик и перевод для них. Приведена булева модель поиска.

Детально рассмотрены алгоритмы перевода входного языка, представленного в БНФ в язык представления древовидных структур и алгоритм интеллектуального поиска.

Использованная литература:

1. NEC ResearchIndex http://citeseer.nj.nec.com/directory.html доступ проверялся
2. DBLP http://dblp.uni-trier.de/ доступ проверялся17.05.2002 г.
3. AgentLink http://www.agentlink.org/resources/clearing-house-view.html доступ проверялся 17.05.2002 г.
4. В.В. Корнеев, А.Ф. Гарев, С.В. Васютин, В.В. Райх “Базы данных: интеллектуальная оработка информации” Москва: Нолидж, 2000. – 352с.
5. Глушков В.М., Цейтлин Г.Е., Ющенко Е.Л. Алгебра. Языки. Программирование. Киев, Наукова думка, 1989.-376 с.
6. http://www.w3c.org/xml доступ проверялся 29.05.2002г.
7. http://w3c.org/rdf доступ проверялся 29.05.2002г.
8. Летичевский А.А. Синтаксис и семантика формальных языков. //Кибернетика.-1968. №4, с.1-10.
9. ACM CS Classification http://daml.umbc.edu/ontologies/classification доступ проверялся17.05.2002 г.

Приложение А

Пространство имен ACM классификации электронных публикаций (XML-представление):

```
<?xml version="1.0"?>
<rdf:RDF
        xmlns:rdf  ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:daml ="http://www.daml.org/2001/03/daml+oil#"
        xmlns:rdfs ="http://www.w3.org/2000/01/rdf-schema#"
          xmlns:prsn ="http://daml.umbc.edu/ontologies/person-ont#"
        xmlns      ="http://daml.umbc.edu/ontologies/topic-ont#"
>
        <daml:Ontology rdfs:about="">
                <daml:versionInfo>$Revision: 1.3 $</daml:versionInfo>
                <daml:imports rdf:resource="http://www.daml.org/2001/03/daml+oil"/>
        </daml:Ontology>
        <daml:Class rdf:ID="Topic">
                <daml:label>Topic</daml:label>
                <daml:subClassOf
rdf:resource="http://www.daml.org/2001/03/daml+oil#Thing"/>
        </daml:Class>
        <rdf:Property rdf:ID="SubTopic">
                <daml:domain rdf:resource="#Topic"/>
                <daml:range rdf:resource="#Topic"/>
        </rdf:Property>
        <rdf:Property rdf:ID="SuperTopic">
                <daml:domain rdf:resource="#Topic"/>
                <daml:range rdf:resource="#Topic"/>
        </rdf:Property>
```

```
        <rdf:Property rdf:ID="Keywords">
                <daml:domain rdf:resource="#Topic"/>
                <daml:range
rdf:resource="http://www.daml.org/2001/03/daml+oil#Literal"/>
        </rdf:Property>
        <daml:Class rdf:ID="InfoResource">
                <daml:label>InfoResource</daml:label>
                <daml:subClassOf
rdf:resource="http://www.daml.org/2001/03/daml+oil#Thing"/>
        </daml:Class>
        <rdf:Property rdf:ID="Interestedin">
                <daml:domain rdf:resource="http://daml.umbc.edu/ontologies/person-
ont#Person"/>
                <daml:range
rdf:resource="http://www.daml.org/2001/03/daml+oil#Thing"/>
        </rdf:Property>
        <rdf:Property rdf:ID="Aboutsubject">
                <daml:domain rdf:resource="#InfoResource"/>
                <daml:range
rdf:resource="http://www.daml.org/2001/03/daml+oil#Thing"/>
        </rdf:Property>
        <rdf:Property rdf:ID="Relatedto">
                <daml:domain rdf:resource="#Topic"/>
                <daml:range rdf:resource="#Topic"/>
        </rdf:Property>
</rdf:RDF>
```

Приложение Б

Основные теоретические положення других авторов

Б.1 Модели и универсальные алгебры

Модель $M_A =< A,\pi >$ - система, состоящая из непустого множества $A \neq \varnothing$ и определенного на данном множестве A множество предикатов $\pi = \{\rho_s^{n_s} s = 1,2,...\}$. Предикат $\rho^n(a_{i_1},...,a_{i_n})$ истинен, если для кортежа элементов $a_{i_1},...,a_{i_n} \in A$ выполняется отношение $\xi^n(a_{i_1},...,a_{i_n})$. Последовательность $n_1,...,n_s,...$ - тип модели M_A, а совокупность $\pi = \{\rho_s^{n_s} s = 1,2,...\}$ - сигнатура модели.

