ОСНОВНЫЕ АРХИТЕКТУРНЫЕ И СИСТЕМНЫЕ РЕШЕНИЯ В ТЕХНОЛОГИИ ИНТЕРИН

Я.И. Гулиев, к.т.н. (Исследовательский центр медицинской информатики ИПС им. А.К. Айламазяна РАН, г. Переславль-Залесский, pps@yag.botik.ru)

INTERIN TECHNOLOGY BASIC ARCHITECTURAL AND SYSTEM SOLUTIONS

Guliev Yadulla I., Ph.D. *(Research Centre for Medical Informatics, Organization of Russian Academy of Sciences Ailamazyan Program Systems Institute of RAS, Pereslavl-Zalessky, pps@yag.botik.ru)*

Abstract. In this paper, the author considers the basic architectural and system solutions of the Interin technology which aims at development of healthcare information systems. The author further presents key features and implementations of the Interin PROMIS healthcare information system based on the Interin technology.

Keywords: information system, medical information system, healthcare information system, information system architecture, object modeling, documents, information security.

В статье приводится описание основных архитектурных и системных решений технологии Интерин для создания медицинских информационных систем. Перечислены основные свойства медицинской информационной системы Интерин PROMIS, построенной на технологии Интерин, приведен список основных ее внедрений.

Ключевые слова: *информационная система, медицинская информационная система, архитектура информационной системы, объектное моделирование, документы, информационная безопасность.*

В 2009 году научным исследованиям в области медицинской информатики в Институте программных систем имени А.К. Айламазяна РАН исполняется 15 лет.

Подходы к созданию интегрированной *медицинской информационной системы* (МИС) авторы рассмотрели в [1], а в [2] кратко осветили технологию Интерин, которая стала главным результатом исследований и разработок.

Данная работа посвящена основным архитектурным и системным решениям технологии Интерин.

При решении первоначально поставленной задачи создания МИС комплексного *лечебно-профилактического учреждения* (ЛПУ) были развернуты исследования и разработки практически по всем направлениям медицинских информационных технологий.

1. Исследование основных проблем создания МИС и разработка:

- архитектурных решений для учрежденческих, региональных, персональных и мобильных систем;
- инструментальных средств и методик создания МИС.

2. Исследования и разработка общесистемных механизмов:

- поддержки информационной безопасности в МИС; сложных организационных структур в МИС; историчности информации в МИС; информационных стандартов и стандартов предметной области в МИС; принятия решений в МИС; телемедицинских технологий;
- организации пользовательского интерфейса и визуализации медицинской информации;
- идентификации в МИС, в том числе с использованием технологий идентификации (штрих-коды и смарт-карты),
- интеграции медицинских приборов и информационных систем,
- взаимодействия и интеграции разных информационных систем, в том числе в гетерогенной среде, обмена медицинскими данными в сети ЛПУ

3. Исследования и решение прикладных задач общесистемного характера:

- создание единой медицинской карты пациентов;
- поддержка экономики лечения, управленческого учета, решение проблем автоматизации материального учета;
- поддержка контроля качества лечения и безопасности пациентов в МИС.

4. Исследование экономической эффективности информационных технологий в медицине.

На рисунке 1 показана взаимосвязь перечисленных направлений исследований в рамках про-

Рис. 1. Концептуальная диаграмма проекта Интерин

екта создания МИС ЛПУ.

Особенности создания МИС

Основные проблемы, стоящие перед разработчиками МИС:

– большие объемы и разнообразие типов медицинской информации;

– недостаточная формализация (концептуализация и стандартизация) предметной области;

– постоянно расширяющаяся понятийная и концептуальная база предметной области;

– необходимость одновременной поддержки бумажной и безбумажной технологий работы;

– необходимость и актуальность поддержки единого информационного пространства (полные данные по каждому пациенту независимо от места оказания медицинской помощи).

Были исследованы и сформулированы требования к *интегрированным информационным системам* (ИИС) для сложных, плохо структурированных и трудноформализуемых предметных областей, таких как медицина, а именно:

– поддержка хранения и обработки фактографической информации с одновременным обеспечением основных функций существующих информационно-поисковых систем и систем обработки данных;

– хранение и обработка документов, построенных на фактографической информации, с обеспечением основных функций существующих систем документооборота;

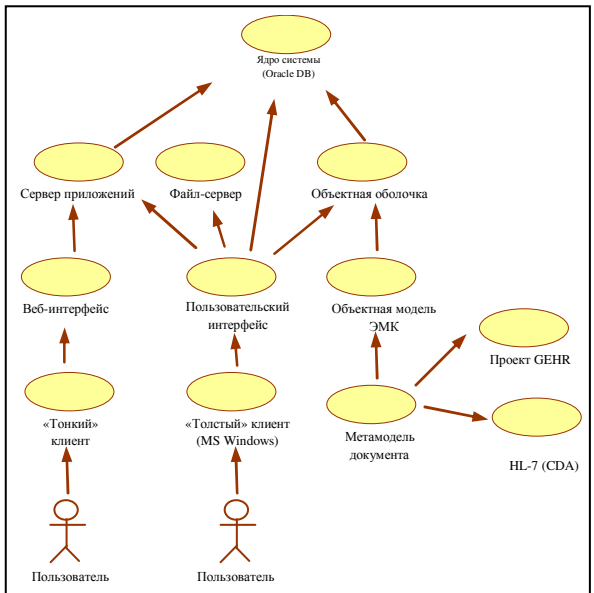


Рис. 2. Архитектурная диаграмма технологии Интерин

– моделирование бизнес-процессов на основе данных и документов, возможность определения «мягкого» и регулируемого регламента работы с фактографической информацией и документами;

– применение методов компонентного проектирования *информационных систем* (ИС) как для фактографических БД, так и для архивов документов и информационных потоков;

– разработка процедур компонентного изменения ИС при изменении бизнес-процедур и применяемых приложений;

– постоянная актуализация логической модели ИС для учета изменений в деятельности предприятий;

– возможность определения дополнительных регулируемых механизмов авторизации и прав доступа при работе с фактографической информацией и документами;

– построение интерфейса пользователя, интегрирующего в себе разные способы представления информации (в том числе манипулирование данными, документами) средства анализа информации, работа с информацией различного характера (графического, текстового, временных рядов).

Исследования показали, что возможностей существующих методологий и инструментальных средств для решения задач построения интегрированных МИС недостаточно, поэтому они были продолжены в направлении поиска как научно-технологических, так и методологических решений.

Были изучены и классифицированы основные устоявшиеся технологии ИС: фактографические системы (банки данных), системы документооборота (*Docflow*) и системы рабочих потоков (*Workflow*).

На основе проведенных исследований сделан вывод, что ввиду особенностей бизнес-процессов в медицинских учреждениях интегрированные МИС должны включать в себя элементы всех трех указанных типов ИС. В то же время каждая технология в отдельности либо не удовлетворяет все потребности МИС (фактографические системы, *Docflow*), либо плохо применима для их построения (*Workflow*).

Стало понятно, что составной частью общей методологии разработки интегрированных МИС может служить понятие «документ», которое сначала пытались выжить в эру АСУ, а потом внести в технологию ИС «как есть» в эру систем документооборота.

Кроме того, понятие «документ» как одна из основ построения МИС так же хорошо применимо для решений проблем поддержки стандартов представления медицинской информации, передачи медицинской информации и т.п.

Исследования в области использования концепции документа в архитектуре МИС привели к разработке механизма информационных объектов и архитектуры HL-X поддержки документов, которые составляют основу технологии Интерин.

Рисунок 2 иллюстрирует архитектуру ИС, построенной на основе механизма информационных

объектов и архитектуры HL-X.

Информационные объекты

Механизм *информационных объектов* (ИО) предназначен для централизованного представления метаданных и описания информационной модели предметной области. В его рамках стало возможным единообразное и системное решение вопросов доступа, отображения и обработки информации, пользовательского интерфейса.

С помощью этого механизма выделяется формализованный метауровень, назначение которого – описание структуры предметной области, включающей понятия и связи между ними (содержательная часть), а также способы манипулирования информацией (функциональная часть).

Механизм ИО представляет собой конструктор системы, позволяющий вводить новые объекты в ИС и определять их функциональность.

ИО – это совокупность семантически связанной информации, имеющей тип и уникальный ключ.

Тип ИО характеризуется названием и используется для классификации объектов. С типом связаны операция создания объекта и множество операций над объектами данного типа.

Для каждого типа могут быть определены **методы**, подобные методам классов в объектно-ориентированном программировании. Методы служат для доступа к содержимому объекта.

Определяются **составные** объекты**,** в которые входят другие объекты, и **атомарные**. Множество объектов, входящих в составной объект, называется **содержанием составного** объекта. В качестве содержания могут выступать другие составные объекты. Связи составного объекта с другими объектами, составляющими его содержание, называются **ссылками.**

Каждый объект имеет **статус**, который определяет состояние объекта.

Операцией над объектом называется неделимая последовательность действий, изменяющих его содержимое и/или состояние (статус). Операции выполняются при помощи процедур и/или программных модулей.

Кроме того, определяются понятия:

– **жизненный цикл** – последовательность операций над объектом; **летопись** объекта – последовательность записей о выделенных операциях жизненного цикла объекта;

– **исполнитель** – лицо, выполняющее операции над объектами;

– **пользователь** – пользователь ИС (любой пользователь является исполнителем, но некоторые исполнители могут не являться пользователями);

– **метапользователь** – абстрактный исполнитель, который задается отношением на множестве исполнителей, например, сотрудник терапевтического отделения, медсестра, член ВТЭК, лечащий врач, дежурный врач, директор центра и т.п.;

– **право** – возможность метапользователя выполнять данную операцию;

– **рабочий стол** – механизм, определяющий отношение владения объектами для пользователя;

– **ярлык** – краткое описание содержимого объекта; ярлыки используются для получения минимальной информации о содержимом объекта, для выбора конкретного объекта из списка объектов, при поиске информации в системе.

Подсистема поддержки жизненного цикла ИО предназначена для выполнения обобщенных **действий** над всеми типами объектов в ИС. Такими действиями являются создание, пересылка, подписание, актуализация и деактуализация, унифицированный доступ и поиск, вызов программных модулей и процедур для выполнения специализированных действий.

Каждый объект существует на двух уровнях:

– на нижнем уровне – БД/ИС – определена структура документа, здесь хранится информация, составляющая содержание документа;

– на верхнем уровне хранится информация о документе: уникальный глобальный ключ, ссылка на содержание объекта, информация о состоянии и местонахождении объекта.

Механизм ИО также служит основой для функционирования унифицированного интерфейса Рабочий стол пользователя.

Документы в МИС. Архитектура HL-X

В основе архитектуры HL-X лежит понятие документа HL-X. Документ HL-X – это свободно конструируемая по заданным правилам информационная структура из формализованных концептов предметной области.

В основе подхода лежит представление о необходимости поддержки ИС эволюционного процесса концептуализации предметной области.

Концептуально документ HL-X задается как множество моделей, раскрывающих его с различных точек зрения: понятийная модель, информационная модель (структурная модель), модель обработки данных документа, модель визуализации, функциональная модель документа, модель безопасности.

Понятийная модель представляет документ в виде структуры абстрактных и соответствующих им конкретных понятий. Она предоставляет следующие возможности:

1) определяет, какие понятия содержатся в документе и какими понятиями владеют БД или БЗ документов HL-X;

2) выполняет перевод понятийной модели на другие языки при наличии словарей;

3) является основой для добавления семантических связей на понятиях и использования технологий искусственного интеллекта для анализа

свободных документов (тематический поиск информации, поиск неизвестных знаний – *Data Mining* и т.п.);

4) фактически самодокументирует документ и может использоваться как умолчательная визуализация документа.

Информационная модель представляет документ HL-X в виде структуры из различных **ИО** (концептов предметной области). Здесь под ИО в широком смысле понимаются некоторые абстрактные связные элементы модели, аналогами которых в различных языках программирования являются объекты, записи, структуры, массивы и т.п. Наличие у документа структуры позволяет декомпозировать его на элементы и при необходимости манипулировать ими. Информационная модель служит основой для

– конструирования документа из элементарных ИО;

– конструирования документа на основе имеющихся документов с учетом накопленных знаний о контекстах использования того или иного объекта (понятия) путем полного или частичного заимствования уже существующих информационных моделей;

– разбора и обработки данных документа (например, для структурированного хранения в реляционной БД или для экспорта и импорта данных, для усвоения знания и пополнения БЗ непосредственно из экземпляров документов и их информационных моделей).

Модели обработки данных документа позволяют специфицировать обработку документа как единого целого в интересах конкретной ИС. Они определяют, какие именно данные нужно обрабатывать и как это делать. Модель обработки данных документа дополняет информационную модель документа конкретными, зависящими от ИС инструкциями. На базе модели выполняются все основные манипуляции с данными документа: разбор и хранение в БД, создание и редактирование документов, использование шаблонов и других документов в качестве источников данных.

Модели визуализации позволяют создавать различные визуальные представления данных документа. В основе визуальной модели лежит понятие визуальной компоненты как некоторой абстрактной элементарной формы представления данных. В различных конструкторах интерфейса аналогами визуальным компонентам служат компоненты визуальных палитр (таблицы, поля, панели, всевозможные элементы управления и т.п.). Основное назначение модели – создание абстрактного, не привязанного к конкретному программному языку или конструктору интерфейса описания представления данных документа в интерфейсе пользователя. Модель визуализации предназначена для

– визуализации документа в конкретной реализации пользовательского интерфейса со свободными документами;

– предоставления совместно с функциональной моделью интерфейса для манипуляции данными документа.

Функциональная модель документа определяет возможные манипуляции с данными документа и соответственно поддерживает определенные ограничения на данные (целостность данных). К области компетентности этой модели отнесем:

– транзакционность – обработку документа и всех входящих в него объектов как единого целого в рамках одной транзакции;

– дисциплину коллективной работы над документом (пользовательские блокировки);

– определение возможных манипуляций над документом на основе данных самого документа (учет статуса и т.п.);

– целостность данных (обязательные элементы данных, домены возможных значений, контроль типов и форматов данных и т.п.).

Модель безопасности документа определяет права доступа к элементам данных и права на манипуляцию данными. В качестве структурного элемента документа, на который можно будет установить права доступа, предлагается принять элементарное понятие из понятийной модели. В информационной модели этому элементу доступа соответствует атрибут ИО или сам объект, а в визуальной модели – отдельное поле визуализации или ввода. Более элементарных целостных единиц данных, к которым можно было бы отнести доступ, в документе нет. На уровне структуры документа предложена самая сильная модель безопасности.

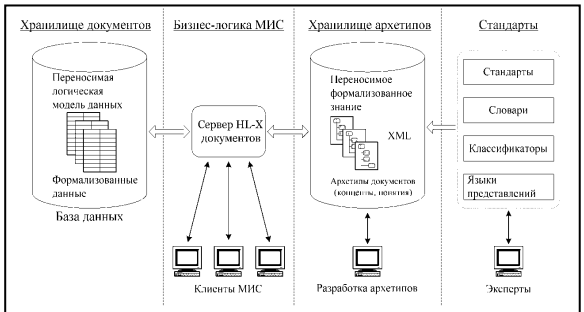


Рис. 3. Процесс концептуализации предметной области в архитектуре HL-X

Документ HL-X рассматривается как единство всех этих моделей. Документ HL-X – это и сами данные, и организация данных в структуру, и знания, заключенные в структуре данных, и правила манипулирования данными, включая права доступа, и визуальное представление данных, и инструкции по обработке этих данных в ИС. Главное достоинство документа HL-X – высокий уровень абстракции этих моделей, делающих его независимым от конкретной ИС, БД, технологических

средств разработки и средств доставки документа:

– документ HL-X может быть легко доставлен из одной ИС в другую с помощью любых каналов связи и средств доставки;

– документ HL-X может быть понят и интерпретирован пользователем без использования модели визуализации на основе своей понятийной модели, благодаря которой он фактически является самодокументируемым;

– модель обработки данных документа в каждой ИС может быть своя в зависимости от целей

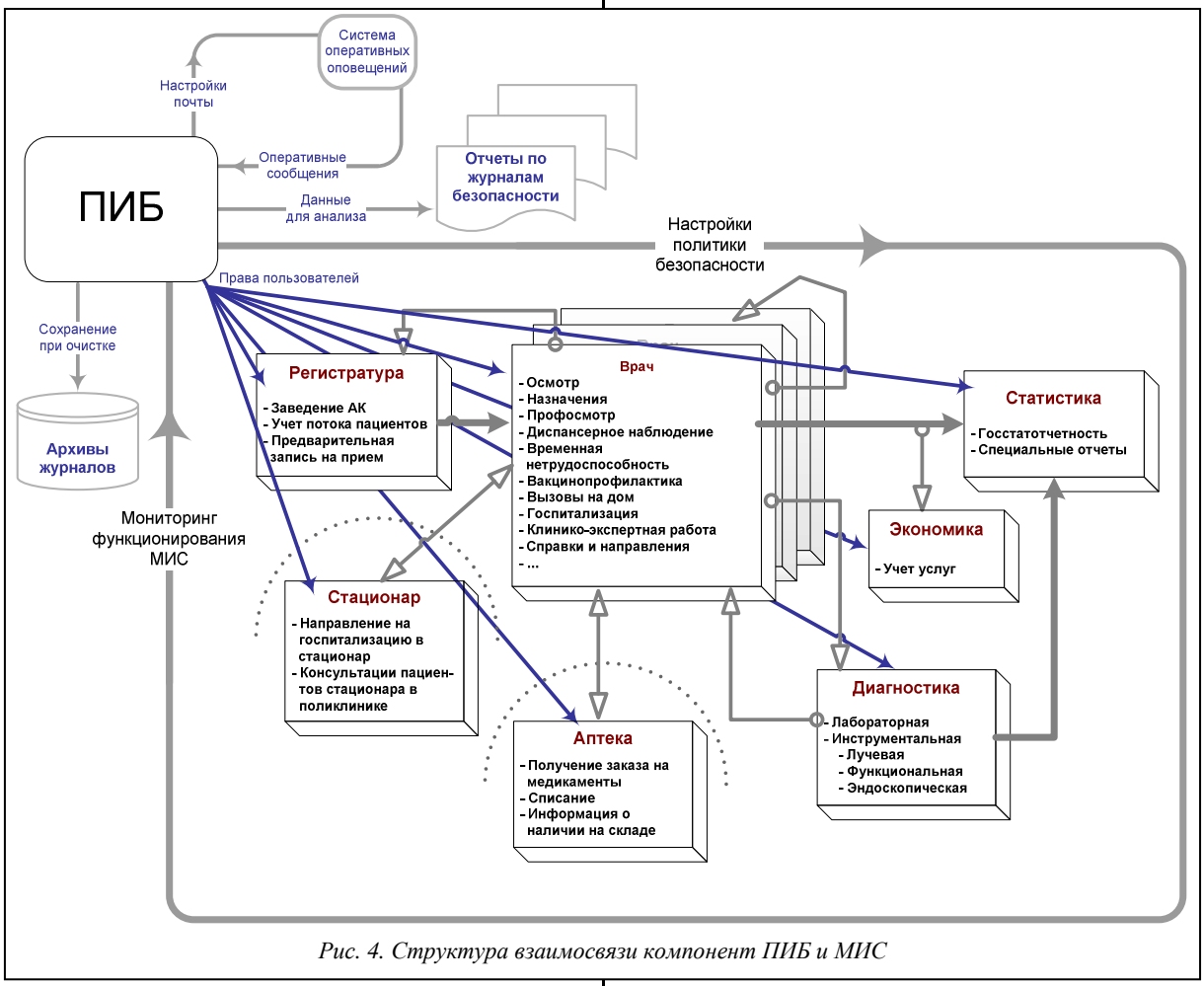


Рис. 4. Структура взаимосвязи компонент ПИБ и МИС

использования этих данных системой (степени их интеграции системой);

– моделей визуализации может быть много в соответствии с желаемыми представлениями данных.

Преимущества документов HL-X:

– свобода от конкретных, подверженных постоянному развитию и изменению технологий реализации ИС;

– самоценность документа HL-X как носителя модели информации, описывающей определенную предметную область, как носителя знания об этой предметной области;

– продление времени жизни модели документа в силу ее инвариантности по отношению к реализации документа HL-X в ИС;

– гибкость при реализации ИС на базе документов HL-X – возможность произвольного выбора БД (реляционной, постреляционной или объектной); возможность выбора различных технологий реализации интерфейса документа HL-X. В качестве языков реализации прототипа архитектуры были опробованы *Pascal*, PL/SQL, *JavaScript*, *Java*, HTML, XML;

– восприимчивость к инновациям в информационных технологиях; возможность быстрого переноса накопленных моделей документов HL-X в новую информационную среду без их существенных переделок и развертывания ИС на базе новых технологий.

Еще одна основная концептуальная идея архитектуры HL-X – **введение процесса концептуализации предметной области непосредственно в саму ИС** – как ответ на трудности, связанные с недостаточной формализацией и динамичностью предметной области (рис. 3).

Информационная безопасность

На государственном уровне все большее значение придается обеспечению информационной

безопасности персональных и медицинских данных. Исследования направлены на создание эффективной *подсистемы информационной безопасности* (ПИБ), обеспечивающей разделение доступа к данным в зависимости от полномочий

– технические меры защиты информации обеспечены техническими средствами защиты (описание необходимых для использования средств защиты и их настроек приводится в эксплуатационной документации МИС);

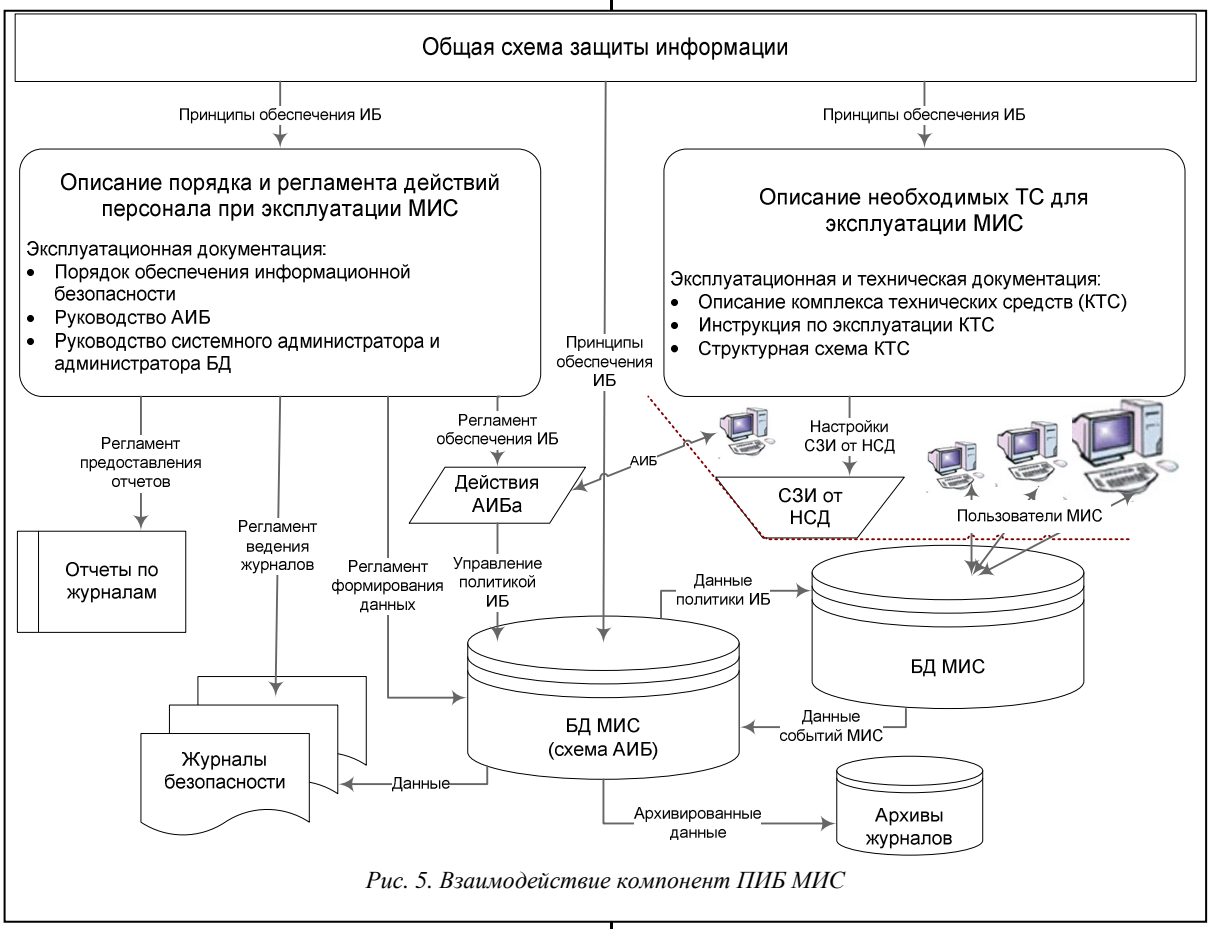


Рис. 5. Взаимодействие компонент ПИБ МИС

пользователей и выделение/мониторинг прав и ролей, предоставляющей возможности настройки политики безопасности, а также гарантирующей сохранность данных и обеспечивающей контроль над функционированием системы и над действиями пользователей.

Построение адекватной схемы защиты данных в конкретном случае является поиском компромисса между конфиденциальностью, целостностью и доступностью данных с учетом специфики работы МИС ЛПУ как системы массового обслуживания.

В настоящее время ПИБ представляет собой завершенное решение в рамках типовой МИС семейства Интерин.