Рассмотрим $M_A =< A,\pi >$, где каждый n - местный предикат, входящий в $\boldsymbol{\pi}$, соответствует функциональному отношению φ^n на множестве A. С каждым таким отношением φ^n связана частично определенная (n-1)-местная функция $F_{\varphi^n}(x_1,...,x_{n-1})$.

Так как φ^n - функционально, то для набора $a_{i_1},...,a_{i_n} \in A$ существует не более одного $a_{i_{n+1}} \in A$, такого, что $(a_{i_1},...,a_{i_n},a_{i_{n+1}}) \in \varphi_A^{n+1}$, причем $a_{i_{n+1}} = F_{\varphi^{n+1}}(a_{i_1},...,a_{i_n})$. Функции называются n-арными частичными операциями.

Частичная универсальная алгебра $U_A =< A,\Omega >$ - система, состоящая из основного множества A и определенной на нем совокупности частичных операций $\Omega = \{F_s^{n_s}(x_q,...,x_{n_s}) : s = 1,2,...\}$. Сигнатура Ω может быть конечной и бесконечной.

Обобщением понятия универсальной алгебры является понятие многоосновной алгебры.

Многоосновная алгебра – система $<\mathrm{M},\Omega>$, состоящая из семейства множеств $\mathrm{M} = \{A_\alpha : \alpha \in I\}$, которые называются основными, и сигнатуры

операций Ω, определенных на M следующим образом: каждой n-местной операции однозначно сопоставлен кортеж $<\alpha_1,...,\alpha_n;\alpha_r>$ - схема данной операции, так что $F_{<\alpha_1,...,\alpha_n;\alpha_r>}$ является отображением декартова произведения $A_{\alpha_1}\times A_{\alpha_2}\times...\times A_{\alpha_n}$ во множество A_{α_r}, где $\alpha_i,\alpha_r\in I(i=1,...,n)$.

Таким образом, операции $F_{<\alpha_1,...,\alpha_n;\alpha_r>}\in\Omega$ являются n-местными функциями $F(x_1,...,x_n)$, аргументы которых $x_q(q=1,...,n)$ определены на множествах $A_{\alpha_q}\in\mathrm{M}$, а сама функция F принимает значения на множестве. Каждая 0-местная операция $F_{<\alpha_r>}\in\Omega$ фиксирует в $A_{\alpha_r}\in\mathrm{M}$ некоторый элемент (константу) $a\in A_{\alpha_r}$.

Пусть $X=\{X_\alpha:\alpha\in I\}$ - некоторое семейство множеств.

Свободная многоосновная алгебра $S=<\{S_\alpha:\alpha\in I\};\Omega>$, порожденная семейством X, называется алгебра, элементы которой – слова в алфавите $\{F:F\in\Omega\cup X\}$, причем по индукции:

а) однобуквенными словами алгебры S являются элементы множеств $X_\alpha\subset S_\alpha(\forall\alpha\in I)$ и только они;

б) если $s_1\in S_{\alpha_1},...,s_n\in S_{\alpha_n}$ - элементы алгебры $\boldsymbol{S}$, то для любой n-местной операции $F\in\Omega$ со схемой $<\alpha_1,...,\alpha_n;\alpha_r>$ слово $Fs_1s_2...s_n\in S_{\alpha_r}$ также элемент алгебры S.