ПИБ основывается на модели угроз и модели нарушителя, разработанной для типового ЛПУ.

ПИБ МИС представляет собой комплекс организационных, технологических, технических и программных мер и средств защиты информации:

– программные меры защиты информации реализованы программными компонентами и механизмами ПИБ;

– организационные меры защиты информации обеспечены выполнением персоналом порядков и регламентов для различных действий при эксплуатации МИС (описание необходимых организационных мер и регламентов работы приводится в эксплуатационной документации МИС);

– управление полномочиями пользователей, настройка политики безопасности, а также оперативный и ретроспективный контроль за действиями пользователей МИС и потенциально опасными событиями обеспечиваются выделенным рабочим местом – АРМ администратора информационной безопасности. Для независимости функционирования от МИС программное обеспечение рабочего места администратора строится на системных таблицах БД и для выполнения своих функций использует встроенные механизмы СУБД.

Схема на рисунке 5 иллюстрирует взаимодействие технических, организационных и программных компонент ПИБ МИС.

Теоретические исследования и практические разработки обусловили создание технологии построения МИС, включающей комплекс инстру-

ментальных средств, технологических решений и методик создания интегрированных ИС ЛПУ, которая впоследствии получила название *технология Интерин*. МИС семейства Интерин представляет собой интегрированную информационную и функциональную среду и обеспечивает информационную поддержку всех служб ЛПУ.

Интерин PROMIS

Интерин PROMIS – типовой вариант МИС, созданной на основе технологии Интерин. Свойства системы Интерин PROMIS позволяют использовать ее практически в любом ЛПУ. Внедрение системы предполагает установку типового варианта МИС, ее настройку и адаптацию к специфике ЛПУ, настройку рабочих мест пользователей, обучение персонала и последующее сопровождение работы МИС. Система Интерин PROMIS имеет Свидетельство Министерства здравоохранения РФ о пригодности к использованию в организациях здравоохранения России.

Основные свойства и возможности МИС системы Интерин PROMIS следующие.

- Полная номенклатура АРМ медицинского персонала разных специальностей. Масштабируемость системы и возможность настройки рабочих мест под конкретные задачи пользователя.
- Интеграция информационных потоков, обеспечивающая актуальность, целостность и непротиворечивость информации.
- Концентрация информации вокруг пациента. Возможность просмотра и анализа информации о пациенте в различных представлениях, сгруппированной тем или иным образом.
- Единое пространство услуг. Вводит систему формальных понятий, к которым можно привести весь спектр действий, выполняемых медицинским персоналом.
- Представление динамики медицинской информации, мониторинг лечебно-диагностического процесса.
- Автоматизация оформления документации: множественное использование данных без дублирования, автозаполнение, использование шаблонов документов, ввод данных в специализированных формах без форматирования с последующим автоматическим формированием документов, планирование технологической лечебно-диагностической цепочки.
- Автогенерация статистических отчетов, динамические подборки документов на Рабочем столе и сводки за период или на дату, формируемые для печати.
- Редактируемые справочники для наполнения предметной информацией, позволяющие настраивать и модифицировать МИС при внедрении или изменении бизнес-процессов.
- Использование новейших разработок в области представления и передачи медицинских данных, дающие возможность взаимодействовать с программными продуктами сторонних разработчиков и с ИС других медицинских учреждений.
- Применение элементов телемедицины, обеспечивающее снижение стоимости лечебного процесса, преодоление профессиональной изоляции, улучшение качества лечения.
- Система управления визуальной информацией для диагностических и отчетных целей, включая удаленный доступ к хранилищам данных.

Функциональные подсистемы МИС Интерин PROMIS: клиническая подсистема, амбулаторно-поликлиническая подсистема, аналитическая подсистема, экономическая подсистема, лабораторная подсистема, регистратура, стоматология, хранение и передача графических данных, аптека, лечебное питание, отдел кадров, удаленный доступ к Рабочему столу, администрирование МИС.

Внедрения

Первая прикладная МИС, созданная с использованием технологии Интерин, была установлена в Медицинском центре Банка России в 1996 году, с этого времени эксплуатируется и развивается. В настоящее время все подразделения и службы Медицинского центра Банка России, включая поликлинику, стационар и диагностический центр, оснащены МИС, основанной на технологии Интерин, и работают в едином информационном пространстве. Система насчитывает около 1 300 пользователей.

Впоследствии ИС на базе технологии Интерин были оснащены и некоторые другие крупные ЛПУ:

– *ИС управления* (ИСУ) республиканской больницы № 1 Национального центра медицины Министерства здравоохранения Республики Саха (Якутия) «КИС НЦМ»,

– ИСУ ЦКБ № 1 ОАО «Российские железные дороги»,

– МИС «Амбулатория», основанная на технологии Интерин, амбулаторно-поликлинических учреждений ГУ Банка России по Вологодской, Костромской, Омской, Нижегородской областям, Республике Марий Эл и Приморскому краю,

– ИСУ ФГУ «Российский кардиологический научно-производственный комплекс МЗ РФ»,

– ИСУ ФГУ «Клиническая больница» Управления делами Президента РФ,

– ИСУ ФГУ «Поликлиника № 3» Управления делами Президента РФ.

В данной работе приведены описание и взаимосвязь основных архитектурных и системных решений технологии Интерин, эффективность которых доказана успешной эксплуатацией многочисленных ИС, построенных на этой технологии.

Основные результаты исследований и разработок изложены в многочисленных публикациях, обновляемый список которых можно найти на

сайте (*www.interin.ru*) Исследовательского центра медицинской информатики ИПС РАН.

Литература

1. Интегрированная распределенная информационная система лечебного учреждения (ИНТЕРИН) / Я.И. Гулиев [и др.] // Программные продукты и системы. 1997. № 3.

2. Гулиев Я.И. Интерин-технология для создания медицинских информационных систем // Программные продукты и системы. 2008. № 2.

3. Айламазян А.К., Гулиев Я.И. Данные, документы и архитектура медицинских информационных систем: тез. докл. Междунар. форума (Информатизация процессов охраны здоровья населения-2001). М., 2001. С. 141–142.

4. Малых В.Л., Пименов С.П., Хаткевич М.И. Объектно-реляционный подход к созданию больших информационных систем: тр. Междунар. конф. / ИПС РАН, Переславль-Залесский; под ред. А.К. Айламазяна. М.: Наука. Физматлит, 1999. С. 177. (Программные системы: Теоретические основы и приложения).

5. Гулиев Я.И., Малых В.Л. Архитектура HL-X: тр. Междунар. конф. / ИПС РАН, Переславль-Залесский; под ред. С.М. Абрамова. В 2 т. 2004: М.: Физматлит. Т. 2. С. 147. (Программные системы: теория и приложения).

6. Особенности решения проблем информационной безопасности в медицинских информационных системах / Г.И. Назаренко [и др.] // Врач и информационные технологии. 2007. № 4. С. 39–43.

7. Назаренко Г.И., Гулиев Я.И., Ермаков Д.Е. Медицинские информационные системы: теория и практика; под ред. Г.И. Назаренко, Г.С. Осипова. М.: Физматлит, 2005. С. 320.

8. Назаренко Г.И., Гулиев Я.И. Информационные системы в управлении лечебно-профилактическим учреждением // Врач и информационные технологии. 2006. № 4. С. 64–67.

9. Гулиев Я.И. Медицинская информатика в ИПС РАН: тр. Междунар. конф. / ИПС РАН, Переславль-Залесский; под ред. С.М. Абрамова. В 2 т. 2004: М.: Физматлит. Т. 1. С. 53. (Программные системы: теория и приложения).

ПРИМЕР РЕАЛИЗАЦИИ РЕГИОНАЛЬНОЙ МЕДИЦИНСКОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ С ЦЕНТРАЛИЗОВАННОЙ АРХИТЕКТУРОЙ

Я.И. Гулиев, к.т.н.; С.И. Комаров, к.т.н.; И.Ф. Казаков; Д.Р. Магсумов
(Исследовательский центр медицинской информатики ИПС им. А.К. Айламазяна РАН, г. Переславль-Залесский, aifa@yag.botik.ru, ksi@interin.ru, kazakov@interin.ru, dimam@interin.ru)

A REGIONAL MEDICAL INFORMATION SYSTEM WITH CENTRALIZED ARCHITECTURE IMPLEMENTATION EXAMPLE

Guliev Yadulla I., Ph.D.; Komarov Sergey I., Ph.D.; Kazakov Ilia F., Magsumov Dmitriy R.
(Research Centre for Medical Informatics, Organization of Russian Academy of Sciences Ailamazyan Program Systems Institute of RAS, Pereslavl-Zalessky, aifa@yag.botik.ru, ksi@interin.ru, kazakov@interin.ru, dimam@interin.ru)

Abstract. The paper presents the results of analysis in the field of regional medical information systems implementation problems and solutions. The Interin DLO regional medical information system is presented. This system is designed for regional common information space management within the framework of the government program of preferential medicinal maintenance.

Keywords: information system, medical information system, information system with centralized architecture, preferential medicinal maintenance, medical information system architecture.

В статье исследуются проблемы и решения в области построения региональных медицинских информационных систем. Приводится описание региональной системы Интерин ДЛО, предназначенной для организации регионального единого информационного пространства в части дополнительного лекарственного обеспечения.

Ключевые слова: *информационная система, медицинская информационная система, региональная медицинская информационная система, дополнительное лекарственное обеспечение, централизованная архитектура, архитектура медицинских информационных систем.*

Современный российский рынок *медицинских информационных систем* (МИС) предлагает достаточно большое количество решений, основанных на последних достижениях информационных технологий. Представленные на рынке МИС в основном решают задачи комплексной информатизации отдельных *лечебно-профилактических учреждений* (ЛПУ), в то время как возрастает потребность в интегрированных МИС масштаба региона.

Основной целью создания региональных интегрированных МИС является информационная поддержка организаций здравоохранения на основе внедрения новых информационно-коммуникационных технологий, позволяющих более эффективно управлять здравоохранением, повышать качество и доступность медицинского обслуживания за счет увеличения объема и качества доступной информации; существенного улучшения информационного обмена между всеми звеньями системы здравоохранения, включая все уровни управления; обеспечения мониторинга состояния здоровья населения и реализации национального проекта «Здоровье».

Создание столь масштабных систем, в которых должно быть учтено множество параметров,

включая потребности и специфику бизнес-процессов всех объектов информатизации, предполагает использование особых технологий разработки и внедрения *региональных информационных систем* (РИС).

Остановимся на основных требованиях к различным аспектам РИС.

Информационное обеспечение должно базироваться на единой системе стандартов, использовать единую систему классификации и кодирования, образовывать целостный и непротиворечивый набор данных.

Требования к эргономике и технической эстетике: модули *информационной системы* (ИС) должны иметь унифицированный интерфейс пользователя; необходимо предусмотреть возможность предоставления хранимой в ИС информации в виде твердой копии.

Требования к информационной безопасности: правовую основу обеспечения информационной безопасности должны составлять правовые акты Российской Федерации и нормативные документы регионального и муниципального уровней.

РИС должна обеспечивать:

- защиту информации от несанкционированной модификации и разрушения на всех этапах ее обработки, хранения и передачи;
- разграничение прав пользователей и обслуживающего персонала при доступе к информационным ресурсам ИС, а также при хранении и предоставлении конфиденциальной информации;
- возможность доказательства неправомочности действий пользователей и обслуживающего персонала ИС;
- защиту информации от несанкционированного доступа средствами проверки полномочий пользователей и обслуживающего персонала на использование информационных ресурсов ИС (возможность несанкционированного изменения или уничтожения этой информации, как и несанкционированное получение, изменение или уничтожение информации третьими лицами, должны быть исключены);
- защиту от несанкционированной модификации программного обеспечения;
- защиту информации от случайных разрушений;
- дублирование информации путем создания резервных копий.

Для реализации Федерального закона об обеспечении доступности качественной медицинской и лекарственной помощи льготным категориям населения РФ была запущена программа *дополнительного лекарственного обеспечения* (ДЛО) льготных категорий населения.

В основе организации системы ДЛО лежит персонифицированный учет граждан, имеющих право на государственную социальную помощь. На его базе строятся финансирование системы, учет выписки рецептов гражданам и расчеты с фармацевтическими организациями за отпущенные лекарственные средства по льготным рецептам. Все это требует соответствующего информационного обеспечения как внутренних процессов каждого участника системы, так и процессов взаимодействия между участниками на основе единых подходов и унификации. Согласование и стыковка всех информационных массивов данных, формируемых и используемых большим количеством участников системы ДЛО, – сложная технологическая проблема. Поэтому функционирование системы ДЛО предъявляет повышенные требования к информационному обеспечению каждого участника и требует эффективной организации и согласования процессов информационного обмена данными между ними.

Задача информационного обеспечения участников программы ДЛО представляет собой массив подзадач разного уровня, сложность реализации которых обусловлена общей комплексностью взаимодействий участников программы. Для создания единого информационного пространства в рамках программы ДЛО прежде всего необходима единая система классификации и кодирования информации, ориентированной на всех участников системы. Решение данной проблемы предлагают справочники федерального и территориального уровней, однако эффективное использование таких справочников в качестве элементов программы ДЛО возможно только при условии их постоянной актуализации.

Важным является также принятие единых стандартов информационного взаимодействия участников программы. Первые стандарты информационного обеспечения программы ДЛО, или, скорее, их прототипы, были приняты в декабре 2004 года. На этапе развития программы стандарты существенно модифицировались, однако этого оказалось недостаточно для ее нормального функционирования. С начала 2006 года в ДЛО приняты единые стандарты информационного обеспечения на основе комплексного подхода: введены штрих-кодирование рецептов, XML-форматы обмена данными, проводится автоматизация *центров обработки данных* (ЦОД). Следует заметить, что по мере развития программы ДЛО и расширения состава ее участников процесс перехода на новые стандарты информационного обеспечения становится все более болезненным. Для редукции рисков, связанных со сменой стандартов или вводом новых категорий стандартов, необходимы более точное прогнозирование изменений, синхронизация действий всех участников программы, что означает смещение приоритетов программы ДЛО в сферу долгосрочного планирования изменений.

Описание ИС «Интерин ДЛО»

В Исследовательском центре медицинской информатики Института программных систем РАН разработана типовая версия ИС «Интерин ДЛО». В основу системы положен опыт разработки, внедрения и использования МИС.

Объектами информатизации являются учреждения, участвующие в системе бесплатного и

- способность ИС функционировать в условиях гетерогенной информационной среды;
- возможность непрерывной модернизации программных модулей, входящих в состав ИС;
- преемственность и обратная совместимость версий системы.

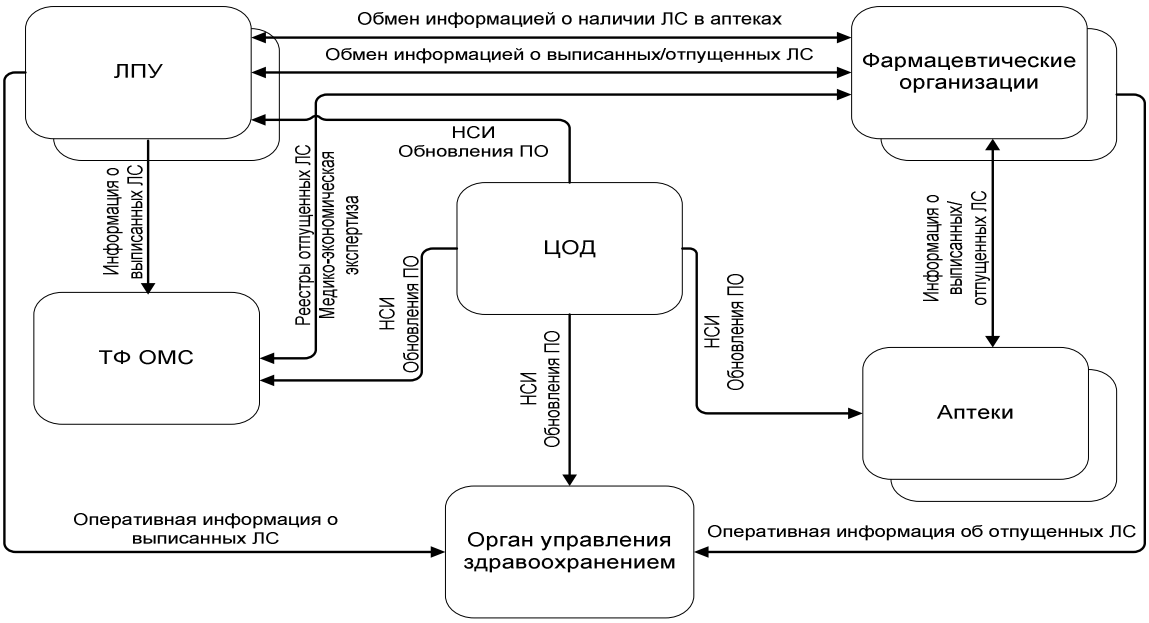


Схема ИС ДЛО, построенной на основе ЦОД

льготного лекарственного обеспечения отдельных категорий граждан в соответствии с федеральным и региональным законодательством, а именно: ЛПУ, аптечные учреждения, *территориальный фонд обязательного медицинского страхования* (ТФ ОМС), департамент социальной защиты населения, органы управления здравоохранением.

В результате были сформулированы основные критерии, которым должна удовлетворять ИС ДЛО:

- оперативный доступ к полной статистической, медицинской и аналитической информации по выбранным параметрам;
- обеспечение механизма консультации специалистов и руководителей учреждений-участников ДЛО;
- обеспечение надежности и отказоустойчивости ИС;
- обеспечение комплекса организационно-технических мероприятий по информационной безопасности;
- сохранение характеристик стабильности при увеличении числа пользователей;
- возможность адаптации ИС к территориальной специфике бизнес-процессов того или иного субъекта Федерации;
- гибкость и способность к эволюции БД и ИС;

Архитектура ИС «Интерин ДЛО»

Перечисленным требованиям в наибольшей степени удовлетворяет решение на основе единого ЦОД. При этом имеется в виду не только техническая база, но и комплексный подход к аппаратной и программной составляющим – централизованный программно-аппаратный комплекс обработки данных, основными характеристиками которого являются:

- *web*-сервисы, доступные преимущественно в круглосуточном режиме, – технологическая основа для организации взаимодействия подсистем участников с ЦОД;
- реализация в многозвенной распределенной компонентной архитектуре;
- основной язык обмена информацией – XML;
- наличие сервисов, поддерживающих интеграцию с внешними ИС;
- соответствие форматов обмена данными нормативным документам, регламентирующим процессы информационного обмена в ДЛО.

Данные характеристики предполагают, что при интеграции региональной и российской систем на верхнем уровне может находиться орган исполнительной власти РФ.

На рисунке приведена укрупненная схема ИС

ДЛО, построенной на основе ЦОД.

Очевидно, что при такой архитектуре ИС возрастают требования к сетям передачи данных, однако при современном уровне развития коммуникационных технологий это не столь критично.

Из предпочтительных параметров данной архитектуры необходимо отметить, что наличие ЦОД сокращает объемы передаваемой информации за счет уменьшения количества посредников в процессе обмена и позволяет работать в *online*-режиме, что приводит к минимизации возможности искажения и потери данных, уменьшению затрат на обмен информацией, увеличению скорости обмена и оперативности доступа к актуальной информации всех участников.

Основным недостатком централизованной ИТ-архитектуры является низкая отказоустойчивость ИС, построенной на ее основе. Однако современный уровень развития программных и аппаратных платформ позволяет уменьшить количество сбоев за счет дублирования критически важных компонентов, применения средств резервного копирования, использования серверов приложений и СУБД, построенных на RISC-технологиях.

Стоит также отметить высокую масштабируемость ЦОД, что позволяет наращивать объемы хранимой информации без потерь в вычислительных возможностях при сохранении максимального уровня производительности и минимального времени отклика.

Кроме того, в ЦОД можно обеспечить максимально доступный относительно программно-аппаратной среды функционирования уровень информационной безопасности за счет использования централизованных средств хранения и архивации данных, единой точки контроля над доступом к приложениям и данным.

В качестве основы для организации АРМ пользователей ИС «Интерин ДЛО» была выбрана широко распространенная технология тонкого клиента, при которой работа пользователей осуществляется в терминальном режиме, что позволяет избежать потери информации при авариях каналов связи.

При проектировании системы выявлено, что часть функционала отдельных подсистем можно объединить в специальные модули и реализовать в виде общесистемных механизмов, используемых большинством АРМ ИС «Интерин ДЛО». Примерами таких общесистемных механизмов являются подсистемы ретроспективного хранения данных, ведения *нормативно-справочной информации* (НСИ) в единой технологии, импорта/экспорта данных, аудита и информационной безопасности.

Функциональным назначением ИС «Интерин ДЛО», построенной на изложенных принципах, является автоматизация процессов информационного взаимодействия участников ДЛО, в том числе ТФ ОМС, органов управления здравоохранением, ЛПУ, аптек и аптечных пунктов, ЦОД.

Основная цель создания ИС «Интерин ДЛО» – информатизация и повышение качества и эффективности работы всех организаций-участников системы ДЛО, а именно:

- повышение уровня актуальности доступной информации;
- устранение возможности появления нестыковок в информации за счет использования единого центра хранения и обработки данных;
- повышение скорости и качества обновления регистров и справочников;
- рационализация использования бюджетных средств, затрачиваемых на оплату лекарственных средств и изделий медицинского назначения, отпускаемых отдельным категориям граждан;
- снижение общего уровня затрат на сопровождение ПО.

В состав ИС «Интерин ДЛО» входят следующие основные АРМ пользователей.

АРМ «Администратор системы», с помощью которого осуществляется импортирование регламентированной НСИ; регионального сегмента федерального регистра лиц, имеющих право на набор социальных услуг (НСУ); просмотр и поиск регламентированной НСИ, а также по региональному сегменту федерального регистра лиц, имеющих право на НСУ; заведение и редактирование учетных записей пользователей ИС; управление полномочиями и ролями пользователей ИС; ведение журналов аудита по работе зарегистрированных пользователей в ИС и функционирования ИС.

АРМ «Администратор ТФ ОМС» и АРМ «Эксперт ТФ ОМС» позволяют осуществлять ведение, просмотр и поиск регламентированной НСИ; просмотр и поиск по региональному сегменту федерального регистра лиц, имеющих право на НСУ; просмотр оперативной информации о выписанных рецептах; импортирование реестров рецептов, отпущенных льготным категориям граждан; проведение медико-экономической экспертизы; формирование регламентированных документов по результатам проведения медико-экономической экспертизы, а также отчетности по ДЛО.

АРМ «Администратор ЛПУ» и АРМ «Сотрудник ЛПУ» производят ведение, просмотр и поиск регламентированной НСИ; просмотр и поиск по региональному сегменту федерального регистра лиц, имеющих право на НСУ; регистрацию данных о выписываемом рецепте; автоматизированную печать выписанных рецептов со впечатыванием штрих-кода, формируемого в соответствии с приказом ФФОМС № 38 от 21.03.2006; ведение реестров выписанных рецептов и их журналов; формирование регламентированной отчетности по ДЛО.

АРМ «Администратор АУ» и АРМ «Сотрудник АУ» осуществляют ведение, просмотр и

поиск регламентированной НСИ; просмотр и поиск по региональному сегменту федерального регистра лиц, имеющих право на НСУ; регистрацию данных об отпущенных лекарственных средствах по льготным рецептам; ведение реестров отпущенных рецептов и их журналов; формирование регламентированной отчетности по ДЛО.

АРМ «Сотрудник Департамента здравоохранения» выполняет ведение, просмотр и поиск регламентированной НСИ; просмотр и поиск по региональному сегменту федерального регистра лиц, имеющих право на НСУ; просмотр оперативной информации о выписанных и отпущенных рецептах; формирование регламентированных отчетных и аналитических материалов по ДЛО.

В составе ИС в одном из регионов, где внедрена система «Интерин ДЛО», функционировали около 330 рабочих мест с охватом всех перечисленных выше типов в 43 учреждениях (данные 2008 года).

В результате внедрения ИС «Интерин ДЛО» в субъекте РФ становится возможным консолидировать показатели субъектов системы согласно разработанным формам в соответствии с нормативными документами по основным аспектам деятельности участников ДЛО (медицинским, финансово-экономическим и потребительским), а также повысить качество и доступность медицинских услуг. Кроме того, ИС позволяет проводить процесс информатизации здравоохранения и системы ОМС региона на основе унифицированного и гибкого системного подхода в соответствии с определяющими нормативно-законодательными актами; объективно оценивать эффективность проведения мероприятий по реформированию здравоохранения и создавать единое региональное информационное пространство в сфере ДЛО.

Литература

1. Интегрированная распределенная информационная система лечебного учреждения (ИНТЕРИН) / Я.И. Гулиев [и др.] // Программные продукты и системы. 1997. № 3.

2. Гулиев Я.И., Комаров С.И. Интегрированная распределенная информационная система крупного лечебно-диагностического учреждения: тез. докл. IV междунар. форума (Стратегии здоровья: информационные технологии и интеллектуальное обеспечение медицины-97). М., 1997.

3. Белышев А.Г., Гулиев Я.И., Морозов В.Ю. Построение медицинских систем с использованием объектных технологий // Программные системы: Теоретические основы и приложения / Под ред. А.К. Айлмазяна. М.: Наука. Физматлит, 1999. С. 169.

4. Малых В.Л., Пименов С.П., Хаткевич М.И. Объектно-реляционный подход к созданию больших информационных систем: тр. Междунар. конф. / ИПС РАН, Переславль-Залесский; под ред. А.К. Айламазяна. М.: Наука. Физматлит, 1999. С. 177. (Программные системы: Теоретические основы и приложения).