Б.2 Контекстно-свободные грамматики

Грамматика $G=(\mathrm{A},V_n,\sigma,P)$ - порождающая грамматика, где $\mathrm{A}=\{a_1,a_2,...,a_m\}$ - основной (терминальный) алфавит, V_n - конечный вспомогательный (нетерминальный) алфавит, символы которого обозначаются малыми греческими буквами, $\sigma\in V_n$ - начальный (нетерминальны) символ – аксиома, $P=\{\psi_i\to u_i\},i=1,2,...,k$ - конечная система подстановок (схема грамматики), левые и правые части которых есть цепочки $u_i,v_i\in F(v)$, где $F(v)$ -

свободная полугруппа над объединенным алфавитом $V = V_n \cup \mathrm{A}$, символ $\to$ является внешним и не принадлежит объединенному алфавиту.

Грамматика $G = (\mathrm{A}, V_n, \sigma, P)$ называется контекстно-свободной, если каждое правило схемы P имеет вид $\psi \to z$, где $\psi \in V_n$, а z - непустая цепочка над объединенным алфавитом $V = V_n \cup \mathrm{A}$.

Б.3 Построение многоосновной алгебры для кс –грамматики

Для любой кс - грамматики (цит.по [5]) $G = (\mathrm{A}, V_n, \sigma, P)$ может быть построена многоосновная алгебра $\mathrm{M}(G) = \{A_\psi : \psi \in V_n\}$ с выделенной компонентой A_σ, $L(G) = A_\sigma$ (т.е. цепочки L - элементы множества A_σ)

Метод построения алгебры:

Рассматривается основное множество $\mathrm{M}(G) = \{A_\psi : \psi \in V_n\}$, A_ψ - множество всех терминальных цепочек, выводимых из нетерминала ψ в грамматике G.

Каждой кс-продукции $p : \psi_i \to u_{i_1}\psi_1^i u_{i_2}\psi_2^i ... u_{i_{r_i}}\psi_{r_i}^i \bullet u_{i_{r_i+1}} \in P$ поставим в соответствие конкатенарную операцию $\omega_p : A_{\psi_1^i} \times A_{\psi_2^i} \times ... \times A_{\psi_{r_i}^i} \to A_{\psi_i}$, определенную на множествах $A_{\psi_1^i}, ..., A_{\psi_{r_i}^i}$ и такую, что при подстановке вместо соответствующих аргументов произвольных цепочек $x_s \in A_{\psi_s^i} (1 \le s \le r_i)$ значением операции ω_p является цепочка $x = u_{i_1} x_1 u_{i_2} x_2 ... u_{i_{r_i}} x_{r_i} u_{i_{r_i+1}} \in A_{\psi_i}$

Конкатенарные операции ω_p, ассоциированные со схемой P кс-грамматики G, составляют сигнатуру Ω. Построенная таким образом многоосновная алгебра $< \mathrm{M}(G), \Omega >$ с выделенной основной компонентой A_σ, соответствующей аксиоме σ грамматики G, называется кс-многоосновной алгеброй, ассоциированной с грамматикой G.

Б.4 Перевод для формальных языков

Пусть A и A’ – порождающие алгебры двух формальных языков, S и S’ – соответствующие им синтаксические алгебры. При условии, что семантическая алгебра для S и S' - общая, диаграмма соотношений между отображениями на порождающих алгебрах и на соответствующих им синтаксических алгебрах показана на рисунке 5.

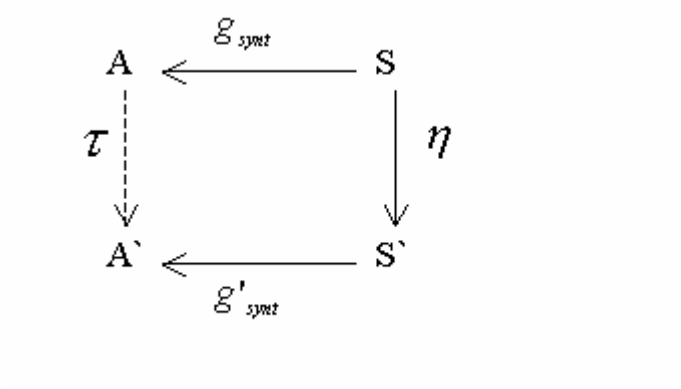

Рисунок 5 – Диаграмма соотношений между отображениями.

Отображение η - отображение синтаксической алгебры S в синтаксическую алгебру S'. Отображение состоит из множества равенств, показывающих семантическую эквивалентность операций Ω алгебры S операциям Ω' алгебры S'.

Отображение τ - искомое отношение перевода на порождающих алгебрах.

Тогда перевод для любого $a \in A$ осуществляется в три этапа.

1. Нахождение синтаксического разложения s элемента a по компонентам системы свободных образующих синтаксической алгебры S с использованием операций синтаксической алгебры:

$$s \in g_{synt}^{-1}(a)$$

2. Перевод на синтаксическом уровне, или нахождение синтаксического разложения $s' \in S'$ с использованием равенств из множества η, связывающих

операции из Ω и Ω':

$$s' = \eta s$$

3. Получение элемента a' порождающей алгебры A', семантически эквивалентного исходному элементу a:

$$a' = g'_{synt}\, s'$$

В [8] отмечается, что если синтаксические алгебры имеют одинаковую сигнатуру операций и одинаковые системы свободных образующих, то перевод одной алгебры в другую осуществляется тождественным отображением системы образующих алгебры A в систему образующих алгебры A', с последующим тождественным отображением операций.

Если две синтаксические алгебры имеют одинаковые системы свободных образующих, но множества и сигнатуры операций этих алгебр различаются, то вместо гомоморфного отображения операций одной алгебры в другую ищут отображение более общего вида, называемое квазигомоморфизмом.

Определение

Пусть $A = < M_1, \Omega_1 >, A' = < M_2, \Omega_2 >$ - многоосновные алгебры.

Отображение $\varphi : A \to A'$ называется **квазигомоморфизмом**, если для каждого $\omega \in \Omega_1$ в алгебре *A'* найдется производная операция $H\omega$, имеющая ту же схему, что и ω, и для любой *n*-арной операции ω будет выполнено условие $\varphi(\omega(a_1,...,a_n)) = H\omega(\varphi(a_1),...,\varphi(a_n))$. Операция $H\omega$ называется образом операции ω при квазигомоморфизме φ.

Если же две синтаксические алгебры имеют разные системы свободных образующих, и разные множества и сигнатуры операций, то вначале выполняют выравнивание систем свободных образующих обоих синтаксических алгебр, добавляя недостающие множества в обе алгебры, и считая добавленные множества пустыми. Затем ищут квазигомоморфизм на

алгебрах с выровненными системами свободных образующих.

Для выполнения перевода с одного формального языка на другой в алгебраическом подходе предполагается наличие в обоих языках синтаксических и семантических алгебр.

Дан язык L, порождающая его алгебра $A =< \mathrm{M}, \Omega >$, где $\mathrm{M} = \{A_\alpha : \alpha \in I\}$ - семейство множеств.

Синтаксическая структура языка задается порождающей алгеброй $A =< \mathrm{M}, \Omega >$, свободной алгеброй S и гомоморфизмом φ алгебры S на A. При этом S называется синтаксической алгеброй языка, а φ - синтаксическим гомоморфизмом.

Семантическая структура языка задается алгеброй G, синтаксической алгеброй S и гомоморфизмом ψ алгебры S на G. При этом алгебра G называется семантической алгеброй, а ψ - семантическим гомоморфизмом.

Семантическая алгебра строится как фактор-алгебра синтаксической алгебры S по некоторому отношению конгруэнтности [5].

При переводе с одного формального языка на другой семантические структуры этих языков предполагаются связанными, а именно, «основные компоненты семантических алгебр должны совпадать» (цит. по Летичевскому). В [8, стр. 1-10] рассматриваются отношения перевода для таких ситуаций: языки с одинаковыми сигнатурами операций порождающих алгебр и общей семантической алгеброй, языки с разными сигнатурами операций порождающих алгебр.

Б.5 Изолированные множества и конгруэнции.

Понятия гомоморфизма и конгруэнции играют важную роль при изучении алгебраических систем, а также в теории формальных языков.