5. Айламазян А.К., Гулиев Я.И. Разработка информационных систем лечебно-профилактических учреждений: проблемы и решения: тез. докл. Междунар. форума. М.: Наука. Физматлит, 2000.

ИССЛЕДОВАНИЕ ТИПОВЫХ ПРОЦЕССОВ ИНТЕГРАЦИИ В МЕДИЦИНСКИХ ИНФОРМАЦИОННЫХ СИСТЕМАХ

Ю.В. Козадой (Исследовательский центр медицинской информатики ИПС им. А.К. Айламазяна РАН, г. Переславль-Залесский, watergad@interin.ru)

RESEARCH OF COMMON INTEGRATION PROCESSES IN HEALTHCARE INFORMATION SYSTEMS

Kozadoy Yuriy V. *(Research Centre for Medical Informatics, Organization of Russian Academy of Sciences Ailamazyan Program Systems Institute of RAS, Pereslavl-Zalessky, watergad@interin.ru)*

Abstract. The paper contains the analysis of integration problems for healthcare information systems. The author described various approaches to solve these problems. The paper includes several classifications according to different parameters and examines advantages and disadvantages of integration models. The result of the research is the most effective schema for the common integration process.

On basis of the suggested model several common integration mechanisms of the Interin PROMIS healthcare information system and information systems of other developers were implemented. These mechanisms proved the effectiveness of chosen solutions.

Keywords: integration, healthcare information systems, healthcare information systems architecture.

В статье проведен системный анализ проблемы интеграции в медицинских информационных системах, рассмотрены возможные подходы к их решению. Введен ряд классификаций по различным параметрам и рассмотрены преимущества и недостатки различных моделей интеграции. Результатом работы стало выделение наиболее эффективной схемы для типового процесса интеграции.

На основе предложенной модели был реализован ряд механизмов интеграции типовой медицинской информационной системы Интерин PROMIS с информационными системами различных разработчиков, потдвердивших эффективность выбранных решений.

Ключевые слова: *интеграция, медицинские информационные системы, архитектура медицинских информационных систем.*

При информатизации медицинского учреждения зачастую встает задача интеграции – объеди-

нения данных, получаемых от используемых *информационных систем* (ИС) различной специализации и различных разработчиков. Проблема интеграции тем острее, чем более жесткие требования (по быстроте взаимодействия, надежности, объемам передаваемых данных и т.п.) предъявляются к ее механизму.

Занимаясь разработкой ИС для медицинских учреждений, специалисты постоянно сталкиваются с задачей совместного использования данных несколькими ИС. Подходы к решению этой задачи могут быть разными.

По результатам обобщения разработки и эксплуатации интеграционных механизмов, объединяющих различные системы как медицинской, так и немедицинской направленности, типовой процесс интеграции *медицинской информационной системы* (МИС) можно свести к следующей схеме: определение типа интеграции по ряду классификаций, выбор транспортного агента и формата передачи данных, реализация механизма интеграции, реализация обработки ошибок в данных, утверждение регламента интеграции и разделение ответственности.

Рассмотрим подробнее указанные этапы, а также некоторые особенности процессов интеграции.

Классификация интеграций

По типам взаимодействия предлагается выделить несколько классификаций интеграционных процессов по различным признакам.

Односторонняя и двусторонняя интеграции. Одним из определяющих признаков интеграции является направление передачи данных.

Наиболее простой тип – односторонняя интеграция, когда данные из одной системы выгружаются в другую (загружаются из другой). В этом случае интеграция происходит лишь на уровне данных и может, говоря формально, рассматриваться только для одной системы, например, в случае, когда интегрируемая система целиком формирует файл БД для интегрирующей системы, после чего в интегрирующей системе производится подмена файла, сама же она процесса интеграции не замечает. Примером такой интеграции может служить включение в процесс информационного обмена некой системы путем автоматизированного искусственного создания среды, в которой она работает.

Если необходимо производить интерактивный обмен данными, осуществлять передачу и контролировать ее результаты, используется двусторонняя интеграция – когда активными участниками процесса выступают обе системы. В этом случае при обмене данными одна система может реагировать на события, произошедшие (или вызванные) в другой системе. Так, например, интеграция на основе обмена сообщениями с подтверждением принятия сообщения уже является двусторонней. Двусторонняя интеграция характерна и для случаев, когда системы оперируют одной областью данных (например списком пациентов), причем и в случае, когда список ведется только в одной из систем, и при синхронизации списков обоих систем в реальном времени. В качестве примера можно рассмотреть интеграцию с диагностическим или лабораторным оборудованием (либо лабораторной системой), когда из МИС поступает направление на проведение исследования, а в ответ направляется результат исследования.

Возможно и большее количество активных сторон в интеграции двух и более ИС.

Для эффективного контроля результатов интеграции и работы ее механизма предлагается использовать двусторонний тип интеграции, поскольку односторонние процессы менее гибки и не позволяют успешно решать возникающие проблемы. Однако односторонняя интеграция может быть исключительно полезной при включении в процессы информационного обмена эксплуатирующихся в медицинском учреждении замкнутых систем, модификация которых невозможна (нет связи с разработчиками, утрачены исходники и пр.).

Стоит заметить, что параллельно может быть организовано несколько процессов взаимодействия систем с использованием одного общего механизма интеграции. Такой подход довольно эффективен, поскольку осуществляется общий контроль для всех процессов интеграции, а спроектированный с учетом такой работы транспортный агент может обслуживать несколько процессов обмена данными одновременно. В этих случаях двустороннюю интеграцию корректно рассматривать как двустороннюю в пределах одного процесса интеграции. Пример такой интеграции – механизм обмена сообщениями, например, с экономической системой, где один тип сообщений обеспечивает синхронизацию списка договоров в МИС и экономической системе, а другой – направляет информацию об оказанных услугах из МИС для формирования счетов.

Интеграция с медицинскими и немедицинскими системами. Зачастую имеет значение, с какой системой, медицинской или немедицинской, производится интеграция. При интеграции с медицинской системой передаваемые данные обычно являются элементами неких лечебно-диагностических мероприятий. В таких случаях, как правило, к интеграции предъявляются более строгие требования, большую важность приобретают время реакции системы на запрос, общее время обмена информацией, защищенность от потери и искажения данных и т.п. Интеграция же с немедицинской системой обычно не проводится в режиме реального времени, поэтому периоды между обменом данными могут быть более длитель-

ными и сам обмен зачастую носит односторонний характер.

Примером интеграции с медицинскими системами может служить взаимодействие МИС с лабораторными системами, диагностическим оборудованием, со сторонними МИС и т.п. Требуется постоянная актуализация данных, поскольку необходимо обслуживать случаи, когда в один день пациент и будет внесен в МИС, и пройдет обследование, и получит диагностические документы на руки, для чего данные о нем должны быть реплицированы в диагностической системе за время его перемещения от регистратуры до кабинета.

Наиболее распространенными примерами интеграции МИС с немедицинскими системами является интеграция на уровне экономических подсистем, систем материального учета, со страховщиками и пр. В таких примерах, как правило, актуализация информации требуется значительно реже – несколько раз в месяц, для составления отчетности и согласований.

Автоматизированная и неавтоматизированная интеграция. Интеграцию ИС можно классифицировать либо как автоматизированный процесс обмена данными, либо как процесс, где присутствует ручное оперирование данными. Примерами могут являться интеграции с лабораторными системами (как правило, автоматизированные) и интеграции со сторонними немедицинскими организациями, в частности, со страховыми компаниями (часто реализованные в виде автоматической обработки данных, введенных вручную).

Качественное различие типов заключается в уровне адаптации внешних данных в МИС.

При полной автоматизации процесса обмена данными ошибок адаптации данных бывает существенно меньше, а случающиеся ошибки носят систематический характер, что позволяет диагностировать причину их возникновения и устранить либо адаптировать МИС к обработке таких ошибок.

Опыт использования различных решений интеграции показывает, что при обработке ошибок в данных автоматизированного взаимодействия число ошибок и количество их разновидностей в несколько раз ниже, чем при обработке данных, вводимых вручную. В частности, при обработке полученных от страховых компаний списков прикрепленных по *добровольному медицинскому страхованию* (ДМС) пациентов, те списки, которые составлялись автоматизированно в ИС страховых компаний, потребовали 2-3 итерации разработки механизма интеграции, после чего обработка списков велась сотрудниками лечебного учреждения систематически в штатном режиме. Те списки, которые составляются страховыми компаниями вручную, регулярно вынуждают персонал затрачивать время на анализ и исправление все новых ошибок в этих списках даже после нескольких итераций разработки механизма интеграции.

Наличие обратной связи. Существенным различием в процессах интеграции является наличие/отсутствие обратной связи со стороны интегрируемой системы. Под обратной связью здесь понимается возможность получения информации и влияния на механизм интеграции со стороны интегрирующей системы.

В случае отсутствия обратной связи организация взаимодействия ограничена пересечением возможных механизмов интеграции, доступных в системах, а обработка ошибок возможна только в варианте адаптации системы к систематизированным ошибкам.

Примерами систем без наличия обратной связи могут служить системы, поддержка которых прекращена, находящиеся вне сферы влияния, архитектура которых не допускает изменений.

Системы с наличием обратной связи, как правило, позволяют строить более совершенные механизмы интеграции за счет своей адаптивности.

Транспортный агент и форматы

Важным моментом при построении интеграции является правильный выбор транспортного агента и формата передачи данных. Под транспортным агентом здесь подразумевается технический набор средств, позволяющий осуществлять физическую передачу данных из одной системы в другую.

Транспортного агента и формат данных необходимо выбирать исходя из классификации по указанным признакам, а также с учетом специфики данных. Так, например, для автоматизированных систем эффективным является выбор агента, основанного на стандартном протоколе передачи данных (например ODBC) между базами данных интегрируемых систем. В частности, при интеграции с диагностическим оборудованием либо системой целесообразно поддерживать постоянную возможность связи по какому-либо протоколу. Однако если данные формируются во внешней системе вручную либо требуется поддерживать возможность принятия кванта данных, составленного вручную, может оказаться удобным выбор менее технологичного, а то и вовсе нестандартного транспортного агента. Например, при работе со страховыми компаниями для обработки списков прикрепленных по ДМС пациентов в качестве транспортного агента применялись файлы *Microsoft Excel*. Это позволило совместить автоматизированное формирование таких файлов в одних страховых компаниях и формирование вручную в других.

От транспортного агента также могут зависеть скорость обмена данными при интеграции, надежность механизма интеграции, доступная полнота контроля работы механизма и многое другое.

В качестве транспортного агента при интеграции МИС наиболее корректным принято считать обмен сообщениями с использованием стандарта HL7. Необходимо заметить, что этот стандарт поддерживается целым рядом медицинского оборудования. Это особенно характерно для медицинских приборов зарубежного производства, поскольку стандарт HL7 получил заметное распространение в странах Европы и в США.

Поскольку применимость HL7 для всей отечественной системы здравоохранения неочевидна, а специалистов, имеющих навык работы с HL7, пока мало, многие ставят под сомнение целесообразность интеграции на основе HL7. Тем более что огромная доля подсистем, разработанных и эксплуатирующихся в различных медицинских учреждениях уже длительное время, была спроектирована без поддержки HL7. Однако из стандарта HL7 можно почерпнуть ряд полезных идей и принципов. Так, механизм обмена сообщениями представляется наиболее предпочтительным, даже если эти сообщения специфичны для некой конкретной интеграции. Такой тип транспортного агента – один из лучших способов доставки данных и контроля с обеспечением разграничения ответственности на каждой стадии.

Безопасность

Одна из главных проблем при интеграции ИС – обеспечение информационной безопасностью. Как правило, требуется защищать и транспортного агента, который физически передает данные, и каждую из интегрируемых систем. За исключением случаев, когда в рамках интеграции разрабатывается новый специфический транспортный агент, для каждого вида агентов существуют типовые решения для защиты либо известно, что выбранный агент защите не поддается.

Внимание следует уделить следующим аспектам:

– необходимо обеспечить безопасность данных, передаваемых транспортным агентом.

– данные, отраженные в интегрированной системе, защищаются средствами этой системы. Таким образом, если защищенная система интегрируется с системой, имеющей менее эффективную защиту, данные, которые в рамках интеграции могут быть переданы между системами, следует считать защищенными на уровне менее защищенной системы.

В то время как транспортного агента можно защитить как техническими средствами (шифрование, ограничение доступа к каналу передачи и пр.), так и административными (например, организационно ограничить доступ к оборудованию, осуществляющему передачу данных), возможности защиты системы без наличия обратной связи бывают весьма ограниченными. Кроме того, необходимо проверять подлинность системы, участвующей в обмене данными.

Разграничение ответственности

Непременным элементом процесса интеграции является разграничение ответственности между интегрируемыми системами. В общих чертах процесс интеграции включает следующие этапы:

– подготовка данных для передачи системой А,

– отправка данных системой А,

– передача данных от системы А к системе Б,

– прием данных системой Б,

– сохранение полученных данных в структуре системы Б,

– отправка подтверждения принятия данных из системы Б в систему А,

– передача подтверждения от системы Б к системе А,

– прием подтверждения системой А.

Каждый из этих этапов должен попадать под ответственность какой-либо из систем. Трудоемкость выявления системы, допустившей ошибку при передаче данных, напрямую зависит от количества этапов, ответственность за которые разделяется между системами.

Обработка ошибок

При интеграции систем неизбежно возникают ошибки в передаваемых данных. Их можно разделить на следующие типы.

– Ошибки из-за несоответствия моделей, архитектур либо структур систем и данных в системах: возникают при некорректной проекции данных одной системы на другую. Чаще всего это следствие недостаточной проработанности механизма интеграции. Некоторые из таких ошибок можно обрабатывать за счет сопоставления и вычисления.

– Ошибки, возникающие при некорректности данных: считаются допустимыми при функционировании внутри отдельной системы, поскольку в ней не происходит столкновения корректных и некорректных (либо некорректных с обеих сторон) данных. А при интеграции это может проявиться.

– Ошибки дублирования данных: вызывают определенные проблемы даже внутри отдельной системы, однако при интеграции сложность проблемы значительно вырастает.

Следует заметить: поскольку процесс интеграции ИС практически всегда основан на обмене данными, многие ошибки можно выявить, описать и устранить на основе нормальных форм, используемых в теории БД. Рассматривая интегрированные системы как целую БД, проводя параллели между понятиями БД и данными в процессе интеграции, можно выявить источник ошибок путем вычисления нарушений нормальных форм. То

есть даже если в интегрируемых системах не соблюдаются нормальные формы, соблюдение их (хотя бы первых трех) в механизме интеграции может сократить количество ошибок и привести к более эффективному функционированию. Подход представляется интересным для исследования, поскольку в случае корректного нахождения аналогий и применения нормальных форм общий принцип нормализации можно использовать для построения эффективного механизма интеграции.

Регламент процедуры обмена данными и временные интервалы

При интеграции систем практически всегда рассматриваются вопросы о том, как часто и в каких случаях должны попадать данные из одной системы в другую. Можно выделить три модели поведения при передаче данных – через определенные временные интервалы, по определенному событию, по запросу.

Рассмотрим их подробнее.

Передача данных через определенные временные интервалы. Эта модель, как правило, реализуется для систем, работающих параллельно на основе неких данных, передача которых лежит в основе интеграции. При таком подходе каждый временной интервал системы начинают с одинаковым синхронизированным набором данных, предполагая, что изменение этих данных в течение временного интервала не имеет значения для процессов, протекающих внутри системы. Данная схема считается одной из наиболее простых и часто используемых. Однако такая модель наиболее подвержена возникновению ошибок, поскольку лишь немногие интеграции действительно абсолютно не зависят от изменения данных в течение временного интервала. Наблюдения показывают, что при достаточно активной работе учреждения в сфере, затрагивающей участвующую в процессе интеграции область данных, значимые изменения регулярно случаются. И чем больше временной интервал, тем больше ошибок возникает при обмене данными. Проблема заключается в том, что, несмотря на длину интервала, которую диктует непосредственно участвующий в интеграции процесс, в системе могут протекать и другие процессы, косвенно использующие эти данные, причем процессы рассчитаны на совершенно иной временной интервал дискретизации. Опыт внедрения показывает, что следует выбирать минимальные из допустимых временные интервалы, а при наличии возможности лучше и вовсе отказаться от данной модели в пользу передачи данных по определенному событию.

Примером передачи данных через определенные временные интервалы может служить ежедневная выгрузка данных о пациентах в специализированную систему скорой помощи.

Передача данных по определенному событию. Данная модель часто используется в случаях, когда в процессе обмена передается не просто общий набор данных, а некие информационные объекты, построенные на основе этих данных. В таком случае требуется передать не только сами данные, но и связь между ними, данные необходимо передать целым набором, так как частичная передача однотипных данных не дает нужного результата. Кроме того, такая модель более удобна для контроля передачи, поскольку данные поступают отдельными квантами, которые можно протоколировать.

Модель предполагает равномерную нагрузку, так как однотипные события в лечебных процессах обычно не обладают высокой плотностью для создания пиковых нагрузок на транспортный агент.

Эта модель наиболее подходит для механизма интеграции на основе обмена сообщениями.

Преимуществом данного подхода также является асинхронность передачи данных с их востребованностью. Так, данные подготавливаются для передачи максимально рано и могут быть востребованы по мере их доставки транспортным агентом.

Недостатком модели является ее непригодность для передачи интенсивного постоянного потока элементарных данных даже при их небольшом совокупном объеме. Примером такой модели являются практически все интеграции с медицинским оборудованием.

Передача данных по запросу является в общем случае наиболее эффективно использующей транспортный агент моделью. Данные передаются только по требованию системы, причем с указанием в запросе заявки на конкретные данные. Это минимизирует нагрузку на транспортный агент в целом, однако может создавать значительную пиковую нагрузку в момент запроса. Следовательно, такая модель оптимальна при достаточно малой пиковой нагрузке в момент запроса.

Недостатком модели является зависимость от работоспособности транспортного агента в момент запроса. Так, если транспортный агент в момент запроса не может доставить данные, то их ожидание равносильно ожиданию транспортного агента. Предпочтительнее случай, когда получение данных происходит заблаговременно.

Примерами для этой модели являются некоторые интеграции с лабораторными системами, которым периодически требуется запрашивать дополнительные данные о пациенте.

Регламент передачи данных. Выбранная модель с указанием ее параметров должна быть описана в соответствующем регламенте и утверждена заинтересованными сторонами, как правило, представителями от разработчиков систем и от медицинских учреждений, использующих ИС.

Пересечение интегрируемых систем

В ряде случаев интегрированные системы могут пересекаться в сфере действий. Например, в одной системе по данным, доступным с помощью интеграции, можно построить отчетность, которая отражает деятельность другой системы. Это особо актуально при интеграции с системами без наличия обратной связи. В частности, за счет интеграции с системой, поддержка которой прекращена, можно развивать дополнительный функционал, недоступный в исходной системе.

Различия в представлениях данных

Неприятной особенностью является такая интеграция, при которой одни и те же данные требуется представлять в различной форме. В частности, это относится к большим системам, охватывающим разные подразделения учреждения. Различные подразделения могут оперировать одними исходными данными, представляя их в разных формах. Например, наименование договора либо услуги по ряду причин может иметь различия в написании или даже в обозначении.

Сопоставление и вычисление данных

При интеграции систем часто возникает необходимость в сопоставлении данных. Это относится к случаям, когда одни и те же данные параллельно ведутся в интегрируемых системах и обмен данными содержит в себе некую проекцию данных и их отношений из одной системы в другую. Бывает, что из схемы данных и административных регламентов можно вычислить однозначное соответствие. Однако в некоторых случаях набор данных, которые может предоставить одна система, меньше набора данных, необходимых для создания (или идентификации) информационного объекта в другой системе. В таком случае нужно либо пытаться выполнить вычисление требуемых данных, либо административно определить процесс дополнения набора передаваемых данных до требуемого.

Например, из лабораторной системы можно извлечь данные о проведенном исследовании пациента, в которых, однако, отсутствует информация о договоре на ДМС, по которому была оказана такая услуга. Учитывая административно утвержденную невозможность оказания услуг пациенту одновременно по нескольким договорам ДМС, есть возможность вычислить договор, по которому была оказана услуга, после чего использовать эту информацию для предоставления отчетов в страховую компанию.

Необходимо отметить, что некоторые данные могут быть как сопоставлены, так и вычислены. В этом случае не всегда однозначно следует выбирать сопоставление: есть вероятность, что данные, вычисляемые внутри МИС, могут оказаться более корректными либо более удобными для использования в системе. Выбор метода дополнения данных необходимо определить на основе анализа репрезентативной выборки данных, на которых основана интеграция.

Проблемы интеграции в МИС

На основании изложенного можно выделить ряд общих проблем, возникающих при интеграции в МИС: использование закрытых, неописанных форматов; ошибки операторов систем; зависимость от технического обеспечения связи из-за физической удаленности систем; необходимость защиты систем и механизма интеграции от угроз информационной безопасности; наличие систем без поддержки и прочих закрытых систем, интеграция с которыми проводится в одностороннем порядке; отсутствие регламентов и проблемы с разграничением ответственности за ошибки обмена данными; затягивающиеся сроки реализации интеграции ввиду количества участвующих сторон и их инертности; необходимость тщательного анализа структур данных, участвующих в обмене в рамках интеграции.

С учетом различных интеграций типовой МИС Интерин PROMIS с лабораторными, диагностическими, экономическими, страховыми и прочими системами, а также со специализированными подсистемами отдельных медицинских учреждений обобщена типовая схема интеграции. Выделены основные модели интеграции, рассмотрены типичные преимущества и недостатки моделей; выявлены и классифицированы наиболее распространенные ошибки в обмене данных интегрированных систем.

Поскольку не существует МИС, охватывающих все процессы медицинского учреждения без исключения, вопрос интеграции крайне актуален для любой устанавливающейся в клиниках и больницах ИС. Однако наиболее актуален он при плавном внедрении, когда на время замещения имеющейся подсистемы модулями МИС требуется сохранить работоспособность новой системы в связке с имеющейся подсистемой.

Из всех означенных типовых процессов интеграции наиболее эффективным в общем случае представляется двусторонняя интеграция систем на основе обмена сообщениями (возможно, стандарт HL7) и передачи данных по событию с учтенными и обоснованными нарушениями аналогий нормальных форм либо без подобных нарушений.

С учетом данных выводов в МИС Интерин PROMIS был реализован ряд механизмов интеграции, в частности, с экономической подсистемой корпорации «Парус», лабораторными системами (УниверЛаб, *Ilims*), собственными разработками медицинских учреждений. Реализованные механизмы показали высокую степень адаптивности, устойчивости к ошибкам и отличное качество контроля интеграции.

Литература

1. Гулиев Я.И., Хаткевич М.И. Процесс и документ в медицинских информационных системах: тр. Междунар. конф. / ИПС РАН, Переславль-Залесский; под ред. С.М. Абрамова. В 2 т. М.: Физматлит. 2004. Т. 2. С. 169 (Программные системы: теория и приложения).

2. Малых В.Л., Пименов С.П., Хаткевич М.И. Объектно-реляционный подход к созданию больших информационных систем: тр. Междунар. конф. / ИПС РАН, Переславль-Залесский; под ред. А.К. Айламазяна. М.: Наука. Физматлит, 1999. С. 177 (Программные системы: Теоретические основы и приложения).

3. Интегрированная распределенная информационная система лечебного учреждения (ИНТЕРИН) / Я.И. Гулиев [и др.] // Программные продукты и системы. 1997. № 3.

ПРЕЦЕДЕНТЫ В МЕДИЦИНСКИХ ИНФОРМАЦИОННЫХ СИСТЕМАХ

В.Л. Малых, к.т.н.; Я.И. Гулиев, к.т.н. (Исследовательский центр медицинской информатики ИПС им. А.К. Айламазяна РАН, г. Переславль-Залесский, mvl@interin.ru)

PRECEDENTS IN MEDICAL INFORMATION SYSTEMS

Malykh Vladimir L., Ph. D.; Guliev Yadulla I., Ph. D. *(Research Centre for Medical Informatics, Aylamazyan Program Systems Institute, RAS, Pereslavl-Zalessky, mvl@interin.ru)*

Abstract. The main goal of this research is to justify a wide usage of precedents in medical information systems. Precedents are determined as functionality reflections of sets of observed events included in different classes. The authors describe the results of practical approbation of this idea and outline some perspectives of precedents usage.

Keywords: *precedents, information system, medical information system, information system architecture, medical diagnostic process.*

Целью работы является обоснование целесообразности широкого использования прецедентов в медицинских информационных системах. Прецеденты вводятся на основе построения отображений определенных на множествах наблюдаемых событий различных классов. Приведены примеры практической апробации этой идеи, намечены дальнейшие многообещающие перспективы использования прецедентов.

Ключевые слова: *прецеденты, информационная система, медицинская информационная система, архитектура информационной системы, лечебно-диагностический процесс.*

Современные интегрированные *медицинские информационные системы* (МИС) ориентированы на всестороннюю автоматизацию и детальный контроль *лечебно-диагностического процесса* (ЛДП). Сам ЛДП состоит из потока событий, ассоциированных с различными лечебно-диагностическими мероприятиями: осмотрами, диагностическими исследованиями, лабораторными анализами, лечебными назначениями, процедурами и манипуляциями, оперативными вмешательствами. События ассоциируются с участвующими в них субъектами: пациентами, врачами, лаборантами, прочим медицинским персоналом. События обладают темпоральными свойствами: датой и временем возникновения, длительностью протекания. Каждое событие ЛДП является по своей сути уникальным, единичным и относится к конкретному пациенту, определенному времени, содержит свой набор значений именованных атрибутов, например, ассоциацию с основным или сопутствующим диагнозом. МИС фиксирует множество таких единичных событий в своей БД. Для крупных лечебных заведений количество детально фиксируемых событий ЛДП исчисляется миллионами в год.