При построении перевода с одного формального языка на другой используются синтаксическая и семантическая алгебры, которые строятся как алгебры, гомоморфные порождающей алгебре формального языка. Собственно, сам перевод также является гомоморфизмом между синтаксическими и семантическими алгебрами соответственно двух языков.

Приведем основные определения, необходимые для выполнения алгебраического анализа формальных языков.

Пусть задано отображение φ множества A в B, где A, B - основные множества в однотипных универсальных алгебрах $U_A =< A,\Omega >$, $U_B =< B,\Omega >$ соответственно. Отображение φ называется **гомоморфным** отображением алгебры U_A в алгебру U_B, если для любых элементов $a_1,a_2,...,a_n \in A$ и произвольной *n*-арной операции $F \in \Omega$ выполняется соотношение

$$[F(a_1,a_2,....,a_n)]\varphi = F[(a_1)\varphi,(a_2)\varphi,...,(a_n)\varphi],$$

где $(a_i)\varphi = b_i \in B, i = \overline{1,n}$

Если при этом отображение φ - взаимно-однозначное, то φ называют **изоморфным** отображением алгебры U_A в алгебру U_B.

Б.6 Системы образующих

Пусть множество $A_1 \subseteq A$ замкнуто в алгебре U_A, т.е. $A_1 = [A_1]$. Тогда систему $U_{A_1} =< A_1,\Omega >$ можно рассматривать как универсальную алгебру. Такая алгебра будет называться **подалгеброй** универсальной алгебры $U_A =< A,\Omega >$.

Пусть $U_{A_1} =< A_1,\Omega >$ - некоторая подалгебра универсальной алгебры $U_A =< A,\Omega >$. Система элементов $\Sigma \subseteq A_1$ называется **системой образующих** или **полной системой подалгебры**, если $[\Sigma] = A_1$. Если $[\Sigma] = A$, система Σ называется **системой образующих алгебры** U_A.

Универсальная алгебра $U_A =< A, \Omega >$ называется **конечно-порожденной**, если она имеет конечную систему образующих. Конечно-порожденная универсальная алгебра имеет конечный базис (теорема 3.6, стр 78. [5])

Порождающая грамматика языка L может быть представлена в виде:

$G = (\mathrm{A}, V_n, \sigma, P)$,

где $\mathrm{A} = \{a_1, a_2, ..., a_m\}$ - основной (терминальный) алфавит,

V_n - конечный вспомогательный (нетерминальный) алфавит,

$\sigma \in V_n$ - начальный (нетерминальный) символ – аксиома,

$P = \{\psi_i \to u_i\}, i = 1,2,...,k$ - конечная система подстановок (схема грамматики), левые и правые части которых есть цепочки $u_i, v_i \in F(v)$, где $F(v)$ - свободная полугруппа над объединенным алфавитом $V = V_n \cup \mathrm{A}$, символ $\to$ является внешним и не принадлежит объединенному алфавиту.

Приложение В

В.1 Пример ACM классификации на входном языке L_{acm}

```
<acm:Topic rdf:ID="ACMTopic">
  <rdfs:label>ACM CS Topic</rdfs:label>
  <rdfs:subClassOf rdf:resource="Topic" />
  <acm:SubTopic>ACMTopic/General_Literature</acm:SubTopic>
  <acm:SubTopic>ACMTopic/Hardware</acm:SubTopic>
  <acm:SubTopic>ACMTopic/Computer_Systems_Organization</acm:SubTopic>
  <acm:SubTopic>ACMTopic/Software</acm:SubTopic>
  <acm:SubTopic>ACMTopic/Data</acm:SubTopic>
  <acm:SubTopic>ACMTopic/Theory_Of_Computation</acm:SubTopic>
  <acm:SubTopic>ACMTopic/Mathematics_Of_Computing</acm:SubTopic>
  <acm:SubTopic>ACMTopic/Information_Systems</acm:SubTopic>
  <acm:SubTopic>ACMTopic/Computing_Methodologies</acm:SubTopic>
  <acm:SubTopic>ACMTopic/Computer_Applications</acm:SubTopic>
  <acm:SubTopic>ACMTopic/Computing_Milieux</acm:SubTopic>
</acm:Topic>

<ACMTopic rdf:ID="ACMTopic/General_Literature">
  <rdfs:label>General Literature</rdfs:label>
  <acm:SuperTopic>ACMTopic</acm:SuperTopic>
  <acm:SubTopic>ACMTopic/General_Literature/General</acm:SubTopic>

<acm:SubTopic>ACMTopic/General_Literature/Introductory_And_Survey</acm:Su
bTopic>
  <acm:SubTopic>ACMTopic/General_Literature/Reference</acm:SubTopic>
```

<acm:SubTopic>ACMTopic/General_Literature/Miscellaneous</acm:SubTopic>
</ACMTopic>

B.1 Пример представления иерархии классификации на выходном языке L_{tree}

<k>1</k><n>General_Literature</n> ACMTopic/General_Literature <s>0</s><s_a></s_a><com></com><END>
<k>2</k><n>Hardware</n> ACMTopic/Hardware <s>0</s><s_a></s_a><com></com><END>
<k>3</k><n>Computer_Systems_Organization</n> ACMTopic/Computer_Systems_Organization <s>0</s><s_a></s_a><com></com><END>
<k>4</k><n>Software</n> ACMTopic/Software <s>0</s><s_a></s_a><com></com><END>
<k>5</k><n>Data</n> ACMTopic/Data <s>0</s><s_a></s_a><com></com><END>
<k>6</k><n>Theory_Of_Computation</n> ACMTopic/Theory_Of_Computation <s>0</s><s_a></s_a><com></com><END>
<k>7</k><n>Mathematics_Of_Computing</n> ACMTopic/Mathematics_Of_Computing <s>0</s><s_a></s_a><com></com><END>
<k>8</k><n>Information_Systems</n> ACMTopic/Information_Systems <s>0</s><s_a></s_a><com></com><END>
<k>9</k><n>Computing_Methodologies</n> ACMTopic/Computing_Methodologies <s>0</s><s_a></s_a><com></com><END>
<k>10</k><n>Computer_Applications</n> ACMTopic/Computer_Applications <s>0</s><s_a></s_a><com></com><END>
<k>11</k><n>Computing_Milieux</n> ACMTopic/Computing_Milieux <s>0</s><s_a></s_a><com></com><END>

<k>12</k><n>General</n> ACMTopic/General_Literature/General <s>1</s><s_a></s_a><com></com><END>
<k>13</k><n>Introductory_And_Survey</n> ACMTopic/General_Literature/Introductory_And_Survey <s>1</s><s_a></s_a><com></com><END>
<k>14</k><n>Reference</n> ACMTopic/General_Literature/Reference <s>1</s><s_a></s_a><com>e.g., dictionaries, encyclopedias, glossaries</com><END>
<k>15</k><n>Miscellaneous</n> ACMTopic/General_Literature/Miscellaneous <s>1</s><s_a></s_a><com></com><END>
<k>16</k><n>Biographies_Autobiographies</n> ACMTopic/General_Literature/General/Biographies_Autobiographies <s>12</s><s_a></s_a><com></com><END>
<k>17</k><n>Conference_Proceedings</n> ACMTopic/General_Literature/General/Conference_Proceedings <s>12</s><s_a></s_a><com></com><END>
<k>18</k><n>General_Literary_Works</n> ACMTopic/General_Literature/General/General_Literary_Works <s>12</s><s_a></s_a><com>e.g., fiction, plays</com><END>
<k>19</k><n>General</n> ACMTopic/Hardware/General <s>2</s><s_a></s_a><com></com><END>
<k>20</k><n>Control_Structures_And_Microprogramming</n> ACMTopic/Hardware/Control_Structures_And_Microprogramming <s>2</s><s_a>323</s_a><com></com><END>
<k>21</k><n>Arithmetic_And_Logic_Structures</n> ACMTopic/Hardware/Arithmetic_And_Logic_Structures <s>2</s><s_a></s_a><com></com><END>
<k>22</k><n>Memory_Structures</n> ACMTopic/Hardware/Memory_Structures <s>2</s><s_a></s_a><com></com><END>
<k>23</k><n>Input_Output_And_Data_Communications</n> ACMTopic/Hardware/Input_Output_And_Data_Communications <s>2</s><s_a></s_a><com></com><END>