Кроме автоматизации, собственно ЛДП МИС должна также поддерживать и другие процессы, связанные с деятельностью лечебного учреждения. Прежде всего речь идет о материальном обеспечении ЛДП (лекарственными средствами, медицинским инвентарем, реактивами и др.). Эффективное управление современным лечебным учреждением невозможно без ведения детального финансово-экономического анализа доходов и затрат. При этом особое внимание уделяют учету движения и расходования всех *материальных ценностей* (МЦ). Многие из этих материалов расходуются непосредственно на пациента (медикаменты, медицинский инвентарь, продукты питания, другие расходные материалы), формируя прямые затраты, существенно влияющие на себестоимость лечения. Желательность точного учета и анализа прямых затрат в условиях рыночной экономики ясна всем [1, 2, 5]. Детальный контроль финансовых и материальных потоков еще больше увеличивает количество различных событий, фиксируемых в БД интегрированной МИС.

МИС не должна ограничиваться только задачами фактографии. На передний план выходят проблемы контроля событий – фактов, отраженных в БД: почему сделано то или иное лечебно-диагностическое назначение, соответствует ли это назначение стандарту лечения данного заболевания, чем обусловлен расход данного материала, какими назначениями или нормативами расхода?

Решить данные проблемы и реализовать автоматизированный контроль событий в МИС можно с помощью принципиально нового подхода, осно-

ванного на прецедентах. Прецедент (от лат. *praecedens* – предшествующий) – случай или событие, имевшие место в прошлом и служащие примером или основанием для аналогичных действий в настоящем. В основе идеи использования прецедентов лежит тот факт, что общее существует в неразрывной связи с единичным (Аристотель) и что событиям ЛДП присущи не только уникальность и единичность, но и общность и повторяемость. Люди болеют одними и теми же болезнями, их лечат по одним и тем же стандартам, назначают одни и те же лечебно-диагностические мероприятия, соответствующие современному уровню медицинских знаний и возможностям лечебного учреждения. При проведении этих мероприятий в среднем расходуется определенное количество МЦ. В частности, из общности и повторяемости событий ЛДП вытекает возможность формирования стандартов лечения, из общности и повторяемости материальных затрат, связанных с той или иной лечебно-диагностической процедурой (услугой), – возможность формирования норм расхода материалов.

События и прецеденты

Используем формализацию события, приведенную в [3]. В основе базиса формализации события лежат понятия, на их основе определим *атомарный факт* (АФ) как пару понятий $\mathbf{a=(a_1,a_2), a_i \in A}$, где $\mathbf{A}$ – множество конечных последовательностей символов некоторого алфавита (тексты). Первое в паре понятие является общим абстрактным понятием: ‘пациент’, ‘диагноз’, ‘группа крови’. Второе – это конкретизация первого: ‘Иванов’, ‘ОРЗ’, ‘IV’.

В качестве базиса представления фактов и знаний рассматриваются события $\mathbf{S}$, которые конструируются из атомарных фактов и являются конечными последовательностями атомарных фактов $\mathbf{S=\{a^i\}, i \in N, a \in A^2}$.

Весь ЛДП может быть представлен однородно в виде множества событий. Можно пытаться рассматривать абстрактную задачу поиска общего у множества событий, сведя ее к выделению общих абстрактных понятий и общих фактов у этих событий. Подобные постановки свойственны искусственному интеллекту. Но более конструктивную и практически значимую постановку можно получить, если определять прецеденты относительно классов событий. Само по себе отнесение событий к некоторому содержательному классу уже говорит о том, что были проведены определенные классификация и генерализация событий и что события, относящиеся к одному классу, имеют что-то общее. Итак, рассматривая некое множество наблюдаемых событий $\{\mathbf{S^i}\}$, принадлежащих некоторому классу $\mathbf{S^i \in K}$, выделим на этом классе отображение $\mathbf{p: K \to S}$, определяющее на множестве наблюдаемых событий $\{\mathbf{S^i}\}$ множество прецедентов. Два события $\mathbf{S^1 \in K}$ и $\mathbf{S^2 \in K}$ порождают один и тот же прецедент $\mathbf{P \in S}$ относительно отображения $\mathbf{p}$, если $\mathbf{p(S^1)=p(S^2)=P}$.

Пример. Рассмотрим класс событий – медикаментозные лечебные назначения. Приведем несколько событий из этого класса, упростив их модель.

S^1=
{
(‘дата и время события’, ‘?’),
(‘автор события’, ‘врач X’),
(‘пациент’, ‘NN’),
(‘диагноз’, ‘аллергический ринит’),
(‘наименование лекарственного средства’, ‘Тавегил’),
(‘способ введения’, ‘перорально’),
(‘дозировка’, ‘1 мг’),
(‘периодичность’, ‘2 раз в день’)
},
S^2=
{
(‘дата и время события’, ‘?’),
(‘автор события’, ‘врач Y’),
(‘пациент’, ‘NN’),
(‘диагноз’, ‘аллергический ринит’),
(‘наименование лекарственного средства’, ‘Тавегил’),
(‘способ введения’, ‘перорально’),
(‘дозировка’, ‘1 мг’),
(‘периодичность’, ‘2 раз в день’)
},
S^3=
{
(‘дата и время события’, ‘?’),
(‘автор события’, ‘врач Z’),
(‘пациент’, ‘NN’),
(‘диагноз’, ‘экзема’),
(‘наименование лекарственного средства’, ‘Тавегил’),
(‘способ введения’, ‘перорально’),
(‘дозировка’, ‘1 мг’),
(‘периодичность’, ‘2 раз в день’)
}

Знаками **‘?’** заменены не определенные нами и несущественные для дальнейшего изложения атрибуты атомарных фактов. Построим отображение $\mathbf{p_d}$ для выделения из указанного класса прецедентов: отображение заключается в удалении из событий атомарных фактов, включающих в себя следующие общие понятия (‘дата и время события’, ‘автор события’, ‘пациент’). Тогда события $\mathbf{S^1}$, $\mathbf{S^2}$ и $\mathbf{S^3}$ порождают относительно $\mathbf{p_d}$ два прецедента:

$p_d(S^1)=p_d(S^2)$=
{
(‘диагноз’, ‘аллергический ринит’),
(‘наименование лекарственного средства’, ‘Тавегил’),
(‘способ введения’, ‘перорально’),
(‘дозировка’, ‘1 мг’),
(‘периодичность’, ‘2 раз в день’)
},
$p_d(S^3)$=
{
(‘диагноз’, ‘экзема’),
(‘наименование лекарственного средства’, ‘Тавегил’),
(‘способ введения’, ‘перорально’),
(‘дозировка’,‘1 мг’),
(‘периодичность’, ‘2 раз в день’)
}.

Рассмотренное отображение $\mathbf{p_d}$ выделяет пре-

цеденты медикаментозных назначений, содержащих информацию о том, какие лекарственные средства назначаются при определенном заболевании.

Рассмотрим другое отображение $\mathbf{p_p}$, которое выделяет прецеденты медикаментозных назначений с информацией о том, какие лекарственные средства назначались конкретному пациенту независимо от заболевания.

$\mathbf{p_p(S^1)=p_p(S^2)=p_p(S^3)=}$
{
('пациент', 'NN'),
('наименование лекарственного средства', 'Тавегил'),
('способ введения', 'перорально'),
('дозировка', '1 мг'),
('периодичность', '2 раз в день')
}.

На первый взгляд кажется, что прецеденты выделяются с помощью простого «проектирования» исходных событий в «подпространство меньшей размерности». Безусловно, такое проектирование имеет место, но выделение прецедентов отнюдь не сводится только к нему. Допустим, что тот же 'Тавегил' назначался пациенту 'NN' в различных дозировках: '1 мг', '2 мг', '3 мг', '4 мг', '5 мг' и '6 мг'. Эти события породят шесть различных прецедентов относительно отображения $\mathbf{p_p}$. Но можно свести все шесть прецедентов к одному, если изменить отображение и ввести в рассмотрение ранг дозировки. В результате получится следующий прецедент:

P=
{
('пациент', 'NN'),
('наименование лекарственного средства', 'Тавегил'),
('способ введения', 'перорально'),
('минимальная дозировка', '1 мг'),
('максимальная дозировка', '6 мг'),
('периодичность', '2 раз в день')
}.

Заметим, что теперь отображение усложнилось и его аргументами выступает не только само событие **S**, элемент класса **K**, но и все наблюдаемые события этого класса $\{\mathbf{S^i}\}, \mathbf{S^i \in K}$, так как ранг атрибута атомарного факта определяется уже всеми наблюдаемыми событиями вместе: $\mathbf{P=p(S,\{S^i\}), S\in\{S^i\}, S^i\in K}$. Ранг в указанных примерах можно вычислять и для периодичности. Кроме ранга, можно использовать любые другие характеристики, вычисляемые по множеству наблюдаемых событий, например, статистические средние и дисперсии. Из других важных характеристик прецедента, вычисляемых по всему множеству наблюдаемых событий определенного класса, введем в рассмотрение мощность прецедента как число событий, порождающих этот прецедент. Учитывая наличие у событий темпоральных характеристик, введем в рассмотрение дату и время последней актуализации прецедента как дату и время самого недавнего события, порождающего данный прецедент. Эти характеристики очень важны для оценки значимости прецедента и позволяют строить различные стратегии применения знаний о прецедентах. Расширим предыдущий пример указанными характеристиками

P=
{
('мощность прецедента', '?'),
('дата и время последней актуализации прецедента', '?'),
('пациент', 'NN'),
('наименование лекарственного средства', 'Тавегил'),
('способ введения', 'перорально'),
('минимальная дозировка', '1 мг'),
('максимальная дозировка', '6 мг'),
('периодичность', '2 раз в день')
}.

В результате такого расширенного понимания прецеденты приобретают изменяющиеся во времени характеристики и перестают быть чисто статическими. При изменении множества наблюдаемых событий за счет появления новых событий могут появиться новые прецеденты и измениться характеристики уже известных прецедентов.

В окончательной формулировке прецеденты определяются отображениями отдельных наблюдаемых событий совместно со множеством всех наблюдаемых событий, которые определены на некотором классе, в событие-прецедент $\mathbf{p:\{S^i\}\times N\to S}$.

Здесь $\{\mathbf{S^i}\}$ – множество наблюдаемых событий определенного класса **K**; **N** – множество натуральных чисел, значения из которого принимает верхний индекс события **i** из множества наблюдаемых событий; **p** – отображение заданного наблюдаемого события, определенного индексом, и всего множества наблюдаемых событий в событие-прецедент **S**.

Обсудим конкретные примеры применения прецедентов в МИС.

Автоматизация учета в МИС прямых материальных затрат

От современных МИС требуется полная аналитика движения МЦ, начиная с источников финансирования закупок материалов и заканчивая точным знанием расходов материалов до непосредственного их списания на пациента (персонифицированный учет). Организация планомерной закупки материалов также требует точной статистики по расходу. Знание всех прямых затрат на конкретные категории пациентов, а также по нозологиям (формально закодированным диагнозам) и отделениям, вплоть до знания расхода материалов в отдельных лечебных процедурах, позволяет более точно оценивать себестоимость как всего лечения, так и отдельных процедур, а также дает возможность составлять прейскуранты на платные услуги.

Задача автоматизации учета прямых материальных затрат в МИС сталкивается с рядом проблем [5, 6]. Одна из них – переход к персонифи-

цированному материальному учету, вовлекающий в себя широкий круг пользователей МИС. Речь идет в первую очередь о среднем медицинском персонале, который непосредственно расходует материальные средства для исполнения предписанных врачами назначений и вносит данные о расходе в МИС.

Решить проблемы и реализовать детальный персонифицированный учет прямых материальных затрат в МИС можно с помощью принципиально нового подхода к организации материального учета, построенного на прецедентах.

В основе прецедентного построения материального учета лежит включение в каждое событие из класса событий – записей о расходе МЦ – информации о том, какой материал был затребован и чем обусловлено это требование. Формально событие о расходе МЦ запишем в следующем виде:

S=
{
(‘дата и время события’, ‘?’),
(‘автор события’, ‘?’),
(‘наименование затребованного материала’, ‘?’),
(‘количество затребованного материала’, ‘?’),
(‘единицы измерения затребованного количества’, ‘?’),
(‘наименование израсходованного материала’, ‘?’),
(‘количество израсходованного материала’, ‘?’),
(‘единицы измерения израсходованного количества’, ‘?’),
}.

Приведем пример события о расходовании МЦ

S=
{
(‘дата и время события’, ‘?’),
(‘автор события’, ‘медсестра N’),
(‘наименование затребованного материала’, ‘Метрагил’),
(‘количество затребованного материала’, ‘1000’),
(‘единицы измерения затребованного количества’, ‘мг’),
(‘наименование израсходованного материала’, ‘Метронидазол 0,5 % 100 мл’),
(‘количество израсходованного материала’, ‘2’),
(‘единицы измерения израсходованного количества’, ‘шт.’),
}.

Содержательная трактовка этого события: автор события – медсестра **N** соотнесла с назначенным врачом «Метрагилом» числящийся по материальному учету «Метронидазол». Затем перевела указанную врачом дозировку 1000 мг в учетные единицы 2 шт., имея в виду, что, согласно форме выпуска этого лекарственного средства «0,5 % 100 мл», в 100 мл раствора содержится 0,5 г «Метронидазола», а в двух штуках соответственно 1 г = 1000 мг. Поставим в соответствие этому событию следующий прецедент:

P=
{
(‘мощность прецедента’, ‘?’),
(‘дата и время последней актуализации прецедента’, ‘?’),
(‘автор события’, ‘медсестра N’),
(‘наименование затребованного материала’, ‘Метрагил’),
(‘единицы измерения затребованного количества’, ‘мг’),
(‘наименование израсходованного материала’, ‘Метронидазол 0,5 % 100 мл’),
(‘количество израсходованного материала’, ‘2’),
(‘единицы измерения израсходованного количества’, ‘шт.’),
(‘пропорция между количеством затребованного материала и количеством израсходованного материала’, ‘500’),
}.

Видно, что возникший прецедент несет в себе определенные знания. Во-первых, произошло согласование наименования лекарственного средства, указанного врачом в лечебном назначении, с торговым наименованием и конкретной формой выпуска лекарственного средства, числящегося по материальному учету. Во-вторых, появилась пропорция между затребованным и израсходованным количеством лекарственного средства, взятым в различных единицах измерения. В указанном примере врач может уменьшить дозировку лечебного назначения до 500 мг. Если при этом будет израсходована 1 шт. указанного лекарственного средства, такое событие приведет к уже известному прецеденту, так как 1000:2 = 500:1 и пропорция не изменилась.

Преимущества введения в материальный учет прецедентов

Во-первых, избавление от проблемы контроля каждой записи о расходе, сведение контроля расхода к контролю прецедентов. Очевидно, что мощность множества прецедентов существенно меньше, чем мощность множества записей о расходе. В БД крупного московского ЛПУ при вводе в работу прецедентного материального учета примерно за один квартал зарегистрировано 468 679 записей о расходе и 21 626 соответствующих прецедентов. В этом примере мощность множества прецедентов примерно в 22 раза меньше мощности первичного множества записей о расходе. В дальнейшем мощность множества записей о расходе будет расти линейно, пропорционально времени, а мощность множества прецедентов будет иметь тенденцию к слабому росту, обусловленному в основном появлением новых пользователей системы (врачей и среднего медперсонала), а также связанному с расширением номенклатуры материалов. Прецеденты еще разбиваются по авторам и, соответственно, по отделениям. В нашем примере 91 автор. В каждом отделении каждому уполномоченному лицу (аудитору) потребуется контролировать 1–3 тыс. прецедентов.

Во-вторых, оперативный контроль сводится в основном только к анализу новых прецедентов. На появление новых записей о расходе, для которых уже имеются зарегистрированные в системе и признанные правильными прецеденты, при контроле можно не обращать внимания. Это позволяет сильно снизить нагрузку на пользователей-аудиторов и свести ее к рассмотрению вновь созданных прецедентов и их анализу на корректность.

В-третьих, все зарегистрированные системой прецеденты обусловлены требованиями к подсистеме материального учета со стороны других подсистем – лечебно-диагностических назначений, ведения клинических документов, учета услуг. В

идеале записей о расходе, не обусловленных требованиями, не должно быть. Лица, непосредственно расходующие материалы в ходе ЛДП и вносящие записи о расходе в БД, как правило, не формируют в системе требования на материалы и, следовательно, не могут создать не обоснованные с точки зрения ЛДП требования. Любая запись о расходе связана с требованием и им обусловлена.

В-четвертых, знание прецедентов позволяет реализовать в подсистеме материального учета механизм автоматической разноски требований. Анализируя строку конкретного разносимого требования, система может обнаружить в БД подходящие для этой строки прецеденты. На основании выделенных, подходящих прецедентов можно автоматически выполнить разноску. Для начала следует проранжировать выделенные подходящие прецеденты. В рассматриваемой модели прецедентов присутствуют атрибуты автора и мощности прецедента, а также даты его последней актуализации. На основании этих атрибутов можно предложить различные простые стратегии ранжирования прецедентов. Например, сначала прецеденты ранжируются по авторам и в первую очередь применяются прецеденты, созданные самим пользователем. Затем прецеденты ранжируются по мере убывания их мощности, то есть в первую очередь применяются наиболее часто используемые прецеденты. При равенстве мощности двух прецедентов предпочтение может быть отдано тому, у которого дата актуализации старше. Очевидно, что это не единственная возможная стратегия. Например, можно включить в атрибуты прецедента признак его корректности и вести разноску только на основании правильных прецедентов. Некоторые прецеденты могут быть нетипичными, хотя и правильными. Эти нетипичные маргинальные прецеденты также удобно размечать отдельным признаком и не использовать при разноске, хотя для нетипичных прецедентов хорошим фильтром также будет служить мощность прецедента. На основании отобранных, подошедших к затребованному и проранжированных прецедентов последовательно делается попытка связать имеющиеся материальные запасы с требованием. В случае успеха формируются связанные с затребованным автоматически разнесенные записи о расходе. Пользователю остается подтвердить правильность выполненной разноски и изменить разноску там, где он не согласен с результатом автоматической разноски, а затем окончательно завершить транзакцию, изменив материальные остатки. В результате в общем случае будут подтверждены ранее известные системе прецеденты (увеличена их мощность), а также будут созданы новые прецеденты.

В-пятых, отпадает необходимость специально согласовывать справочники различных подсистем. Например, подсистемы лечебно-диагностических назначений и подсистемы материального учета. Собственно, сами прецеденты теперь предоставляют пользователю механизм согласования.

В-шестых, прецеденты, связывая требования и отпуск, становятся элементами базы знаний системы. Появление нового прецедента равносильно появлению новых знаний в системе, за которым следует возникновение соответствующих событий, например, уведомления о необходимости контроля нового прецедента – необходимости проверки новых знаний, немедленная возможность функционального использования новых знаний, например, для выполнения автоматической разноски.

Для авторов статьи чрезвычайно важно, что предложенная концепция прецедентного материального учета не осталась чисто теоретической разработкой, а нашла свое практическое применение. Успешная апробация прецедентного материального учета состоялась в 2008 году в ФГУ «Клиническая больница Управления делами Президента Российской Федерации» [6].

Использование прецедентов в подсистеме лечебно-диагностических назначений

Существует такая максима, что весь ЛДП сводится к лечебно-диагностическим назначениям. Во всяком случае, основной функцией врача в МИС наряду с ведением различных клинических документов (осмотров, дневников, протоколов, заключений и т.п.) является формирование назначений. Этот трудоемкий процесс можно облегчить и автоматизировать, если использовать шаблоны назначений и выбрать из них уже готовые формализованные назначения. Использование прецедентов позволяет по-новому решить проблему шаблонирования назначений. Обычной практикой в МИС является либо непосредственный ручной ввод шаблонов назначений, либо выбор уже сделанных назначений для их включения в тот или иной шаблон. Автоматически шаблоны не пополняются, для этого требуется ручная работа. Все эти недостатки обычных шаблонов могут исправить прецеденты. Во-первых, прецеденты формируются информационной системой полностью автоматически на основе возникших событий (на основе уже сделанных назначений). Во-вторых, прецеденты над классами лечебно-диагностических назначений могут быть построены несколькими различными отображениями, что позволяет достичь совершенно нового качества в анализе и функциональном использовании прецедентных знаний.

В прецеденты можно включать информацию об авторе прецедента. В этом случае каждый врач может использовать в качестве шаблонов свои прецеденты, но одновременно с этим использовать прецеденты назначений других авторитетных врачей. В прецеденты можно включать ассоциа-

цию с услугой, в рамках которой было сделано назначение. В этом случае при выборе конкретной услуги можно сразу же получить все прецеденты назначений, с нею связанных. Коммерциализация современной медицины делает подобный отбор прецедентов вполне оправданным. Прецеденты можно связать с медицинской картой, историей болезни или амбулаторной картой. История болезни выделяет некий определенный законченный эпизод (этап) в жизни и лечении пациента. Прецеденты, связанные с историей болезни, характеризуют все лечебно-диагностические назначения, сделанные в ходе этого эпизода ЛДП. Прецеденты, ассоциированные с пациентом, несут всю имеющуюся в данной МИС информацию обо всех лечебно-диагностических назначениях для этого пациента. Прецеденты можно ассоциировать с диагнозом (формальным кодом по международному классификатору болезней). В этом случае можно выделить прецеденты назначений, связанных с тем или иным заболеванием, что отражает общепринятую практику врача: идти от поставленного диагноза к назначениям. Напомним, что прецеденты также несут информацию о своей мощности, что позволяет ориентироваться при их выборе на типичные, наиболее часто применяемые прецеденты.

Стандарты лечения и прецеденты

Современная медицина выдвигает жесткие требования к составу и содержанию лечебных и диагностических мероприятий по каждой нозологии. С одной стороны, должен быть обеспечен достаточный их набор, гарантирующий пациенту адекватное лечение, с другой – следует избегать избыточных лечебных вмешательств, неадекватное увеличение которых ведет к повышению риска для больного (развитие осложнений в процессе инструментальных исследований) и возрастанию стоимости лечения. Стандартизация медицинских технологических процессов, удовлетворяющих таким требованиям, – весьма трудоемкая задача [4], однако ее можно эффективно решать на основе анализа ретроспективных данных, хранящихся в МИС, извлечения потоков событий ЛДП, анализа прецедентов, выделения на этой основе фактических стандартов лечения с их последующим экспертным анализом и оптимизацией.

Очевидно, что в общем лечебно-диагностический процесс не может быть сведен только к прецедентам. Важны не только выделенные в прецеденты события и их кратность, но и сам порядок событий. Полную информацию об этом несут только исходные, не свернутые в прецеденты события. Однако если обратиться к стандартам оказания медицинской помощи, разработанным Министерством здравоохранения и социального развития РФ, можно обнаружить, что сами стандарты имеют ярко выраженный прецедентный характер. В этих стандартах не указана явная последовательность лечебно-диагностических событий, приводятся частота предоставления (фактически это требование к вероятности появления события в ЛДП) и среднее количество лечебно-диагностических событий в расчете на некоторый временной период. В качестве примера приведем выдержки из стандарта медицинской помощи больным инсулинозависимым сахарным диабетом.

Министерство здравоохранения и социального развития. Стандарт медицинской помощи больным сахарным диабетом от 07.04.2005 № 262.

Модель пациента

Категория пациента: дети, взрослые

Нозологическая форма: инсулинозависимый сахарный диабет

Код по МКБ-10: Е10

Фаза: хроническая

Стадия: все стадии

Осложнение: нефропатия, ретинопатия, нейропатия, катаракта, диабетическая стопа, хайропатия

Условие оказания: амбулаторно-поликлиническая помощь

Лечение из расчета 12 месяцев.