<k>24</k><n>Register_Transfer_Level_Implementation</n> ACMTopic/Hardware/Register_Transfer_Level_Implementation <s>2</s><s_a></s_a><com></com><END>

<k>25</k><n>Logic_Design</n> ACMTopic/Hardware/Logic_Design <s>2</s><s_a></s_a><com></com><END>

<k>26</k><n>Integrated_Circuits</n> ACMTopic/Hardware/Integrated_Circuits <s>2</s><s_a></s_a><com></com><END>

<k>27</k><n>Performance_And_Reliability</n> ACMTopic/Hardware/Performance_And_Reliability <s>2</s><s_a>127</s_a><com></com><END>

<k>28</k><n>Miscellaneous</n> ACMTopic/Hardware/Miscellaneous <s>2</s><s_a></s_a><com></com><END>

Приложение Г

Текст программы перевода классификации на входном языке L_{acm} в классификацию на выходном языке L_{tree}

```
#include <stdio.h>
 #include <iostream.h>
 #include <conio.h>
 #include <string.h>
 #include <stdlib.h>

ACMTopic (void);
int ACM_SubTopic (char *);
int ACM_SuperTopic (char *);
int Comments (char *);
int Rdf_See_Also (char *);

int main() {
 FILE *acm_class, *result_acm, *result_comment, *result_see_al;
 int flag=0, langth=0;
 char str_acm[300], *spec;
 char *uk_1="<ACMTopic rdf:ID=", *uk_2="</ACMTopic>",*ptr_1,*ptr_2,*ptr_3;
 char *ptr_4, *ptr_5, *uk_3="=",*uk_4="\n", *uk_5;
 clrscr();

if ((acm_class = fopen("acm_clas.txt", "r")) == NULL)
{
   fprintf(stderr, "Cannot open acm_class file, rename file classifica\n");
   fprintf(stderr, "tion.daml to acm_clas.txt");
   spec="Переименуйте файл classification.daml в файл acm_clas.txt";
   cout<<spec;
   return 1;
}
if ((result_acm = fopen("res_ACM.txt", "w+t")) == NULL)
```

```
{
 fprintf(stderr, "Cannot open res_ACM file\n");
 spec="Переименуйте файл classification.daml в файл acm_clas.txt";
 cout<<spec;
 return 1;
}

if ((result_comment = fopen("res_cmnt.txt", "w+t")) == NULL)
{
 fprintf(stderr, "Cannot open res_Sals file\n");
 spec="Переименуйте файл classification.daml в файл acm_clas.txt";
 cout<<spec;
 return 1;
}

if ((result_see_al = fopen("res_sAls.txt", "w+t")) == NULL)
{
 fprintf(stderr, "Cannot open res_cmnt file\n");
 spec="Переименуйте файл classification.daml в файл acm_clas.txt";
 cout<<spec;
 return 1;
}

int counter_acm=1;
while (flag==0)
{
   fgets(str_acm,300,acm_class   ptr_1=strstr(str_acm,uk_1);

   if (ptr_1!=0)
   {
```

```
    ptr_4=strstr(str_acm,uk_3);
    ptr_4=ptr_4+2;
    langth=strlen(ptr_4);
    ptr_4[langth-3]='\0';
    char L[200]={NULL};
    strcpy(L,ptr_4);
    ptr_2=NULL;
    while(ptr_2==0)
    {

        fgets(str_acm,300,acm_class);
        ptr_2=strstr(str_acm,uk_2);
char *p1="<acm:SubTopic>",*p2="<rdfs:comment>",*p4="\n";
char *p3="<rdfs:seeAlso",*pt1,*pt2,*pt3, p[6], *p5, *p6;
char *q1="<k>",*q2="</k>",*q3="<n>",*q4="</n>",*q5="<p>";
char *q6="</p>",*q7="<s>",*q8="</s>",*q9="<s_a>",*q10="</s_a>";
char *q11="<com>",*q12="</com>",*q13="<END>";
int kk=0;
        if(strstr(str_acm,p1)!=0)
        {
          pt1=strstr(str_acm,p1);
          pt1=pt1+14;
          kk=strlen(pt1);
          pt1[kk-16]='\0';
          itoa(counter_acm,p,10);
          fputs(q1,result_acm);
          fputs(p,result_acm);
          fputs(q2,result_acm);
          fputs(q3,result_acm);
char bb1[150]={NULL},bb2='/',bb3[150]={NULL};
```

```
int er1=0, er2=0, dl1=0, dl2=0;
        er1=strlen(pt1);
        while (pt1[er1]!=bb2)
        {
         bb1[er2]=pt1[er1];
         er1--;
         er2++;
        }
        dl1=strlen(bb1+1);
        dl2=dl1;
        for (int i=0;i<dl1;i++)
        {
         bb3[i]=bb1[dl2];
         dl2--;
        }

        fputs(bb3,result_acm);


        cout<<dl1<<" ";
        fputs(q4,result_acm);
        fputs(q5,result_acm);
        fputs(pt1,result_acm);
        fputs(q6,result_acm);
        fputs(q7,result_acm);
        fputs(q8,result_acm);
        fputs(q9,result_acm);
        fputs(q10,result_acm);
        fputs(q11,result_acm);
        fputs(q12,result_acm);
```

```
        fputs(q13,result_acm);
        fputs(p4,result_acm);
        counter_acm++;
      }
      if(strstr(str_acm,p2)!=0)
      {
      p5=strstr(str_acm,p2);
      p5=p5+14;
      kk=strlen(p5);
      p5[kk-16]='\0';
      char *we="|";
      fputs(L,result_comment);
      fputs(we,result_comment);
      fputs(p5,result_comment);
      fputs(p4,result_comment);
      }

      if(strstr(str_acm,p3)!=0)
      {
       p6=strstr(str_acm,p3);
       p6=p6+28;
       kk=strlen(p6);
       p6[kk-5]='\0';
       char *we="|";
       fputs(L,result_see_al);
       fputs(we,result_see_al);
       fputs(p6,result_see_al);
       fputs(p4,result_see_al);
      }
}
```

```
   }

   flag=feof(acm_class);//proverka konca fayla i esli konec to flag ne
                        //raven nulu
}
FILE *i;
i = fopen("r_i","wt");
char vv[200]={NULL}, bb[200]={NULL}, *dd1, *ll="|";
char *er1={NULL}, *e1="A";
int fl1=0, ma=0, gr1=0, gr2=0;
rewind(result_comment);
rewind(result_acm);
flag=0;
while (fl1==0)
{
        rewind(result_comment);
        flag=0;
        while (flag==0)
        {
          if (strcoll(vv,er1)==0){cout<<"gut";};
          fgets(vv,200,result_comment);
          gr1=strlen(vv);
          dd1=strstr(vv,ll);
          gr2=strlen(dd1);
          gr1=gr1-gr2;
          vv[gr1]='\0';
          fputs(vv,i);
          flag=feof(result_comment);
        }
 fputs(er1,i);
```

```
 fgets(bb,200,result_acm);
 er1=strstr(bb,e1);
 int ko=0;
 ko=strlen(er1);
 er1[ko-1]='\0';
 fl1=feof(result_acm);
}
  fclose(acm_class);
  fclose(result_acm);
  fclose(result_comment);
  fclose(result_see_al);
  return 0;
};

ACMTopic ()
{
   FILE *re;
   char *ptr_1, *ptr_2, *ptr_3, *ptr_4, str_ac[300]={NULL};
   char *uk_1="\n",STR[2000]={NULL};
   int lang_1=0, lang_2=0, flag=0;
   //otkrivaem fail promezhutochniy v kot pishem <ACMTopic>
if ((re = fopen("result.txt", "rt")) == NULL)
{
   fprintf(stderr, "Cannot open f_tags file\n");
   return 1;
}
while (flag==0)
{
   fgets(STR,300,re);
   STR=0;
```

```
    cout<<STR;
    flag=feof(re);
}
fclose(re);

};
```