Код	Наименование	Частота предоставления	Сред. кол.
A11.05.001	Взятие крови из пальца	1	12
A09.05.023	Исследование уровня глюкозы в крови	1	12
A09.05.084	Исследование уровня гликилированного гемоглобина крови	1	2
A11.12.009	Взятие крови из периферической вены	1	2
A09.05.017	Исследование уровня мочевины в крови	1	1
A09.05.020	Исследование уровня креатинина в крови	0,5	1
A09.05.021	Исследование уровня общего билирубина в крови	0,1	1
A09.05.026	Исследование уровня холестерина в крови	0,1	1
A09.05.027	Исследование уровня липопротеинов в крови	0,1	1
A09.05.028	Исследование уровня липопротеидов низкой плотности	0,1	1
A09.05.041	Исследование уровня аспартат-трансаминазы в крови	0,1	1
A09.05.042	Исследование уровня аланин-трансаминазы в крови	0,1	1
A09.28.001	Микроскопическое исследование осадка мочи	1	2
A09.28.011	Исследование уровня глюкозы в моче	1	4
A09.28.015	Обнаружение кетоновых тел в моче	0,5	4
A09.28.023	Определение удельного веса (относительной плотности) мочи	1	2
A09.28.003	Определение белка в моче	1	2
A05.10.001	Регистрация электрокардиограммы	1	1

Медикаменты

Фармакотерапевтическая группа: гормоны и средства, влияющие на эндокринную систему (**частота назначения** 0,7)

Международное непатентованное наименование	Частота назначения	Ориентировочная дневная доза	Эквивалентная курсовая доза
Анатомо-терапевтическо-химическая группа – инсулины (**частота назначения** 0,2):			
инсулин растворимый (человеческий генноинженерный)	0,2	30 МЕ	10950 МЕ

Международное непатентованное наименование	Частота назначения	Ориентировочная дневная доза	Эквивалентная курсовая доза
инсулин аспарт	0,2	30 МЕ	10950 МЕ
инсулин лизпро	0,2	30 МЕ	10950 МЕ
инсулин изофан (человеческий генноинженерный)	0,2	20 МЕ	7300 МЕ
инсулин гларгин	0,2	20 МЕ	7300 МЕ
Анатомо-терапевтическо-химическая группа – сахароснижающие препараты для перорального применения (**частота назначения** 0,8):			
глибенкламид	0,3	8 мг	2880 мг
гликлазид	0,15	100 мг	36000 мг
глимепирид	0,15	4 мг	1440 мг
метформин	0,15	1,5 г	540 г
гликвидон	0,15	120 мг	43800 мг
зепаглинид	0,1	16 мг	5840 мг

Анализ структуры стандарта показал, что контроль фактического ЛДП на соответствие стандарту можно свести к выделению из ЛДП прецедентов и их ассоциации с лечебно-диагностическими событиями из стандарта, сопоставлению мощности и других характеристик прецедента с соответствующими характеристиками из стандарта. С помощью тех же прецедентов ЛДП, ассоциированных с определенной болезнью (нозологией), можно решить задачу выделения стандартов лечения *de facto*.

Документы и прецеденты

Электронные документы занимают очень важное место в МИС. В них отражается весь ЛДП. Медицинская карта фактически представляет собой коллекцию различных документов как в классическом бумажном, так и в электронном виде. Документы описывают лечебно-диагностические события. Следовательно, структура документа становится носителем знания о ЛДП. В принятой в ИПС РАН архитектуре реализации документов в МИС [7] структура документа изначально принимается в виде дерева, в узлах которого находятся атомарные факты. Задается порядок обхода дерева. Для каждого атомарного факта вводится понятие локального и глобального контекстов [8], определяемых всеми другими атомарными фактами, пройденными при обходе дерева от вершины до заданного атомарного факта. Контексты нужны для правильной семантической интерпретации данного атомарного факта, а также для разбора данных документа и их сохранения в БД в разобранном виде. Опыт автоматизации медицинского клинического документооборота показал, что структуры документов даже одного типа варьируются от учреждения к учреждению и отражают разные предпочтения врачей, различную сложившуюся практику ведения медицинских карт в ЛПУ, различие медицинских школ в отражении ЛДП в документах. Все разработанные в ИПС РАН модели клинических документов собраны в единый банк документов, который уже сейчас содержит свыше 250 структурированных документов различных типов (а с учетом внутритиповых вариаций гораздо больше). Прецедентный анализ структур документов позволяет решить сразу несколько задач. Во-первых, можно выделить прецеденты связи каждого АФ с родительскими и дочерними АФ (из АФ в прецеденты включается только абстрактное понятие). Множество выделенных абстрактных понятий и отношений включения на них позволяют сформировать локальный внутрисистемный тезаурус понятий для предметной области, после чего становится возможным конструирование документов на основе тезауруса. Конструирование можно начать с любого абстрактного понятия из тезауруса и затем двигаться от него как вверх, так и вниз по указанным в тезаурусе связям с другими абстрактными понятиями, включая их в структуру. Очевидно, что таким образом можно получить все структуры (модели документов) из банка данных, а также сконструировать новые гибридные документы. Во-вторых, для каждого данного АФ можно выделить прецеденты всех контекстов использования данного факта. Это позволит при конструировании структуры документа вести ее проверку на соответствие контекстов всех АФ известным контекстам их использования, и автоматизирует семантический контроль конструируемого документа. Каждая новая модель документа вносит с собой новое знание, которое усваивается системой в соответствующих прецедентах и в тезаурусе.

Система информационной безопасности и прецеденты

К современным МИС выдвигаются жесткие требования по защите информации от несанкционированных изменений и от несанкционированного доступа. Особую роль приобретают персональные данные. Защита персональных данных регламентируется нормативными документами, принятыми на федеральном уровне. Вопросы информационной безопасности также поднимаются в национальном Стандарте РФ ГОСТ- 52636 «Электронная история болезни. Общие положения».

Для МИС крупного ЛПУ характерна проблема большой мощности потока событий, которые подвергаются аудиту системой информационной безопасности. Огромное число источников данных: на уровне реляционной БД число таблиц и представлений может исчисляться тысячами. Громадное количество программных модулей и функциональных программных единиц (функций, процедур), вызываемых пользователями, – тысячи единиц. Число пользователей в крупных ЛПУ уже приближается к тысяче. Все это создает проблемы для аудита событий доступа к информации. Прецеденты в состоянии решить проблему мощности потока событий, отказавшись от контроля непосредственных первичных событий и сведя аудит к контролю новых, ранее не подвергавшихся аудиту прецедентов.

Реализация уровня прецедентов в архитектуре МИС

Методические рекомендации по реализации уровня прецедентов в архитектуре МИС следующие. В первую очередь необходим тщательный отбор классов наблюдаемых событий для выделения из них прецедентов. Хороший класс – это уже полезное предварительное обобщение информации. Далее строятся различные отображения классов в прецеденты. Отображения определяются практическим назначением выделяемых прецедентов. Некоторое опасение может вызвать тот факт, что при построении каждого отдельного прецедента в общем случае имеется необходимость в обработке всего множества наблюдаемых событий данного класса. Это требуется для получения характеристик прецедентов, вычисляемых по всем наблюдаемым событиям. Может показаться, что появление каждого нового события потребует заново пересчитать все уже выделенные прецеденты, так как произошло расширение множества наблюдаемых событий. Легко видеть, что необходимые для пересчета «интегральные» характеристики наблюдаемого множества событий можно хранить в самих прецедентах (ранг, мощность прецедента и т.п.). Зачастую этих ранее рассчитанных интегральных характеристик оказывается достаточно для их обновления в связи с появлением нового события. Например, если появилось новое событие, для которого ранее был выделен прецедент, потребуется просто увеличить мощность этого прецедента на 1 и перенести из события в прецедент дату и время актуализации события. Технически в системе можно реализовать выделение прецедентов с помощью синхронных и асинхронных процессов. Синхронные процессы выделения прецедентов работают в той же транзакции, в которой создаются новые события ЛДП. Поэтому создание событий и выделение соответствующих прецедентов происходят практически одновременно. Процесс выделения прецедентов может оказаться достаточно ресурсоемким в первую очередь по времени. Тогда можно выделять прецеденты с помощью асинхронных процессов. В этом случае возникновение новых событий ЛДП фиксируется в виде некоторой очереди вторичных событий, несущих информацию о произошедших первичных событиях. Асинхронный процесс периодически запускается и производит обработку очереди вторичных событий, актуализируя уже известные прецеденты и выделяя новые прецеденты из новых событий ЛДП.

В работе перечислены далеко не все возможности применения прецедентов в МИС. Несмотря на относительную простоту данной концепции прецедента, практические перспективы прецедентов (даже неожиданно для авторов) оказались весьма широки. Связано это с тем, что прецеденты – мощное средство обобщения и генерализации знаний, заключенных в событиях и документах. В статье описаны возможности использования прецедентов для автоматизации процесса формирования лечебно-диагностических назначений, для контроля ЛДП на соответствие стандартам оказания медицинской помощи, при формировании фактической базы стандартов лечения *de facto* для разработки собственных стандартов лечения. Велика роль прецедентов для структуризации и концептуализации знаний в виде тезауруса понятий предметной области. Прецеденты позволяют автоматизировать конструирование моделей документов на основе тезауруса с включением в процесс конструирования семантического контроля. Прецеденты могут найти широкое применение в системе информационной безопасности и автоматизировать аудит событий, контролирующих доступ к медицинской информации, включая персональную информацию. Они уже нашли практическое применение в области детального персонифицированного учета и контроля прямых материальных затрат ЛДП. Прецеденты позволяют справиться с большой мощностью потока событий ЛДП. Отдельного внимания заслуживает тот факт, что они являются носителями знаний, выражающихся в ассоциировании между собой АФ, входящих в прецедент. Контроль нового прецедента – это фактически подтверждение новых знаний компетентным аудитором с немедленной возможностью дальнейшего функционального использования подтвержденных знаний. На основании изложенного можно сделать вывод, что прецеденты вышли за границы отдельного частного механизма и претендуют занять отдельный уровень в архитектуре современной МИС.

Литература

1. Кадыров Ф.Н. Экономические методы оценки эффективности деятельности медицинских учреждений. М.: Издат. дом «Менеджер здравоохранения», 2007.

2. Мчедлидзе Т.Ш., Янченко В.М., Касумова М.К. Управление медицинским бизнесом: Система управления стоматологической организацией. СПб.: ООО «MEDИ издательство», 2007.

3. Гулиев Я.И., Малых В.Л., Юрченко С.Г. Контекстный анализ событий и синтез структуры медицинских знаний. Современные информационные и телемедицинские технологии для здравоохранения (AITTH'2008): матер. II междунар. конф. Минск: Объединенный институт проблем информатики НАН Беларуси, 2008. С. 164–168.

4. Назаренко Г.И., Полубенцева Е.И. Управление качеством медицинской помощи. М.: Медицина, 2000.

5. Калинин А.Н., Малых В.Л., Юсуфов Т.Ш. Управление материальными ресурсами ЛПУ в МИС. Аптека и диетпитание // Врач и информационные технологии. 2006. № 4. С. 87–90.

6. Малых В.Л., Гулиев Я.И., Крылов А.И., Рюмина Е.В. Проблемы автоматизации учета прямых материальных затрат в медицине. Архитектура прецедентного материального учета // Аудит и финансовый анализ. 2009. № 2. С. 465–471.

7. Гулиев Я.И., Малых В.Л. Архитектура HL-X: тр. Междунар. конф. / ИПС РАН, Переславль-Залесский; под ред. С.М. Абрамова. В 2 т. 2004: М.: Физматлит. Т. 2. С. 147–168. (Программные системы: теория и приложения).

8. Малых В.Л., Юрченко С.Г. Документальный уровень

представления знаний в интегрированной медицинской информационной системе: Там же. С. 217–230.

РЕАЛИЗАЦИЯ ДОКУМЕНТОВ В МЕДИЦИНСКОЙ ИНФОРМАЦИОННОЙ СИСТЕМЕ ИНТЕРИН

С.Г. Юрченко (Исследовательский центр медицинской информатики ИПС им. А.К. Айламазяна РАН, г. Переславль-Залесский, yurch@interin.ru)

DOCUMENTS IMPLEMENTATION IN THE INTERIN MEDICAL INFORMATION SYSTEM

Yurchenko Sergey G. *(Research Centre for Medical Informatics, Organization of Russian Academy of Sciences Ailamazyan Program Systems Institute of RAS, Pereslavl-Zalessky , yurch@interin.ru)*

Abstract: The article lists main requirements for electronic medical documents. The author uses as example their implementation in the Interin medical information system.

Keywords: *electronic medical document, medical information system, information system architecture.*

В статье приводятся основные требования к электронным медицинским документам. В качестве примера описывается их реализация в медицинской информационной системе Интерин.

Ключевые слова: *электронный медицинский документ, медицинская информационная система, архитектура информационных систем.*

В *медицинских информационных системах* (МИС) очень важна поддержка медицинских документов. Согласно ГОСТ Р 52636-2006, *персональная медицинская запись* (ПМЗ) может содержать описание проведенного осмотра или обследования (в том числе лабораторного или инструментального), консультации, назначения, выполненной операции или процедуры, обобщенного заключения о состоянии больного и т.д. Совокупность таких записей, выполненных традиционным способом в конкретном медицинском учреждении, составляет историю болезни, или амбулаторную карту пациента.

По сути весь лечебно-диагностический процесс находит свое отражение в документах, являющихся основным носителем медицинской информации о пациенте. В крупных медицинских учреждениях за год может заполняться до нескольких сотен тысяч документов. Поскольку подобные документы являются источником персональных и конфиденциальных данных, они должны отвечать следующим требованиям:

- неизменность и достоверность на протяжении всего периода хранения;
- регламентация прав доступа и конфиденциальность;
- персонифицируемость (возможность определить автора и происхождение записи в любой момент – аналог подписи на традиционном документе).

Счет различным типам документов идет на сотни, причем зачастую отсутствует общепринятый стандарт, регламентирующий содержание информации и порядок, в котором она должна включаться в документ. Один и тот же осмотр в разных больницах может выглядеть по-разному в зависимости от предпочтений специалистов, работающих в них. В процессе работы эти предпочтения могут меняться: например, врач решает добавить какое-то новое поле или изменить список возможных значений в существующем. Следует учитывать и различия между стационарными и поликлиническими учреждениями. Дополнительно могут существовать несколько вариантов визуального представления одного и того же документа, скажем, выписка из него, выдаваемая на руки пациенту, где указан адрес медицинского учреждения или иная контактная информация. В электронных версиях документов, в отличие от традиционных бумажных, могут подавляться незаполненные поля.

Так как МИС являются коллективными, то есть подразумевают одновременную работу большого количества пользователей, должно соблюдаться несколько условий. Во-первых, запрет на одновременное внесение изменений несколькими пользователями: пока один не закончит работу с документом, другие могут открывать его лишь для просмотра, но не для редактирования. Во-вторых, у одного документа может быть несколько авторов, поэтому соответствующим образом должна быть авторизована любая информация, вносимая в него, чтобы можно было узнать, кто менял те или иные данные. В-третьих, права доступа к документу могут быть ограничены, например, его смогут видеть лишь лечащий врач пациента и заведующий отделением, для остальных пользователей системы он будет недоступен.

Для соблюдения неизменности окончательно заполненного документа может использоваться электронная цифровая подпись. После этого автор теряет возможность исправлять, менять либо удалять его, такое право остается только за специальными сотрудниками, указанными в политике безопасности МИС.

Информация в том виде, в каком она хранится в документе, может отличаться от структуры данных, используемой в БД МИС, более того, не все

данные могут быть жестко структурированы и находить свое отражение в таблицах БД. Часть из них хранится лишь в теле документа и извлекается из него в случае необходимости.

Представляется важной поддержка описания структуры документа с использованием соответствующей терминологии. В качестве примера можно привести стандарт CDA (*Clinical Document Architecture*), определяющий структуру данных клинического документа и процесс его машинной обработки. В стандарте подразумевается использование архетипов и медицинских терминов в описании структуры. Однако в отсутствие общепринятого соответствия русско- и англоязычных медицинских терминов адаптация западных стандартов затруднительна, а национальные стандарты подобного типа пока не приняты.

Учитывая большое количество документов и возможность изменения со временем их структуры, представляется необходимым наличие средств быстрого конструирования документов.

Исследовательским центром медицинской информатики ИПС РАН была разработана архитектура HL-X для поддержки документов. В работе [1] сформулирована концепция документа и определены различные требования (информационные, функциональные, терминологические) к нему. Автор статьи принимал самое непосредственное участие в реализации данной архитектуры.

Были выдвинуты следующие основные принципы реализации в МИС подсистемы поддержки медицинских документов.

Содержимое документа и его визуальное представление разделены. Используется трехуровневая архитектура: БД, содержимое документа (назовем его моделью данных), описание визуального представления (модель визуализации). Процесс их взаимодействия подчиняется следующим правилам: модель данных взаимодействует с БД МИС, модель визуализации – с моделью данных, взаимодействие визуальной модели напрямую с БД должно быть сведено к минимуму.

Структура БД МИС может отличаться от структуры данных в документах. Они должны быть независимыми друг от друга, структура таблиц БД не должна перестраиваться под структуру данных, содержащихся в документах, и наоборот.

Данные в документах могут быть слабо структурированными (анамнез жизни, данные осмотра) и сильно структурированными (диагноз, назначения, движение по отделениям). Для заполнения сильно структурированных данных может понадобиться вызов внешнего приложения либо заполнение отдельного документа. В таком случае используется следующая схема работы: вызывается внешняя форма либо другой документ, которые сохраняют информацию в БД, откуда она зачитывается в модель данных текущего документа.

Структура и внешний вид документов могут со временем меняться, отражая новые стандарты, личные предпочтения врачей, специфику медицинского учреждения. Должна поддерживаться историчность моделей документов.

Желательно обеспечить оптимизацию ввода данных пользователем. Если врач будет тратить много времени на заполнение документов, у него останется меньше времени на осмотр и лечение пациентов. В качестве одного из решений данной проблемы предлагается использовать заимствование в документах информации, уже содержащейся в МИС (например, личных данных пациента), содержимого других документов, шаблонов (заранее сформированных пользователями наборов данных).

Может потребоваться ввод в документы не только текстовой, но и графической информации, а также прикрепление внешних файлов (например данных с прибора).

Разработанная архитектура электронного медицинского документа включает в себя формализованные описания структуры модели данных и модели визуализации документа, метаописание структуры БД, набор визуальных компонентов (для отображения информации в документе). Все модели и описания структуры хранятся в той же БД, к которой обращается и МИС.

Для визуализации документов на сервере приложений используются динамически генерируемые HTML-страницы.

Модели документов описываются на языке XML, чем обеспечивается их гибкость (с дальнейшим расширением возможностей) и простота внесения изменений. Поддерживается историчность не только моделей, но и визуальных компонентов – это сохраняет неизменность внешнего представления документа с момента подписания.

Для запроса информации, необходимой в документах, из БД МИС используется набор так называемых объектов-запросов (*query*-объектов). Чтобы обеспечить дополнительную гибкость, возможно прямое указание в модели запросов на языке SQL.

Выгрузка информации из документа в БД МИС осуществляется на основе тезауруса, где хранится описание таблиц БД и связь полей таблицы с узлами модели данных документа.

Для поиска и отбора экземпляров документов с каждым из них связывается набор описывающих его атрибутов, таких как пациент, автор, дата создания и т.п.

Доступ к документам контролируется с использованием механизма пользователей СУБД, что позволяет осуществлять сквозную авторизацию, не запрашивая имя и пароль дважды, сначала при входе в МИС, а затем отдельно для доступа к документам.

Для того чтобы исключить одновременную правку документа двумя или более пользователя-

ми, при открытии документа на редактирование он помечается как *заблокированный*. В таком случае другой пользователь при попытке открыть тот же документ сможет увидеть его лишь в режиме просмотра, не имея возможности ничего в нем менять. Как только первый пользователь закроет документ, блокировка будет снята и документ станет свободным для редактирования.

При сохранении документ обретает статус черновика. Если же врач закончил ввод данных и не собирается больше ничего менять в документе, он выполняет команду его подписания. После этого редактирование документа становится невозможным, его можно открывать лишь для просмотра.

Документ в статусе черновика может быть удален, но запись о нем все равно остается; он становится недоступным в МИС, однако может быть при необходимости восстановлен (его статус изменен на *черновик*) администратором системы.

Каждому документу сопоставлен автор. В некоторых случаях требуется, чтобы документ содержал информацию, внесенную несколькими авторами, тогда в модели данных документа отражается, кто был последним, кто изменял значение того или иного поля.

Структура модели данных документа описывается на языке XML в соответствии с некоторыми правилами.

В качестве наименований узлов выбираются названия понятий, описывающих содержащиеся в них данные. Например, узел, где будет храниться диагноз, называется английским словом <DIAGNOSIS> либо транскрипцией <DIAGNOZ>. Это обеспечивает удобочитаемость модели в процессе разработки, легко понять, какое поле из визуального представления документа в какой узел модели данных попадет. Допускаются альтернативные названия вложенных узлов, например, <PATIENT_NAME /> либо <PATIENT><NAME /></PATIENT>, второй вариант предпочтительнее. Новая версия стандарта XML дает возможность использовать в качестве названия термины на русском языке, без перевода, в версии XML 1.0 можно было использовать лишь латинский алфавит.

Кроме того, есть несколько названий для служебных узлов, позволяющих обозначить разделы документа или иную техническую информацию.

Для обеспечения заимствования данных между узлами модели используются команды работы с контекстом. При обработке документа узел может поместить свое значение в динамически поддерживаемый ассоциативный массив, откуда другой узел это значение может забрать. Например, узел, содержащий дату создания документа, через этот массив может передать свое значение узлу с *датой осмотра*. Порядок обхода узлов соответствует некоторым заданным правилам, таким образом, каждому узлу при обходе может быть доступен свой контекст (набор именованных значений), отсюда было выбрано такое название для данного массива. Знание всех контекстов использования данных документа позволяет ставить задачу построения конструктора документов, обеспечивающего в процессе конструирования семантическую корректность модели [2]. Например, конструктор не позволит включить в документ данные о результатах лабораторных исследований вне контекста конкретного пациента и его медкарты, даты забора лабораторных материалов и т.п.

Специальные объекты-запросы (получающие данные из БД) помечаются особым образом, чтобы отделить их от узлов, служащих лишь для хранения данных. Запрос может выполняться однократно (при создании документа) либо при каждом обращении к документу.

Примеры объектов-запросов: информация о пациенте (Ф.И.О., дата рождения, домашний адрес и т.п.), диагноз, имя пользователя, список возможных типов диет.

Запрос может возвращать как одно, так и несколько значений, во втором случае узел такого объекта тиражируется нужное количество раз. *Обычные* узлы (не запросы) также могут быть помечены как *множественные*. Например, в предоперационной концепции есть поле «симультанная операция», а их может быть несколько. Для каждой выбранной операции данный узел и все вложенные в него узлы будут размножены. Если пользователь затем удалит какую-то из выбранных операций, соответствующее поддерево модели будет удалено.

Для того чтобы экземпляры документов можно было отбирать по заданным параметрам, специальный раздел модели отведен под подобные значения, такие как идентификатор пациента, ссылка на автора, дата создания документа и т.п. Они выгружаются в специальную таблицу БД, проиндексированную для ускорения поиска, чтобы не обращаться каждый раз к XML-модели документа.

Для ускорения заполнения документов врач может пользоваться шаблонами. Например, можно создать шаблоны для наиболее часто встречающихся нозологий (заболеваний) и затем выбирать нужный, сразу же получая заполненный документ, где ему останется лишь поменять несколько значений.

Шаблон – такая же модель данных, как и обычный документ, а потому в его качестве может использоваться другой экземпляр документа того же типа. Кроме того, добавлена возможность создания схем преобразования моделей из одного типа в другой, что позволяет заимствовать данные, например, из переводного эпикриза в выписной.

Шаблоны могут быть личными (то есть доступными лишь создавшему их) и доступными любому врачу отделения или всего медицинского

учреждения.

Кроме того, можно создать так называемый шаблон *по умолчанию*: при создании каждого нового экземпляра документа данного типа в него будут сразу же заимствованы данные из такого шаблона. Таким образом, врач может создать некий типичный заполненный осмотр, в котором ему надо будет только изменить значения, отклонившиеся от нормы, что может многократно ускорить заполнение.

Визуальная модель документа представляет собой XML-документ, состоящий из узлов, соответствующих так называемым визуальным компонентам. Каждый компонент формирует тот или иной тип поля или набор полей. Они могут быть редактируемыми и нередактируемыми.

Примеры редактируемых компонентов: текстовое поле (однострочное либо многострочное), число, дата и время, выбор файла, переключатели (*checkbox*), выпадающий список, графическое изображение (с возможностью добавления пометок).

Примеры нередактируемых компонентов: заданный статичный текст, заголовок раздела документа, данные назначений.

Работа с документом осуществляется в двух режимах: редактирование и просмотр. Для облегчения поддержки оба режима описываются одной моделью. В зависимости от режима визуальные компоненты формируют различающийся *html*-код. Поля, в которые не вводились значения, в режиме просмотра обычно не отображаются. Это и более удобно для пользователя, и экономит место при печати бумажной копии документа.

Все компоненты являются гибконастраиваемыми, имея набор заданных параметров. Например, можно указать, что поле даты осмотра не должно быть пустым, тогда, если пользователь удалит его значение, документ не даст ему продолжить работу, пока он не введет новое значение. Можно ограничить максимальную длину вводимого значения и т.п.

Так как предусмотреть все возможные варианты, которые могут понадобиться для описания визуального представления документа, невозможно, имеется компонент «фрагмент *html*-кода», в котором можно задать любой *html*-код, включая активные сценарии на *JavaScript*.

У большинства компонентов есть параметр, связывающий их с узлом модели данных. Это означает, что при открытии документа он будет брать оттуда значение, а при его изменении помещать новое значение обратно в тот же узел.

Существует возможность сформировать группу компонентов, которую можно растиражировать нужное количество раз, например, в осмотре стоматолога дать возможность врачу описать лечение нескольких зубов (количество изначально неизвестно: один зуб или несколько).

Некоторые поля можно пометить как обязательные для заполнения, иначе автор не сможет закончить работу с документом. Другие поля могут быть динамически изменяющимися: например, «индекс массы тела» может меняться в зависимости от введенных значений веса и роста пациента.

Существует возможность скрывать или показывать отдельные поля либо целые разделы в зависимости от выбранного значения. Это может понадобиться для более полного контроля над вводимыми данными, чтобы врач не мог, к примеру, выбрать «патологий нет» и после этого по ошибке ввести текст в поле описания патологий.

Для того чтобы дать возможность выгружать некоторые наборы полей из документа в БД МИС, но без необходимости писать каждый раз программный код, был разработан тезаурус. Он представляет собой описание соответствия между узлами модели данных документа и полями таблицы БД. На основе этого описания приложение автоматически составляет SQL-запрос на вставку, обновление либо удаление записи в таблице.

Все наборы узлов в модели данных, которым требуется такая возможность, помечаются особым образом, формируя объект для выгрузки. Так как некоторые поля таблицы могут оказаться обязательными для заполнения, можно указать набор узлов, который предварительно проверяется на отсутствие пустых значений, иначе данные выгружаться не будут.

Очевидно, что не все данные могут соответствовать таблицам, в каких-то случаях вместо этого требуется вызвать некую функцию с нужными параметрами и т.п. Такая возможность также доступна.

Разработанная подсистема удовлетворяет большинству поставленных условий, однако имеет несколько путей улучшения: использование электронной цифровой подписи, возможность выгрузки документов в форматы PDF либо *Microsoft Word* для управления печатью, а также добавление средств форматирования текста и проверки орфографии для всех либо некоторых полей.

Литература

1. Гулиев Я.И., Малых В.Л. Архитектура HL-X: тр. Междунар. конф. / ИПС РАН, Переславль-Залесский; под ред. С.М. Абрамова. В 2 т. М.: Физматлит. 2004. Т. 2. С. 147–168 (Программные системы: теория и приложения).

2. Малых В.Л., Юрченко С.Г. Документальный уровень представления знаний в интегрированной медицинской информационной системе: Там же. С. 217–230.

3. Guliev Y.I., Malykh V.L., Yurchenko S.G. Conceptual models for representing information in healthcare information systems. Advanced Information and Telemedicine Technologies for Health (AITTH-2005). Minsk, 2005, vol 1, pp. 198–201.

ПОДДЕРЖКА МНОГОКОМПОНЕНТНОСТИ В МЕДИЦИНСКИХ ИНФОРМАЦИОННЫХ СИСТЕМАХ

Д.В. Алимов (ИПС им. А.К. Айламазяна РАН, г. Переславль-Залесский, alimov@interin.ru)

***Alimov Dmitry V.** (Research Centre for Medical Informatics, Organization of Russian Academy of Sciences Ailamazyan Program Systems Institute of RAS, Pereslavl-Zalessky, alimov@interin.ru)*

Abstract. This paper is intended to explore the automation problems of complex healthcare institutions based on the usage of several single-type system components. The author justifies the need to build a special mechanism providing both data separation and integration between the components into the architecture of such multicomponent systems. Author describes user-proved engineering solutions of realization technology of multicomponent mechanism in the Interin PROMIS healthcare information system.

Keywords: *multicomponents, data integration, healthcare information systems, healthcare information system architecture.*

В статье исследуются проблемы автоматизации комплексных лечебно-профилактических учреждений на основе использования нескольких однотипных системных компонент. Обосновывается необходимость встраивания в архитектуру таких многокомпонентных систем специального механизма, обеспечивающего разделение и интеграцию данных между компонентами. Описываются апробированные технические решения по реализации механизма многокомпонентности в МИС Интерин PROMIS.

Ключевые слова: *многокомпонентность, интеграция данных, медицинские информационные системы, архитектура медицинских информационных систем.*

Одним из важных свойств, которыми должна обладать интегрированная *медицинская информационная система* (МИС), является ее способность автоматизировать работу больших и очень больших *лечебно-профилактических учреждений* (ЛПУ), представляющих собой сложные комплексы с многократно повторяющимися структурными подразделениями. Для таких комплексных ЛПУ актуальны задачи получения данных о работе каждой структурной компоненты учреждения, которая сама может выступать как самостоятельное учреждение, а также задачи получения данных о работе всего комплекса (многокомпонентного учреждения в целом).

Одним из подходов к решению задачи объединения/разделения данных является интеграция различных МИС в единое целое. При интеграции встает вопрос о реализации логики обмена данными между различными МИС. Есть разнообразные пути интеграции *информационных систем* (ИС): разработка протоколов передачи данных, создание программных интерфейсов и пр. Но как бы тесно системы ни были интегрированы, при таком подходе они продолжают оставаться обособленными МИС со своей логикой работы, своими справочниками и IT-специалистами по обслуживанию каждой такой ИС.

Другой подход – это создание единой ИС комплексного ЛПУ, в которой подсистемы, информатизирующие то или иное направление деятельности учреждения, являются компонентами (возможно, множественными). В этом случае возможно объединение в ИС сразу нескольких однотипных компонент ЛПУ в единое целое. Это необходимо, например, при информатизации лечебного учреждения, имеющего в своем составе несколько стационаров. ИС, с одной стороны, должна разделять данные и хранить информацию о том, какие именно какой компоненте принадлежат, а с другой – каждый пользователь должен иметь возможность использовать данные всех подсистем (например, при статистической обработке информации). Системный механизм, позволяющий объединять в едином информационном пространстве компоненты как одного, так и нескольких типов, получил название *механизма поддержки совместной работы МИС в мультипликативных (множественных) структурах ЛПУ*.

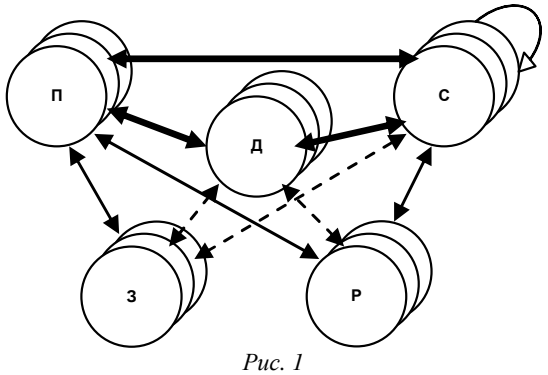


Рис. 1
З – здравпункты, П – поликлиники, С – стационары комплексного ЛПУ, Д – диагностический центр, Р – реабилитационный центр (санаторий)

От того, насколько удачно в ИС комплексного ЛПУ будет реализован механизм поддержки работы МИС в мультипликативных структурах, зависит эффективность работы всей системы.

В данной статье рассматриваются схема комплексного ЛПУ и один из методов реализации ме-

ханизма работы в мультипликативных структурах, используемый в МИС Интерин PROMIS.

В общем случае комплексное ЛПУ может состоять из стационара, диагностического центра, поликлиники, реабилитационного центра и здравпункта. Причем каждая компонента может быть не единственной. Например, может быть несколько профильных стационаров, территориально удаленных здравпунктов и т.д. Заметим, что даже в случае с выделенным диагностическим центром каждая компонента может иметь свой внутренний диагностический центр, как почти всегда и бывает с территориально разнесенными подразделениями ЛПУ.

Схема обобщенного комплексного ЛПУ представлена на рисунке 1.

Наиболее активное информационное взаимодействие происходит между компонентами *поликлиника, стационар, диагностический центр*. Оно происходит как на уровне пациентов, так и на уровне врачей, когда врачи одних структурных компонент привлекаются для оказания услуг в других компонентах ЛПУ. В качестве примера можно привести ситуацию, когда для консультации пациента в стационаре приглашается врач поликлиники.

Требования к механизму поддержки совместной работы МИС

Корпоративная медицинская система комплексного ЛПУ характеризуется следующими параметрами: наличие модулей для информатизации каждой компоненты, единое хранилище медицинских карт, единые справочники медицинского персонала и подразделений комплексного ЛПУ.

Наличие единого хранилища данных и единых системных справочников МИС позволяет легко получить данные о работе всего комплексного ЛПУ в целом. Однако, если имеются две и более компонент одного и того же типа (например, когда в ЛПУ несколько стационаров), встают задачи разделения данных и определения их принадлежности к той или иной структурной компоненте.

Если в систему входят компоненты *стационар* и *поликлиника*, медицинские карты пациентов стационара можно выделить по типу карты (в стационаре тип медкарты – история болезни). Но если в общем хранилище размещаются, например, медицинские карты двух стационаров, разделение по типу медкарты использовать нельзя.

Учитывая это, механизм поддержки многокомпонентности должен иметь функциональность разметки данных, позволяющую определить принадлежность данных к той или иной компоненте.

В комплексном ЛПУ врачи работают, как правило, в рамках одной структурной компоненты, но в каких-то случаях обращаются и к данным, относящимся к другой компоненте.

В качестве примера можно рассмотреть перемещение пациента в стационаре. Пациент перемещается по отделениям одного стационара, но возможны и переводы в отделения другого. Такое случается, если в комплексном ЛПУ пациенту, находящемуся в отделении терапевтического стационара, требуется хирургическая помощь, оказать которую можно только в отделении хирургического стационара. В этом случае пользователь МИС должен иметь возможность, работая в рамках одной структурной компоненты, перевести пациента в отделение другой компоненты.

Другой пример: пациенту требуется консультация специалиста, работающего в подразделении другой компоненты (пациенту хирургического стационара требуется консультация специалиста, работающего в поликлинике). В этом случае пользователю должна предоставляться возможность выбора специалиста, приписанного к другой компоненте.

Таким образом, механизм поддержки многокомпонентности МИС должен ограничивать работу пользователя рамками его компоненты, так как для повседневной работы пользователю нет необходимости видеть ресурсы и данные других компонент ЛПУ. Но для случаев, когда требуется доступ к ресурсам другой компоненты, пользовать должен иметь возможность кратковременно снимать ограничения на структурную компоненту, накладываемые функциональностью модуля.

Принципы реализации механизма поддержки многокомпонентности

Реализация описанной бизнес-логики в клиентских модулях нецелесообразна, так как усложняет клиентские модули и требует существенной доработки большого количества уже реализованных подсистем. Таким образом, для удовлетворения выдвигаемых требований при разработке корпоративной интегрированной МИС возможны два подхода, реализуемые на уровне СУБД:

1) создание динамических представлений и работа с таблицами данных через эти представления;

2) использование технологии виртуальных БД.

Оба метода требуют реализации контекста приложения, в котором задавались бы правила фильтрации данных, доступных пользователю для работы.

При первом методе правила встраиваются в динамические представления и вся работа приложения с таблицами перенастраивается на работу с использованием созданных динамических представлений.

Второй метод основан на том, что SQL-запросы пользователей (любое обращение к данным – *insert*, *update*, *delete*, *select*) к таблицам БД автоматически модифицируются с помощью соответствующих правил защиты, накладываемых посред-

ством динамически вычисляемой декларации *where*. Такая декларация вырабатывается специальной функцией, реализующей правила защиты; это могут быть любой предикат, выражение или некая формула, возвращаемая функцией. В СУБД *Oracle 8i* имела место такая особенность работы модуля вычисления правила: система кэшировала значения, получившиеся в результате вызова функции, реализующей правила защиты данных, и впоследствии могла выдавать сохраненный результат вместо очередного вызова функции. В СУБД *Oracle 9i* данная особенность работы базового ПО устранена, и вызов функции-правила происходит каждый раз при обращении к защищенному объекту.

К недостаткам подобного решения можно отнести то, что такая функциональность доступна только в СУБД класса *Enterprise Edition*, стоимость лицензии которой заметно выше стандартной.

Реализация механизма поддержки многокомпонентности в МИС Интерин PROMIS

В качестве примера промышленной реализации общесистемного механизма поддержки совместной работы МИС в мультипликативных структурах ЛПУ рассматривается МИС Интерин PROMIS – МИС масштаба крупного предприятия, представляющая собой типовое решение при информатизации медицинских учреждений. Механизм поддержки многокомпонентности в составе МИС Интерин PROMIS запущен в промышленную эксплуатацию в ряде крупных отечественных ЛПУ. Опыт его использования позволяет делать выводы о правильности избранной концепции и примененных технологических решений.

Для разметки данных и получения возможности отслеживать их принадлежность к той или иной структурной компоненте ЛПУ было принято решение добавить к сущностям, хранимым в БД МИС Интерин PROMIS, атрибут «компонента», предназначенный для задания компоненты, к которой принадлежит тот или иной экземпляр этой сущности.

Для ограничения отображаемых пользователю ресурсов был введен параметр «область видимости», задающий набор компонент, ресурсы которых могут отображаться пользователю. Для уменьшения времени, затрачиваемого на вычисление множества компонент, входящих в область видимости, на регистрацию содержимого наложено следующее ограничение: область видимости может содержать в себе только компоненты, но не другие области видимости.

Данное ограничение на действие «содержат» может быть записано в следующей форме:
СОДЕРЖИТ=<ОБЛАСТЬ ВИДИМОСТИ, КОМПОНЕНТА>

Для хранения пользовательских настроек был реализован контекст приложения, в котором в виде пар {КЛЮЧ, ЗНАЧЕНИЕ} хранятся настройки для каждого пользователя. Для удобства работы с хранимыми значениями используется программный интерфейс доступа к переменным контекста. Функция извлечения значения переменной контекста вызывается из различных хранимых процедур, пакетов и клиентских модулей, поэтому возникла задача ускорения работы данной функции.

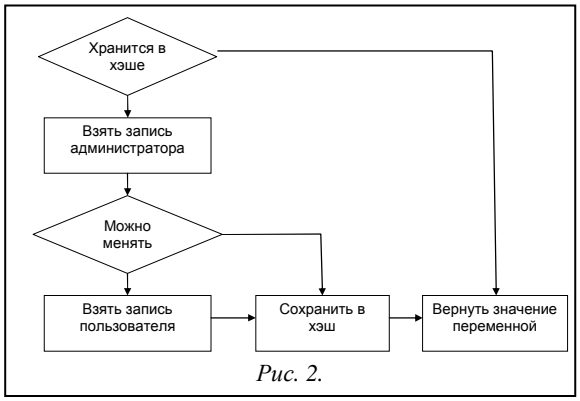


Рис. 2.

В результате проведенной оптимизации логику работы функции получения значения переменной контекста многокомпонетности можно представить в виде схемы, изображенной на рисунке 2.

Для решения задачи отсечения данных, отображаемых пользователю, были совмещены оба описанных метода, итогом стала гибкая система.

В результате анализа структуры МИС принято решение о необходимости разметки лишь ограниченного набора таблиц, а не всех объектов ее БД. Разметке подверглись единые справочники системы: единое хранилище медицинских карт и единый справочник подразделений комплексного ЛПУ.

При создании записи в указанных справочниках ИС автоматически записывает значение переменной контекста пользователя «компонента» в соответствующий атрибут справочника.

На клиентском уровне были реализованы редактор компонент и областей видимости, пользовательская форма настройки переменных контекста и редактор контекста многокомпонентности, используемый администратором МИС.

Кроме системных модулей, были модифицированы модули, используемые пользователями для работы с данными: редактор титульных листов медицинских карт, форма регистрации переводов пациентов по отделениям.

Для подключения новой функциональности были откорректированы статистические отчеты, позволяющие получать данные о работе конкретной компоненты комплексного ЛПУ и комплексного ЛПУ в целом.

Возможности механизма поддержки многокомпонентности

Созданный механизм поддержки совместной работы МИС в мультипликативных структурах ЛПУ характеризуется следующими отличиями.

Имеется контекст механизма многокомпонентности. Для каждого пользователя хранится набор переменных, используемых механизмом поддержки многокомпонентности. Каждая переменная хранится в двух экземплярах: значение, проставленное администратором, и значение, заданное пользователем в процессе работы. Благодаря этому у администратора всегда есть возможность восстановить значение любой переменной контекста.

Имеется системный механизм поддержки многокомпонентности. На сервере БД реализован механизм фильтрации данных, выдаваемых пользователю по его запросу. Параметры ограничения данных хранятся в контексте каждого пользователя. При вычитывании параметров проверяется возможность смены значения переменной пользователем. Если пользователю не разрешено менять значение, берется значение переменной, заданное администратором системы.

Имеется редактор переменных контекстов пользователей. У администратора есть возможность управления контекстом пользователей.

В клиентских модулях предоставлена возможность динамического управления настройками пользовательского контекста механизма многокомпонентности. При задании значения переменной контекста в редакторе эти изменения сразу вступают в силу, и пользователь без дополнительных действий видит другой набор данных, ограниченный новыми настройками системы.

Литература

1. Гулиев Я.И., Комаров С.И. Интегрированная распределенная информационная система крупного лечебно-диагностического учреждения: тез. докл. IV междунар. форума. М., 1997. (Стратегии здоровья: информационные технологии и интеллектуальное обеспечение медицины-97).

2. Никитина Галина. Механизм виртуальных частных баз данных в СУБД Oracle. URL: www.oracle.com/ru/oramag/octnov2002/easy_vpd.html (дата обращения: 01.12.2008).

3. The Virtual Private Database in Oracle9iR2. Understanding Oracle9i Security for Service Providers. An Oracle Technical White Paper. January 2002.

4. Сайт Исследовательского центра медицинской информатики Института программных систем РАН www.interin.ru (дата обращения: 01.12.2008).

5. Сайт корпорации Oracle, разработчика СУБД Oracle, www.oracle.com (дата обращения: 01.12.2008).

РАЗРАБОТКА ТЕМПОРАЛЬНОЙ МОДЕЛИ ДАННЫХ В МЕДИЦИНСКОЙ ИНФОРМАЦИОННОЙ СИСТЕМЕ

А.Н. Базаркин (ИПС им. А.К. Айламазяна РАН, г. Переславль-Залесский, bugs@interin.ru)

TEMPORAL DATA MODEL DEVELOPMENT IN A HEALTHCARE INFORMATION SYSTEM

Bazarkin Aleksey N. *(Research Centre for Medical Informatics, Organization of Russian Academy of Sciences Ailamazyan Program Systems Institute of RAS, Pereslavl-Zalessky, bugs@interin.ru)*

Abstract. The paper investigates the basic methods of temporal data models construction. The author describes some criteria of classification of the methods. The paper further discusses the experience of realization of one of temporal models as a part of the MIS Interin PROMIS integrated subsystem.

Keywords: temporality, temporal data model, temporal data base, medical information system, information system architecture, medical diagnostic process.

В статье исследуются основные методы построения темпоральных моделей данных в реляционных СУБД. Приводится несколько критериев классификации методов их построения, а также обобщается опыт реализации одной из темпоральных моделей в интегрированной подсистеме МИС Интерин PROMIS.

Ключевые слова: *темпоральность, историчность, темпоральная модель данных, темпоральные БД, здравоохранение, медицинские информационные системы, лечебно-профилактические учреждения.*

В отличие от традиционных моделей данных, обеспечивающих хранение лишь мгновенного снимка объектов предметной области, темпоральные модели данных позволяют хранить информацию об эволюции объектов: для любого объекта, который был создан в момент времени **T1** и закончил свое существование в момент времени **T2**, в БД будут сохранены все его состояния на временном интервале **[T1, T2]**.

Под *темпоральностью* объекта следует понимать явную или неявную связь объекта с определенными датами или промежутками времени. В самом широком смысле темпоральные данные – это данные, которые могут изменяться с течением времени.

На рынке коммерческих БД отсутствуют СУБД, обладающие полноценными темпоральными возможностями.

Для их реализации в рамках *информационной системы* (ИС) программистам, как правило, приходится разрабатывать специальные средства, расширяющие и дополняющие существующие реляционные модели. Весьма распространенной проблемой разработки таких приложений является отсутствие полного понимания того, каким образом и на каком уровне должна поддерживаться

темпоральность в БД.

В соответствии с устоявшимися понятиями *темпоральная модель данных* (ТМД) – это модель данных, ориентированная на хранение темпоральных данных, все аспекты которой также должны быть темпоральными. Традиционная модель данных – **M=(DS, OP, C)** – состоит из трех компонент: структура данных **DS**, операции **OP** и ограничения целостности **C**, а темпоральная – **MT=(DST, OPT, CT)** – должна поддерживать все понятия, входящие в каждый из трех компонент, с учетом изменений данных во времени [1]. Структура данных должна быть адаптирована таким образом, чтобы имелась возможность хранить темпоральные данные. Алгебру и операции модификации следует переопределить, используя темпоральную семантику. Дополнительно для каждого ограничения целостности в нетемпоральной модели данных **M** темпоральная – **MT** – должна поддерживать темпоральный аналог нетемпорального ограничения. Семантика темпоральных ограничений целостности также должна быть переопределена.

Таким образом, разработка ТМД предполагает развитие следующих темпоральных составляющих: структура данных, ограничения целостности и ключи, запросы и модификации, алгебра.

Если рассматривать данные, представленные в БД, как некое отражение текущего состояния действительности для моделируемого мира, каждая запись может быть воспринята как факт, являющийся истинным в определенный момент или интервал времени. При переходе к темпоральной БД для каждого факта можно указать тот промежуток времени, в который этот факт являлся истинным в моделируемом мире, представленном в БД. Подобное представление времени, когда с данными связывается промежуток времени их актуальности (с точки зрения моделируемого мира), называется модельным, или действительным (*valid*) временем.

Другим типом линии времени, который рассматривается исследователями темпоральных БД, является транзакционное время. В любой СУБД каждой записи БД можно сопоставить тот промежуток времени, когда данная запись была представлена в БД, то есть промежуток времени между моментами добавления записи и ее удаления из БД.

Исследователями выделяются три фундаментальных типа темпоральных данных [1]:

- момент времени (*instant*) (событие, которое произошло или произойдет в определенный момент, например, сейчас или 1 августа 2009 года в 13.40);
- интервал времени (*interval*) (длительность временного отрезка, например 2 года);
- период времени (*period*) (конкретный отрезок времени, например, с 23 апреля 2007 года по 1 августа 2009 года).

Битемпоральная модель данных оперирует как модельным, так и транзакционным временем. Именно битемпоральная модель наиболее востребована в большинстве ИС.

Методы представления данных

Модифицирование реляционной модели данных для обеспечения поддержки работы с темпоральными данными предполагает изменения модели на уровне СУБД. Однако устройство большинства СУБД – это *черный ящик*, изменения в котором не представляются возможными. Поэтому основные способы обеспечения поддержки темпоральных данных заключаются в поддержке темпоральной функциональности на уровне приложения либо расширение реляционной модели данных до темпоральной.

На практике существуют два принципиальных подхода к реализации ТМД: реализация темпоральной поддержки на уровне приложения и расширение нетемпоральной модели данных до темпоральной.

В литературе встречаются и другие способы, такие, как генерализация модели данных до темпоральной и использование абстрактных типов, однако на практике использование этих подходов сопряжено с определенными сложностями [2]. Метод реализации темпоральности на уровне приложения предполагает разработку специальных средств поддержки темпоральности на уровне приложения. Однако на практике данный подход приводит к существенным проблемам, например, когда требуется изменить или заменить часть кода в приложении. Темпоральная семантика в таком случае проектируется каждым разработчиком заново. Темпоральная логика, реализованная на уровне приложения, может быть удобным сиюминутным решением, но не дальновидной стратегией проектирования ИС.

Расширение нетемпоральной модели данных до ТМД означает, что для спецификации темпоральных понятий используются основные концепции, поддерживаемые нетемпоральной моделью данных. Язык запросов и алгебра расширяются дополнительными операциями для того, чтобы иметь возможность описывать темпоральные операции с данными. На практике этот подход расширения схемы данных наиболее широко используется для построения ТМД. Его преимущество состоит в том, что данный метод предполагает изменение лишь отдельных частей модели, например, языка запросов или ограничения целостности. Метод доступа к информации и структура данных остаются неизменными.

В рамках данного подхода предложены различные ТМД. Критерии принципиальных отличий этих моделей друг от друга следующие:

- тип темпоральных данных (дискретное или интервальное представление времени);

• обеспечение темпоральности на уровне отдельных атрибутов или на уровне кортежа.

Темпоральные данные могут быть связаны как с дискретным представлением времени – моментом, так и с интервальным. Преимущество модели, основанной на дискретном представлении, заключается в ее простоте с точки зрения поддержки стандарта SQL-92. Однако связь темпоральных объектов с одним атрибутом времени может усложнить и без того непростые темпоральные запросы и операции. В этом плане проще в реализации модель с интервальным представлением времени. Одним из недостатков этого подхода является отсутствие поддержки понятия интервала в стандарте SQL-92, оно может быть смоделировано использованием двух моментов времени.

Второй критерий построения ТМД определяет следующие пять подходов к представлению темпоральных данных [2].

1. Модель представления темпоральных данных, предложенная Р. Снодграсом.

Пусть битемпоральное отношение **R** имеет набор атрибутов **(A1, ..., An, T)**, где **T** – битемпоральный атрибут, определенный на множестве битемпоральных элементов. Тогда **R** можно записать в следующем виде: **R=(A1, …, An, Ts, Te, Vs, Ve)**.

Дополнительные атрибуты **Ts**, **Te**, **Vs**, **Ve** – это атомарные темпоральные атрибуты времени, содержащие дату начала и окончания транзакционного и модельного времени. Данное представление отношений является самым естественным и наиболее часто используемым способом представления битемпоральных отношений.

2. Модель представления темпоральных данных, предложенная К. Дженсеном.

Особенность данного представления в том, что историчные кортежи никогда не обновляются, то есть доступны только для чтения. Таким образом, это представление данных хорошо подходит для основанного на архивах хранения битемпоральных отношений. Битемпоральное отношение **R** с набором атрибутов **A1, …, An** может быть представлено в следующем виде: **R=(A1, …, An, Vs, Ve, T, Op)**.

Как и в предыдущей схеме представления данных, атрибуты **Vs** и **Ve** хранят даты начала и окончания актуальности факта в моделируемой реальности соответственно. Атрибут **T** хранит информацию о времени внесения кортежа в журнал изменений. Запросы на создание и удаление кортежей обозначаются в атрибуте **Op** соответствующими символами – **I** (*Insert*) и **D** (*Delete*). Модификация данных представляет собой пару запросов – удаление и создание записи – с одинаковым атрибутом времени **T**.

3. Модель представления темпоральных данных, предложенная С. Гадией.

Данный подход предполагает наличие битемпоральных меток у каждого из атрибутов кортежа, что обеспечивает возможность более гибкого моделирования реальности. Пусть битемпоральное отношение **R** имеет атрибуты **(A1, ..., An, T)**, где **T** – темпоральный атрибут, определенный на множестве битемпоральных элементов. Тогда битемпоральное отношение **R** может быть представлено в виде отношений, где каждый из атрибутов имеет свою темпоральную метку: **R=({([Ts, Te] [Vs, Ve] A1)},…,{ ([Ts, Te] [Vs, Ve] An)})**.

Кортеж состоит из **n** элементов. Каждый элемент представляет собой тройку значений: транзакционное время **[Ts, Te]**, модельное время **[Vs, Ve]** и значение атрибута **Ai**.

4. Модель представления темпоральных данных, предложенная Е. МакКензи.

В данной модели битемпоральное отношение – это последовательность состояний в модельном времени, проиндексированная транзакционным временем. В кортежах с модельным временем атрибуты имеют свои темпоральные метки. Битемпоральное отношение **R** с набором атрибутов **A1, …, An** может быть представлено в виде отношения, в котором каждый атрибут помечается временной меткой**: R=(VR, T)**, где **VR** –отношение в модельном времени; **T** – транзакционное время. Схема состояний отношения модельного времени имеет вид: **VR=(A1V1, ..., AnVn)**, где **A1, …, An** – набор атрибутов; **Vi** – это атрибут модельного времени, связанный с каждым атрибутом **Ai** и обозначающий время актуальности значения атрибута **Ai** в моделируемой реальности.

5. Модель представления темпоральных данных, предложенная Дж. Бен-Зви.

Пусть битемпоральное отношение **R** состоит из набора атрибутов **(A1, ..., An, T)**, где **T** – темпоральный атрибут, определенный на множестве битемпоральных элементов. Тогда **R** может быть представлено в модели Бен-Зви следующим образом: **R=(A1, …, An, Tes, Trs, Tee, Tre, Td)**.

В кортеже значение атрибута **Tes** (*effective start*) – это время, когда значение атрибута кортежа становится актуальным. Атрибут **Trs** хранит информацию о том, когда **Tes** было сохранено в БД. Аналогично **Tre** хранит информацию о том, когда факт перестает быть актуальным в моделируемой реальности, а **Tee** – когда **Tre** было зафиксировано в БД. Атрибут **Td** указывает на время, когда запись была логически удалена из БД.

Кроме этого, темпоральные модели данных могут быть различны по дополнительным критериям, таким как возможность работы с ошибочно введенными данными. Существует ряд методов, предложенных отечественными авторами, суть которых сводится к расширению традиционной модели до темпоральной посредством введения дополнительных таблиц-связей [3]. Авторы доказывают жизнеспособность и пригодность данных методов на примере разработки прикладного программного обеспечения, однако реализованные

ими темпоральные возможности не являются полноценными с точки зрения определения ТМД.

Реализация темпоральной модели

Опишем реализацию ТМД в *медицинской информационной системе* (МИС) Интерин PROMIS. Система представляет собой информационную и функциональную среду, объединяющую элементы различных классов. МИС обеспечивает комплексную автоматизацию и информационную поддержку всех служб медицинского учреждения. Важное место в ней занимает подсистема управления персоналом, эффективная работа которой является необходимым условием нормального функционирования учреждения [4]. Подсистема управления персоналом предназначена для автоматизации работы с кадровым составом учреждения, в ее функции входит сквозное ведение штатного расписания, а также оформление и проведение в системы приказов по кадрам на дату как в прошлом, так и в будущем [5]. Реализация темпоральности в данной подсистеме имеет ряд системотехнических сложностей, для решения которых потребовалось принятие научно-обоснованных решений архитектурного и методологического характера.

Опишем некоторые моменты реализации ТМД за счет расширения существующей реляционной модели. Автором сознательно выбраны наиболее важные аспекты реализации темпоральности, которые, по его мнению, заслуживают наибольшего внимания.

Выбор метода. В качестве основной модели структуры темпоральных данных выбрана модель, предложенная Р. Снодграсом и реализующая темпоральные характеристики на уровне кортежа. Данный метод выбран из-за наибольшей естественности и простой реализации по сравнению с другими [2].

Модель данных подсистемы управления кадрами представляет собой набор более чем из тридцати таблиц, двенадцать из которых темпоральные. Рассмотрим ТМД на примере четырех таблиц: K_PERSONS, K_ISPOLS, K_DOLS и K_DOL_DICT. Таблица K_PERSONS содержит тридцать четыре поля, для примера возьмем только некоторые из них: таблицу K_ISPOLS, состоящую из двадцати восьми полей, включая PERSON_ID и DOL_ID; таблицу K_DOLS, содержащую восемнадцать полей, включая DOL_ID и DOL_DICT_ID; таблицу K_DOL_DICT, в которой два поля.

Таким образом, рассмотрим следующие таблицы и атрибуты:

- K_PERSONS (PERSON_ID, FAMILY, NAME);
- K_ISPOLS (PERSON_ID, DOL_ID, SALARY);
- K_DOLS (DOL_ID, DOL_DICT_ID);
- K_DOL_DICT (DOL_DICT_ID, DOL_NAME).

Таблица K_PERSONS содержит информацию о сотрудниках организации; K_DOLS – список должностей организации; K_ISPOLS – информацию о том, какую должность занимает сотрудник; K_DOL_DICT – это справочник должностей. PERSON_ID – *первичный ключ* (ПК) таблицы K_PERSONS, ПК таблицы K_ISPOLS состоит из пары атрибутов (PERSON_ID, DOL_ID), ПК таблицы K_DOLS – DOL_ID.

Добавление темпоральности. Для того чтобы ТМД стала битемпоральной, в таблицы были добавлены по четыре темпоральных атрибута. Первая пара атрибутов (ACTUAL_FROM, ACTUAL_TO) отражает период актуальности информации в моделируемой реальности (модельное время), а вторая пара (IN_DATE, OUT_DATE) – время фактической регистрации факта в БД и время его логического удаления (транзакционное время).

Темпоральные ключи. Для таблиц с темпоральной поддержкой требуется изменить состав первичных ключей – в них необходимо включить темпоральные атрибуты. Значение первичного ключа таблицы должно быть уникальным. Для оригинальной таблицы K_ISPOLS значение пары (PERSON_ID, DOL_ID) уникально в любой момент. Это означает, что ни один из сотрудников не может числиться более чем на одной должности (речь идет об основных видах исполнения). С добавлением темпоральной поддержки в данной таблице пары значений (PERSON_ID, DOL_ID) могут повторяться. Добавление темпорального атрибута ACTUAL_FROM или ACTUAL_TO в состав ключа не решает проблемы темпоральных ключей. Проблема остается в том случае, когда даты модельного времени отличаются, например, на один день, то есть периоды актуальности данных пересекаются, что приводит к неуникальности значений первичного ключа на некоторых промежутках времени. Для решения этой задачи была разработана более сложная конструкция – последовательное (*sequenced*) условие уникальности в каждый момент времени:

```
WHERE NOT EXISTS (SELECT *
    FROM K_ISPOLS I1,
         K_ISPOL I2
    WHERE I1.PERSON_ID = I2.PERSON_ID
    AND I1.DOL_ID = I2.DOL_ID
    AND I1.ACTUAL_FROM < I2.ACTUAL_TO
    AND I2.ACTUAL_FROM < I1.ACTUAL_TO
    AND I1.rowid <> I2.rowid )
    AND NOT EXISTS (
        SELECT *
          FROM K_ISPOLS I1
         WHERE I1.K_PERSON_ID IS NULL OR
               I1.DOL_ID IS NULL)
```

Темпоральная уникальность. Для оригинальной таблицы K_ISPOLS условие уникальности может быть записано в виде UNIQUE (PERSON_ID, DOL_ID). Однако с добавлением темпоральной поддержки в таблицы этого условия оказывается недостаточно. Недостаточным является и добавление одного темпорального атрибута или

пары атрибутов UNIQUE (PERSON_ID, DOL_ID, ACTUAL_FROM, ACTUAL_TO), поскольку пара тех же значений (PERSON_ID, DOL_ID) может быть добавлена с датами, отличающимися, например, на один день. В этом случае условие уникальности выполняться не будет. Для решения этих проблем были разработаны более сложные условия уникальности следующего вида:

```
WHERE NOT EXISTS (SELECT *
      FROM K_ISPOLS AS I1
     WHERE 1 < (SELECT COUNT(PERSON_ID)
           FROM K_ISPOLS AS I2
          WHERE I1.PERSON_ID = I2.PERSON_ID
            AND I1.DOL_ID = I2.DOL_ID
            AND I1.ACTUAL_FROM <= CURRENT_DATE
            AND CURRENT_DATE < I1.ACTUAL_TO
            AND I2.ACTUAL_FROM <= CURRENT_DATE
            AND CURRENT_DATE < I2.ACTUAL_TO)))
```

Темпоральные ограничения целостности. Добавление темпоральной поддержки порождает три возможных случая ограничения ссылочной целостности: ни одна из двух таблиц не темпоральна, одна из двух таблиц темпоральна, обе таблицы темпоральны.

Наиболее сложным случаем реализации условия ограничения целостности данных является третий случай, когда в обеих таблицах реализована темпоральная поддержка:

```
NOT EXISTS ( SELECT *
    FROM K_ISPOLS AS I
   WHERE NOT EXISTS (
       SELECT *
         FROM K_DOLS AS P
         WHERE I.DOL_ID = P.DOL_ID
          AND P.ACTUAL_FROM <= I.ACTUAL_FROM
          AND I.ACTUAL_FROM < P.ACTUAL_TO)
     OR NOT EXISTS (
       SELECT *
        FROM K_DOLS AS P
          WHERE I.DOL_ID = P.DOL_ID
         AND P.ACTUAL_FROM < I.ACTUAL_TO
         AND I.ACTUAL_TO <= P.ACTUAL_TO)))
```

Темпоральные запросы. В теории темпоральных БД выделяют три фундаментальных типа запросов и модификаций: текущие (*current*), последовательные (*sequenced*), произвольные (*non sequenced*) [1].

Запросы к оригинальным таблицам до добавления темпоральной поддержки соответствуют текущему состоянию моделируемой реальности. Запросы к битемпоральным таблицам приобретают специфику. Рассмотрим более подробно различные типы запросов.

Текущий запрос представляет собой запрос значений на какой-нибудь момент в прошлом, то есть создание среза истинности фактов на произвольную дату. Например, для обычного реляционного запроса «какую зарплату сейчас получает каждый из сотрудников?» можно легко сформулировать его темпорального двойника «какую зарплату получал каждый из сотрудников в указанную дату?». В этом случае результат запроса останется в рамках реляционного представления.

Более сложным случаем являются ***последовательные запросы***. Вполне естественным оказывается запрос «когда и какую зарплату получал каждый из сотрудников?». Здесь уже в результатах запроса появляется линия времени. Алгоритм формирования результатов подобных запросов можно упрощенно представить следующим образом: для каждого момента вычисляется реляционный подзапрос «какую зарплату получает каждый из сотрудников?», после чего к общему результату добавляются результаты этих подзапросов с учетом интервалов истинности. Подобная семантика последовательной интерпретации реляционных запросов называется последовательной.

Для случая последовательных запросов рассмотрим более подробно особенности операций выборки и связывания.

Последовательная выборка данных не требует особых средств поддержки темпоральности и достаточно просто реализуется.

Более сложной задачей является связывание двух темпоральных таблиц. Например, для вычисления зарплаты и должности для каждого сотрудника нужно вычислить значение заработной платы для каждого промежутка времени.

История заработной платы хранится в таблице K_ISPOLS. Прямое связывание двух таблиц K_ISPOLS и K_PERSONS для каждого момента неэффективно, потому что значения заработных плат и информация о сотрудниках остаются неизменными в течение длительных периодов. Поэтому данные таблицы следует связывать с использованием их темпоральных периодов. Возможны два случая:

- период в первой из двух связываемых таблиц является более продолжительным, чем период во второй таблице;
- период во второй из двух связываемых таблиц является более продолжительным, чем период в первой таблице.

Таким образом, выборки темпоральных данных в случае, когда все таблицы имеют темпоральную поддержку, становятся более сложными и состоят, как правило, из нескольких подзапросов в зависимости от пересечений и наложений периодов темпоральности. Так, например, запрос на выборку истории зарплат всех сотрудников потребует рассмотрения двух общих случаев наложения периодов таблиц в одном запросе.

Темпоральные модификации

Текущие модификации в общем случае предполагают обновление записи и изменение периода актуальности с некоторого момента в прошлом и по настоящее время [1].

Операции создания записи в этом случае требуют дополнительных явных условий проверки в блоке WHERE на ограничения ссылочной целостности и уникальности.

В общем случае текущее удаление записи представляет собой обновление даты окончания периодов модельного и транзакционного времени.

Текущее обновление записи в общем случае состоит из следующей последовательности действий:

- создание новой записи с соответствующими битемпоральными атрибутами;
- обновление всех записей, дата актуальности которых больше даты начала действия созданной записи.

Последовательные модификации предполагают обновление записи в некоторый период в прошлом, то есть границы актуальности этого периода на оси модельного времени располагаются до настоящего момента [1].

Новая запись в случае последовательных модификаций создается при выполнении следующих условий:

- в этом периоде актуальности нет дубликатов записи;
- для этого периода есть актуальное значение в таблице, на которую ссылается новая запись;
- нет разрывов во временной оси модельного времени.

При удалении записи возможны четыре варианта пересечения периодов актуальности – *оригинального* (ОП) и *удаляемого* (УП): УП целиком входит в ОП, УП начинается в рамках ОП, УП заканчивается в рамках ОП и ОП целиком входит в УП. В общем случае удаление записи при последовательных модификациях состоит из следующих действий:

- копируется запись ОП, атрибут ACTUAL_FROM задается равным дате окончания УП;
- для этой записи атрибут ACTUAL_TO задается равным дате начала УП.
- для этой записи атрибут ACTUAL_FROM задается равным дате окончания УП;
- удаляются записи ОП, которые целиком входят в период УП.

В случае обновления записи также возможны четыре варианта взаимного расположения двух периодов – *оригинального* (ОП) и *модифицируемого* (МП). Обновление записи при последовательных модификациях представляет собой следующий набор операторов:

- копируется запись ОП, атрибут ACTUAL_TO задается равным дате начала МП;
- копируется запись ОП, атрибут ACTUAL_FROM задается равным дате окончания МП;
- обновляются необходимые атрибуты у тех записей, период актуальности которых пересекается с МП;
- атрибут ACTUAL_FROM задается равным дате начала МП для тех записей, период актуальности которых пересекает МП;
- атрибут ACTUAL_TO задается равным дате окончания МП для тех записей, период актуальности которых пересекает МП.

Запросы и модификации произвольного типа оперируют темпоральными данными произвольно, являются достаточно редкими и должны рассматриваться отдельно в конкретном случае. Например, к таким запросам можно отнести запросы, требующие сравнения нескольких последовательных моментов времени, обычно включающие агрегационные функции «во времени», например, «вывести среднюю заработную плату сотрудника за все периоды времени» [1]. Запросы данного типа при реализации ТМД в рамках системы управления кадрами МИС Интерин PROMIS не рассматривались.

В заключение следует отметить, что исследования в области темпоральных БД ведутся уже более трех десятилетий и до сих пор остаются актуальными и востребованными. За это время было сформулировано множество методик построения темпоральных БД, предложено множество различных способов построения ТМД. Функциональные возможности ИС, разработанной на базе темпоральной БД, поднимаются на качественно новый уровень. Практически все данные, которыми оперируют ИС, являются темпоральными, то есть в той или иной мере связаны с динамикой изменения во времени. Корректная и оперативная работа с темпоральными данными приобретает особое значение в сфере медицинских информационных технологий, где качество информации в конечном счете может оказывать влияние на здоровье человека.

В данной работе сделан обзор основных темпоральных структур данных. На основе наиболее естественного и простого из них реализована ТМД в рамках подсистемы управления персоналом МИС Интерин PROMIS. Подсистема управления персоналом является интегрированной частью МИС Интерин PROMIS, где особенно востребованы темпоральные возможности. Изложенный подход построения темпоральной модели успешно реализован, а также апробирован в ЦКБ РАН.

Литература

1. Snodgrass R. Developing Time-Oriented Database Applications in SQL – Morgan Kaufmann Publishers, 1999.

2. Jensen C. S., Soo M. D., Snodgrass R. T. Unifying Temporal Data Models Via a Conceptual Model, Information Systems Vol. 19, No. 7, 1994, pp. 513–547.

3. Порай Д.С., Соловьев А.В., Корольков Г.В. Реализация концепции темпоральной базы данных средствами реляционной СУБД: сб. науч. тр. / Ин-т системного анализа РАН; под ред. В.Л. Арлазарова, Н.Е. Емельянова. М.: Едиториал УРСС, 2004. С. 92–109.

4. Назаренко Г.И., Гулиев Я.И., Ермаков Д.Е. Медицинские информационные системы: теория и практика; под ред. Г.И. Назаренко, Г.С. Осипова. М.: Физматлит, 2005. 320 с.

5. Базаркин А.Н., Хаткевич М.И., Хаткевич Ю.И. Подсистема управления кадрами в интегрированных медицинских информационных системах: тр. Междунар. конф. Переславль-Залесский: ИПС РАН, Переславль-Залесский, 2006. Т. 1, С. 113–124. (Программные системы: теория и приложения).

РАСПРЕДЕЛЕННЫЙ АРХИВ ИЗОБРАЖЕНИЙ ДИСТАНЦИОННОГО ЗОНДИРОВАНИЯ ЗЕМЛИ

(Работа выполнена при поддержке Программы № 1 фундаментальных исследований Президиума РАН «Проблемы создания национальной научной распределенной информационно-вычислительной среды на основе развития GRID-технологий и современных телекоммуникационных сетей» и проекта РФФИ 07-07-12038-офи)

А.А. Московский, к.х.н.; Е.О. Тютляева; А.Ю. Первин; Е.В. Шевчук
(ИПС им. А.К. Айламазяна РАН, г. Переславль-Залесский,
moskov@phys069b-2.chem.msu.ru, xgl@pereslavl.ru)

DISTRIBUTED REMOTE SENSING DATA STORAGE

Moskovsky Alexander A., Ph.D.; Tyutlyaeva Ekaterina O.; Pervin Artem Yu.; Schevchuk Elena V.
(PSI RAS, Pereslavl-Zalessky, moskov@phys069b-2.chem.msu.ru, xgl@pereslavl.ru)

Abstract: Approach to implementation of distributed remote sensing data storage is described. The approach based on active storage concept. Consolidation both FS LUSTRE and T-Sim parallel library ensure sensing data processing on storage node, it decrease network data transfer and enhance processing performance.

Keywords: *distributed data storage, active storage, remote sensing, cluster file system, distributed data processing.*

Описывается подход к созданию распределенного архива данных дистанционного зондирования Земли, реализующий концепцию активного хранилища. Интеграция параллельной файловой системы Lustre и системы автоматического динамического распараллеливания (T-Sim) обеспечивает обработку данных ДЗЗ непосредственно на узлах хранения, позволяя сократить расходы на пересылку данных по сети и повысить эффективность обработки.

Ключевые слова: распределенный архив, активное хранилище, дистанционное зондирование Земли, кластерная файловая система, распределенная обработка данных.

Дистанционное зондирование Земли (ДЗЗ) – область человеческой деятельности, тесно связанная с обработкой больших объемов информации при помощи компьютерной техники. При этом накапливаются архивы данных по наблюдениям от одного или нескольких приборов, установленных на искусственных спутниках Земли. Объем архивов может составлять несколько петабайт (10^{15} байт) информации.

Эффективное управление возрастающими объемами данных до сих пор остается значительной проблемой. Рост емкости хранилищ только обостряет проблему большой стоимости перемещения данных между обрабатывающими узлами и устройствами хранения. В последние годы были разработаны *файловые системы* (ФС) для решения проблемы управления данными в контексте высокопроизводительных вычислительных систем. Некоторые из таких ФС, например *Lustre* [1] и PVFS, используют выпускаемые промышленные серверы в качестве северов ввода/вывода, то есть мест для хранения данных. Общая вычислительная мощность сотен и тысяч узлов хранения может быть очень значительной, но обычно она не используется, так как этим узлам отводится только роль хранилищ.

Один из подходов по снижению требований к пропускной способности между хранилищами и вычислительными устройствами и задействованию вычислительных мощностей устройств хранения – это приближение вычисления к местам хранения данных. Такой подход позиционируется как активные хранилища в контексте параллельных ФС. Обработка данных непосредственно на узлах хранения существенно снижает объемы передачи данных по сети и, следовательно, общесетевой трафик.

Активные хранилища нацелены на приложения с интенсивной стадией ввода/вывода, входные данные для которых можно разбить на независимые наборы. Они могут использоваться для обработки результатов моделирования в различных научных областях. Задачи обработки и хранения данных ДЗЗ имеют все свойства, позволяющие говорить о том, что реализация для их решения концепции активных хранилищ является прекрасным выбором.

Использование распределенных активных хранилищ позволяет создавать масштабируемые, высокоскоростные, управляемые распределенные системы с высокой пропускной способностью.

Предлагаем один из возможных подходов к организации активного хранилища, реализованный в программной системе для распределенного хранения и обработки данных ДЗЗ. Описываемая программная система объединяет возможности параллельной ФС *Lustre* и системы автоматического динамического распараллеливания (библиотеки *T-Sim* [2]), позволяя создать прототип распределенного архива изображений – активного хранилища, в котором данные обрабатываются на тех же узлах, где хранятся, и обеспечивается автоматическая балансировка нагрузки на узлах кластера.

Технология активных хранилищ в последние годы приобрела особую популярность, что свиде-

тельствует о ее актуальности. Существуют схожие по задачам системы, такие как *ActiveStorage*, *Cascading* [3], *Pig* [4], *Hadoop* [5]. Большинство разработок базируются на какой-либо специализированной кластерной файловой системе, удобной для организации активного хранилища. Предложенный авторами подход не является исключением – данная разработка использует ФС *Lustre*. Отличительной особенностью подхода является наличие шаблонов задач для различных стратегий распараллеливания, специализированных для обработки данных ДЗЗ, что позволяет повысить эффективность решения задач этого типа.

Структура программной системы

Разработанная программная система для хранения и обработки данных ДЗЗ состоит из двух компонент, которые отвечают за выполнение основных функций активного хранилища:

- кластерная ФС *Lustre*, которая отвечает за хранение данных и распределение их по узлам; при обращении к прикладному программному интерфейсу ФС *Lustre* можно получить информацию о расположении данных;

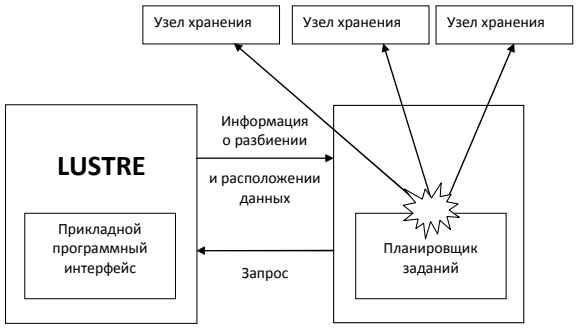


Схема взаимодействия основных компонент распределенного архива изображений

- программный комплекс, базирующийся на библиотеке шаблонных классов C++ *T-Sim*, позволяющий проводить эффективную обработку данных, расположенных в ФС *Lustre*, путем направления вычислений на узлы, хранящие обрабатываемые данные.

На рисунке изображена упрощенная схема взаимодействия основных компонент системы.

В общем случае обработка данных ДЗЗ в описываемой программной системе происходит по следующей схеме.

- Задаем разбиение для отдельного файла или для отдельной директории в ФС *Lustre*, наиболее отвечающее особенностям решаемой задачи.
- Планировщик заданий *TSim* вызывает функцию прикладного программного интерфейса ФС *Lustre*, которая возвращает, помимо другой информации, число и размер порций (*stripes*), на которые разбит файл для того, чтобы в соответствии с идеологией активного хранилища произвести обработку данных на узле, где они расположены. Таким образом, определяется схема расположения файла в кластере.
- Определяем, зная стратегию, которая используется *Lustre* для назначения порций файла (стрипов) на узлы кластера, какой стрип на каком узле кластера будет находиться. IP-адреса всех узлов кластера (по номерам узлов) содержатся в служебной информации *Lustre*. Планировщик *T-Sim* направляет задание на обработку стрипа на IP-адрес, который соответствует номеру узла кластера для данного стрипа.

Шаблоны хранения и обработки данных

Информация, полученная от спутников ДЗЗ, может храниться в файлах различных форматов и размеров, что влияет на выбираемые способы обработки данных и разбиение для хранения в ФС *Lustre*. Кроме того, алгоритмы обработки могут сильно отличаться по требовательности к ресурсам (процессорному времени, оперативной памяти). В связи с этим были разработаны два шаблона задач, реализующих различные подходы к хранению и обработке изображений. Для каждого шаблона определены свойства, которыми должна обладать решаемая задача, чтобы применение шаблона было целесообразно. Для каждого шаблона также был реализован пример соответствующей задачи обработки данных ДЗЗ.

Шаблон задачи 1. Данный шаблон предоставляет подход к решению задач по обработке изображений, которые обладают следующими свойствами: входные данные содержатся в файлах достаточно большого размера; формат описания данных – TIFF; вычислительная сложность задачи достаточно высока, общий алгоритм обработки можно представить в виде нескольких независимых потоков, так что затраты на распараллеливание вычислений несущественны по сравнению с затратами на обработку одной порции данных.

При реализации этого шаблона задачи были использованы следующие подходы:

- использована библиотека *ltiff* для получения дополнительной информации о разбиении данных в TIFF-файле (разбиение на полоски формата TIFF);
- файл с данными разбивается для хранения на различных узлах кластера с помощью программного интерфейса ФС *Lustre*;
- обработка данных пополосочно – планировщик библиотеки *T-Sim* направляет задание на обработку данных на тот узел, на котором расположена обрабатываемая в данный момент полоска;
- использование для обработки изображения шаблона параллельного программирования *Map* (отображение) из библиотеки *T-Sim*;
- распараллеливание алгоритма обработки изображения происходит автоматически с помо-

щью библиотеки *T-Sim* (для этого алгоритм обработки должен быть модифицирован – в него необходимо внести изменения, связанные с использованием шаблонов библиотеки *T-Sim*).

Примером использования данного шаблона является задача контролируемой классификации изображения ДЗЗ при помощи метрики Махаланобиса.

Шаблон задачи 2. Данный шаблон предоставляет подход к решению задач по обработке изображений, которые обладают следующими свойствами: имеется последовательный код обработки изображения, который нужно многократно запускать для различных наборов данных; данные хранятся в файлах небольших размеров, так что обработка файла целиком на одном узле более эффективна, чем разбиение его на порции; формат данных не имеет значения.

При реализации этого шаблона задачи файл с данными целиком располагается на узле, его обработка происходит там же; последовательный код обработки отдельного изображения не подвергается никакой модификации, реализуется параллелизм по данным. Файлы равномерно распределены по узлам кластера при помощи средств ФС *Lustre*. При этом планировщик заданий, специально реализованный для данного шаблона с помощью библиотеки *T-Sim*, используется для направления вычислений на тот узел кластера, на котором расположен обрабатываемый файл.

Отличительной особенностью данного шаблона является то, что последовательный код обработки изображения, как уже было указано, не подвергается никакой модификации.

С применением данного шаблона была реализована задача по перепроецированию данных ДЗЗ.

В заключение можно сделать следующие выводы. Предлагаемая программная система представляет собой распределенный архив данных ДЗЗ, реализованный в соответствии с идеологией активных хранилищ. Эффективность работы системы обеспечивается обработкой данных на тех же узлах распределенного хранилища, на которых они хранятся, что позволяет существенно снизить расходы на передачу больших массивов данных по сети. Балансировка нагрузки на узлах осуществляется планировщиком библиотеки *T-Sim* на основе информации, предоставляемой интерфейсом кластерной ФС *Lustre*.

Для решения различных видов задач обработки космических снимков, хранящихся в распределенном архиве, реализованы стратегии распараллеливания, оформленные в виде шаблонов задачи. Применение шаблонов задачи позволяет быстро и эффективно реализовать параллельные вычисления для задач обработки, удовлетворяющих условиям применения шаблона.

В качестве примеров использования предлагаемых шаблонов были реализованы задача контролируемой классификации изображений ДЗЗ при помощи метрики Махаланобиса и задача представления данных ДЗЗ в нужном масштабе и проекции. Тестирование задач, которое проводилось на двух различных установках, продемонстрировало повышение производительности приложения при использовании предлагаемых шаблонов задач, если соблюдались условия применения шаблонов: достаточная вычислительная сложность задачи для шаблона 1 и многократный запуск на различных наборах данных для шаблона 2.

Целесообразность использования описанной программной системы возрастает при работе с большими объемами исходных данных или при большой вычислительной сложности задач обработки изображения, что, впрочем, справедливо для большинства систем, использующих параллельные вычисления.

Литература

1. Sun Microsystems : официальный сайт кластерной файловой системы Lustre. URL: http://wiki.lustre.org/index.php (дата обращения: 10.04.2009).

2. Московский А.А. T-Sim – библиотека для параллельных вычислений на основе подхода Т-системы: тр. Междунар. конф. М.: Наука, Физматлит, 2006. Т. 1. С. 183–193. (Программные системы: теория и приложения).

3. Cascading Home Page : официальная стр. проекта Cascading. URL: www.cascading.org (дата обращения: 10.04.2009).

4. Welcome to Pig! : официальная стр. проекта Pig. URL: http://hadoop.apache.org/pig (дата обращения: 10.04.2009).

5. Welcome to Hadoop! : официальная стр. проекта Hadoop. URL: http://hadoop.apache.org/core (дата обращения: 10.04.2009).

ВЫДЕЛЕНИЕ И АНАЛИЗ СКЕЛЕТОВ ОБЪЕКТОВ НА ЦВЕТНЫХ СНИМКАХ

С.В. Погодин (ИПС РАН, г. Переславль-Залесский, sodzo@mail.ru)

OBJECT SKELETONS ANALYSIS IN COLOUR PICTURES

Pogodin Sergei V. *(Program Systems Institute, Pereslavl-Zalessky, sodzo@mail.ru)*

Abstract. The paper surveys object skeletons allocation and analysis in colour pictures. The author describes object skeletons automatic representation in the form of a weighed graph.

Keywords: *allocation of objects, skeletonization, weighed graph, image recognition.*

Данная работа посвящена исследованию выделения и анализа скелетов объектов на цветных снимках. Описаны методы автоматического представления скелетов объектов в виде взвешенного графа.

Ключевые слова: *выделение объектов, скелетизация, взвешенный граф, распознавание образов.*

Исследования в области распознавания образов ведутся повсеместно и всегда востребованы из-за высокой актуальности различных приложений. Разработано большое количество алгоритмов, однако многие задачи остаются нерешенными. Распознавание образов имеет одну из наиболее часто решаемых и остро стоящих проблем – выделение значимой части потока информации (локальных объектов) и отсеивание ненужной, а иногда и мешающей распознаванию информации. В данной работе предлагаются алгоритмы выделения локальных объектов на цветных снимках и анализа их скелетов для последующего решения задач кластеризации и распознавания.

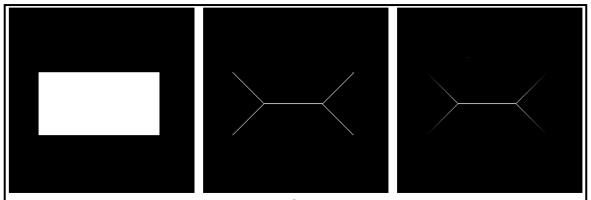
Подготовка изображения

В большинстве случаев объект на изображении находится рядом с другими объектами, которые мешают анализу изображения. В работах [1, 2] рассмотрены некоторые алгоритмы по удалению фона на растровом изображении, по понижению количества шума, выделению локальных объектов и т.п. Основная идея данных работ заключается в применении волнового алгоритма удаления фона на растровых изображениях. Одним из результатов такого подхода к анализу изображений стало выделение объектов из фона с использованием прозрачности пикселей (*alpha*-канала). К сожалению, данный метод не лишен недостатков. Так как исследуемые объекты, как правило, имеют причудливую форму и некоторые их части визуально сливаются с фоном, после обработки отдельные области объектов отделяются и удаляются. Для решения проблемы целостного выделения объектов были разработаны алгоритмы склейки и отсеивания изображений, не отвечающих критерию близости к эталонам. Этим критерием являлось отношение площади выделенного объекта к площади описанного вокруг него круга. На основе дальнейших тестов было выяснено, что данный критерий не универсален и не всегда дает желаемый результат.

Для улучшения алгоритма выделения предлагается учитывать не только размеры объекта, но и его форму, а именно, строить **скелет** объекта и анализировать полученный граф.

Волновой алгоритм для нахождения скелета

Одним из способов нахождения скелета изображения является волновой алгоритм [3, 4]. В нем используется особый вид волны, названный сферическим. Это комбинация последовательного применения 4- и 8-связной волны. В результате волна распространяется в виде восьмиугольника, уверенно огибающего препятствия. Здесь наблюдается интересный эффект: не более чем через **2*N** шагов распространение приобретает устойчивый характер вне зависимости от начальной точки. Поведение волны предсказуемо и на траекториях, отличных от прямой. Огибающие свойства волны имеют свои особенности. Мелкие препятствия (1–2 пикселя) мало влияют на распространение волны, внося незначительные помехи. Более крупные препятствия на изображении вызывают значительные искажения картины распространения волны. Следовательно, необходима предварительная обработка изображения, направленная на устранение нежелательных помех.



a) *b)* *c)*

Рис. 1. Обработка прямоугольника

Алгоритм утоньшения для нахождения скелета

Рассмотрим подходы к нахождению скелета графического объекта, основанные на последовательном утоньшении границ объекта [5, 6]. Предлагаются два алгоритма утоньшения.

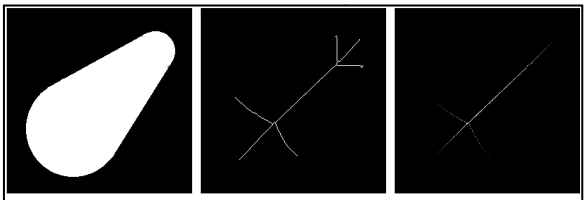
Первый алгоритм основан на применении к изображению восьми масок, полученных из двух (см. табл.) путем поворота каждой на 90, 180 и 270 градусов. Каждая маска отвечает за удаление горизонтальных, вертикальных и диагональных линий, наклоненных под углом 45 градусов.

Маски утоньшения

0	0	0
2	1	2
1	1	1

2	0	0
1	1	0
2	1	2

Алгоритм заключается в следующем. Каждый пиксель изображения и его ближайшие соседи подвергаются тесту на соответствие наложенной

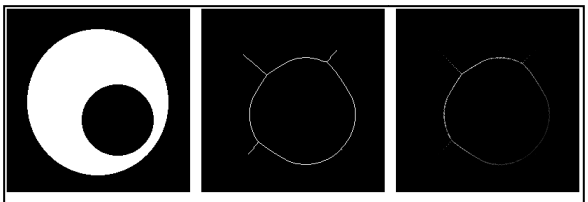


a) *b)* *c)*

Рис. 2. Обработка вытянутого овала

маски. Маска накладывается таким образом, что текущий пиксель становится в ее середину, а ближайшие соседи покрываются остальной частью маски. Если сам пиксель и его соседи удовлетворяют условию, что под ячейкой маски с номером **0** находится фон объекта, а под ячейками с номером **1** – сам объект, то пиксель, лежащий под центральной ячейкой маски, удаляется на изображении.

Второй алгоритм основан на применении к



a) *b)* *c)*

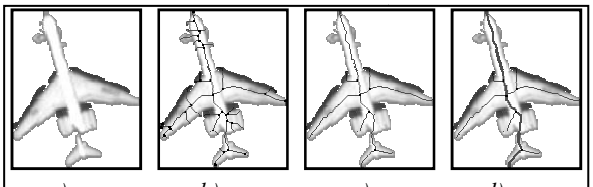
Рис. 3. Обработка объекта с отверстием

изображению взвешенных матриц. Причем вес точки объекта определяется ее яркостью, а яркость каждой точки на скелете объекта убывает пропорционально расстоянию от центра тяжести. Данный алгоритм не подходит в нашем случае, так как скелеты объектов могут распасться на части, что затруднит их дальнейший анализ. На рисунках 1–3 показаны результаты работы указанных алгоритмов. Здесь a) – исходное изображение, b) – результаты работы алгоритма применения восьми ядер, c) – результаты работы алгоритма со взвешенной матрицей.

В итоге был выбран первый алгоритм построения скелета, так как выполненные эксперименты позволяют утверждать, что в результате его применения скелет отдельно взятого объекта получается связным. При дальнейшей обработке скелеты целесообразно упростить, используя дополнительные методы.

Получение и сравнение графов скелетов

Для анализа скелета объекта необходимо представить его в виде взвешенного графа, как это показано на рисунке 4. Для удобства визуализации



a) *b)* *c)* *d)*

Рис. 4. Результаты выделения и анализа скелетов самолетов

получившийся граф наложен на исходное изображение (черные точки – вершины графа, серые – его дуги). Веса дуг определяются количеством пикселей, лежащих на пути передвижения по изображению скелета между узловыми точками. На рисунке отображены результаты выделения и анализа скелетов самолетов, в том числе a) – выделенный самолет без фона, b) – полученный граф скелета, c) – упрощенный граф скелета, d) – самый длинный путь в графе скелета (темно-серая линия).

Алгоритм построения графа скелета состоит из следующих этапов.

1. Изображение со скелетом объекта представляется в виде матрицы, размеры которой соответствуют размерам изображения. Скелет объекта в матрице маркирован цифрами **1**, а фон – цифрами **0**.

2. Находится ячейка матрицы с одним соседом, маркированным цифрой **1**. Если таких ячеек нет, остановка алгоритма.

3. Координаты текущей ячейки заносятся в массив описания дуги.

4. В текущую ячейку ставится цифра **2**.

5. Если у текущей ячейки соседей, маркированных цифрой **1**, не больше одного, координаты текущей ячейки изменяются на координаты соседа. Переход к пункту 3.

6. В массив вершин графа добавляются координаты текущей ячейки.

7. Если у текущей ячейки нет соседей, маркированных цифрой **1**, переход к пункту 10.

8. В ячейки соседей, маркированных цифрами **1**, ставятся цифры **3**.

9. Координаты текущей ячейки заносятся в стек столько раз, сколько было маркировано соседей.

10. Массив точек дуги добавляется к описанию текущей вершины графа.

11. Если стек не пуст, извлекаются координаты ячейки. Переход к пункту 3. Иначе описание скелета добавляется в массив и осуществляется переход к пункту 2.

Алгоритм имеет недостаток. Возможна ситуация, когда появится дуга, в которую входят всего две точки – узловая и конечная. Такой пример от-

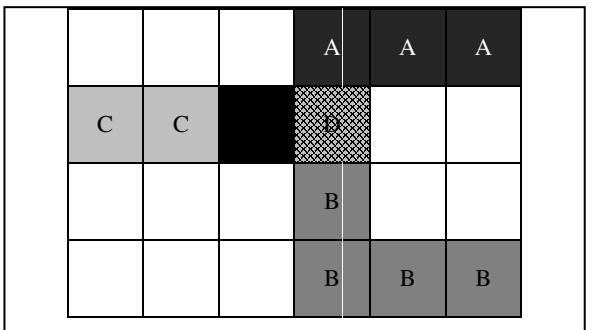


Рис. 5. Слабая сторона алгоритма

ражен на рисунке 5, где показаны четыре пути из узловой (черной) точки, маркированные буквами **A**, **B**, **C** и **D**.

Как видно на рисунке, путь **D** сразу заканчи-

вается. Эта ситуация может осложнить процесс сравнения графов. Решить данную проблему можно путем удаления подобных дуг и связных с ними конечных точек. Поскольку путь через узловую точку прерываться не будет, граф останется связным, а относительный вес взвешенного графа мало изменится.

Для дальнейшего анализа скелета необязательно иметь всю ветвистую структуру. Достаточно выделить самый длинный путь в скелете и связанные с ним относительно большие дуги (пути) в графе. Применительно к летательным объектам в таком случае будут выделены фюзеляж самолета и оба его крыла. По соотношению весов самого длинного пути и связанных с ним дуг можно отсеивать мешающие распознаванию объекты (деревья, мелкие водоемы и др.).

После достаточного упрощения скелетов сравниваются полученные и эталонные графы. Полное соответствие графов совершенно необязательно, так что, если веса дуг или их отношения мало отличаются, графы считаются похожими. Для решения задачи можно применить метод сравнения иерархий графов [7]. Для этого вершины одного графа упорядочиваются относительно какой-нибудь конечной вершины, а для другого графа подбирается подобная комбинация вершин и дуг. Упорядочение вершин визуально можно представить как подвешивание дерева за одну из конечных вершин. Если находится подобная комбинация вершин и дуг, полученный и эталонный объекты схожи по своей структуре.

Рассмотренный метод позволяет достаточно эффективно находить полезную составляющую потока графической информации, а именно, выделять и сравнивать скелеты локальных объектов для их распознавания. Особенность предлагаемого подхода заключает в том, что при анализе учитываются расстояния между узловыми точками выделенного скелета. Это означает, что графы одинаковых объектов, масштабированных и повернутых в любом направлении, будут подобны.

Литература

1. Выделение и распознавание локальных объектов на аэрокосмических снимках / А.Н. Виноградов [и др.] // Авиакосмическое приборостроение. 2007. № 9. С. 39–45.

2. Kalugin F. Pogodin S. Object contour extraction at aerospace photos. Proceedings of the 9th International Conference (Pattern Recognition and Information Processing). Minsk, Belarus: 2007. Vol. 1. pp. 177–182.

3. Клубков И. Применение волнового алгоритма для нахождения скелета растрового изображения, 2004. URL: http://igdrassil.narod.ru/pc/work/Vectorisation.html (дата обращения: 04.09.2008).

4. Шикин Е.В., Боресков А.В. Компьютерная графика. М.: Мир, 1995.

5. Fisher R. Perkins S. Walker A. and Wolfart E. Skeletonization / Medial Axis Transform, 2003. URL: http://homepages.inf.ed.ac.uk/rbf/HIPR2/skeleton.htm (дата обращения: 15.05.2008).

6. Ballard D. and Brown C. Computer Vision: Prentice-Hall, 1982. 8 p.

7. Загоруйко Н.Г. Прикладные методы анализа данных и знаний. Новосибирск: Изд-во Ин-та матем., 1999. 270 с.

ГРАФИЧЕСКИЙ АНАЛИЗ ИНФОРМАЦИИ В СИСТЕМАХ КОСМИЧЕСКОГО НАЗНАЧЕНИЯ

Ю.Г. Емельянова (ИПС им. А.К. Айламазяна РАН, г. Переславль-Залесский, tajra@mail.ru)

GRAPHICAL ANALYSIS OF INFORMATION IN THE SYSTEMS OF SPACE FUNCTION

*Emelyanova Juliya G. (Organization of Russian Academy of Sciences Ailamazyan Program Systems Institute of RAS, Pereslavl-Zalessky, **tajra@mail.ru**)*

Abstract. The paper considers the conception of interface construction's earth station of command measuring system of new generation with intelligence system of cognitive, graphical support which was build in it. The suggested conception consists in development of split-screen technology which can combine the usual data representation with cognitive generalizing and amplifying information. The goal of the paper is the enhancement of efficiency, reliability and quality of operations of ES CMS control.

Keywords: *cognitive graphics, visualization, intelligent interface.*

В работе рассмотрена концепция построения интерфейса наземной станции командно-измерительной системы нового поколения с встроенной интеллектуальной системой когнитивной графической поддержки. Предлагаемая концепция заключается в разработке полиэкранной технологии, сочетающей обычное представление данных с когнитивной обобщающей и уточняющей информацией. Цель работы – повышение эффективности, надежности и качества управления работой наземной станции.

Ключевые слова*: когнитивная графика, визуализация, интеллектуальный интерфейс.*

Для управления *космическими аппаратами* (КА) используется *наземный автоматизированный комплекс управления* (НАКУ) [1]. Одним из основных средств НАКУ являются наземные станции командно-измерительной системы (НС КИС). По радиоканалам НС КИС осуществляется информационный обмен между НАКУ и КА. Эта станция предназначена для непрерывного выполнения задач управления, измерения орбитальных параметров, контроля и поддержания технических и баллистических характеристик КА, находящихся в ориентированных и неориентированных ре-

жимах функционирования [2].

Положительные перспективы развития НАКУ связаны с введением в эксплуатацию новых средств, удовлетворяющих повышенным требованиям надежности, функциональности и мобильности. Развитие систем НС КИС связывается с использованием технологий искусственного интеллекта для управления, контроля, диагностирования состояний оборудования и прогнозирования ситуаций.

Визуализация текущей информации о состоянии наблюдаемых объектов и окружающей среды, о ходе выполнения команд *центра управления полетом* (ЦУП) и о собственном состояния является одной из важнейших задач управления НС КИС. С этой целью специально для НС КИС разрабатываются интерфейсы, способные отражать числовую информацию, поступающую от различных источников, а также формировать ее в виде графиков и таблиц. Главная задача визуализации – предоставлять ЛПР информативное графическое отображение текущей ситуации, аккумулирующее контролируемые данные в единый образ.

Рассмотрим действующие способы отображения состояния НС КИС на примере интерфейса командно-измерительной станции головного предприятия космической связи. Интерфейс НС КИС предназначен для отображения текущей информации, включая телеметрическую информацию; информацию о состоянии окружающей среды (метеоинтформация): температура, влажность, направление ветра и др.; состояние аппаратуры НС КИС; командную информацию.

Следует отметить многофункциональность интерфейса, на котором располагается информация не только о внутреннем состоянии НС КИС, но и об окружающей среде, о контролируемом объекте, а также командная информация [3]. Все это делает простую графическую визуализацию динамических процессов недостаточно информативной и эффективной. Диспетчеру сложно следить за всеми изменениями, происходящими одновременно в разных частях экрана, что мешает целостному восприятию ситуации.

Для облегчения восприятия человеком больших массивов данных стали активно применяться полиэкраны, которые не только отражают текущие данные, но и формируют графический образ ситуации. Чтобы отобразить текущую ситуацию, необходимо многомерные данные представить в особом графическом виде, для чего подойдет только такой графический образ, в котором можно удачно сопоставить каждое из условий задачи отдельной части изображения. Синтезированный графический образ должен позволять ЛПР использовать свойства абстрактного изображения для визуального решения поставленной задачи. Именно такой графический образ, с помощью которого можно решить задачу, называют когнитивным, то есть способствующим пониманию ситуации. Когнитивная графика часто применяется в задачах с большим объемом анализируемой информации [4, 5]. Она позволяет представить данные в более понятном для ЛПР виде, что существенно сокращает время ее обработки.

Анализ деятельности оперативно-диспетчерского персонала сложного объекта показал, что достаточно использовать **трехуровневую систему представления информации** об управляемом объекте или процессе, включающую уровень системы (или объекта) в целом, на котором сообщается о том, в каком (нормальном, аномальном или критическом) состоянии она находится и в каких подсистемах возникли отклонения, уровень подсистемы, на котором выявляется состояние конкретной подсистемы, уровень непосредственно измеряемых параметров с указанием значений параметров и динамики их изменения [5].

Цветояркостный образ состояния НС КИС

Специально для НС КИС разработан интеллектуальный интерфейс, предназначенный для формирования когнитивного графического образа ситуаций, возникающих в процессе работы НС

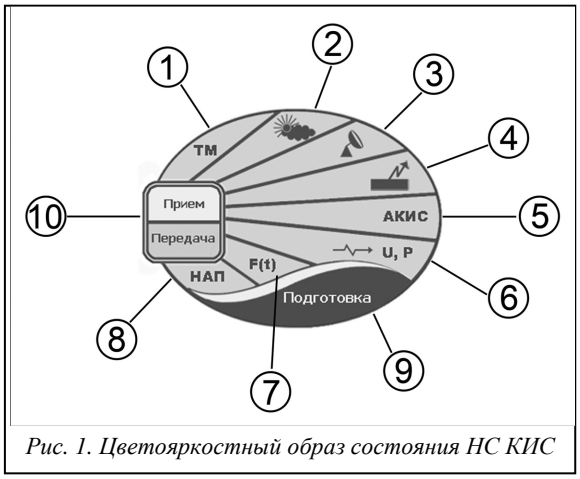


Рис. 1. Цветояркостный образ состояния НС КИС

КИС. Дополнительно он отслеживает и фиксирует события, требующие особого внимания или принятия решений.

На основе трехуровневой системы предоставления информации о контролируемом объекте разработано отображение, объединяющее два верхних уровня. Слияние образов состояния всей системы и ее подсистем способствует сокращению времени восприятия визуализируемых данных. Таким образом, получена следующая структура системы предоставления информации о состоянии НС КИС:

– уровень системы, на котором отображается состояние НС КИС в целом и выделяются подсистемы, в работе которых возникли отклонения;

– уровень подсистем и их параметров, на котором выявляется состояние конкретной подсис-

темы, указываются значения параметров и их отклонения от нормы.

Графический когнитивный образ при этом отображает следующие контролируемые пункты:

1) состояние телеметрии (наличие телеметрических кадров и частота их поступления);

2) метеоусловия: температура (градус С), давление (мм рт. ст.), влажность (%), направление ветра (градус), скорость ветра (м/сек.);

3) *антенная система* (АС) (номер работающего комплекта, точность наведения АС (градус), информация о необходимости поправки работающего комплекта АС (поправка ручная или поправка автоматическая), ситуация отказа работающего комплекта);

4) передатчик (номер работающего комплекта и его параметры: аттенюатор (дБ), температура (градус С), мощность передаваемого сигнала (Вт));

5) аппаратура НС КИС (номера работающих комплектов блоков аппаратуры, их работоспособность);

6) уровень (дБ) и мощность (Вт) передаваемого сигнала;

7) состояние установок из Цуп;

8) расхождение между фактическим временем и временем *навигационного приемника* (НАП) (мкс);

9) название режима функционирования НС КИС (работа, имитатор *промежуточной частоты* (ПЧ), имитатор *высокой частоты* (ВЧ), имитатор *низкой частоты* (НЧ), подготовка);

10) обмен служебной информацией и передача специальной информации.

На уровне системы выделяются цветом подсистемы, в которых возникли отклонения. На рисунке 1 представлено соответствующее обобщенное когнитивное представление уровня системы. Данный образ является модификацией круговой диаграммы, разбитой на секторы. Все секторы пронумерованы так, чтобы их названия совпадали с названиями пунктов списка, расположенного выше.

Появление аномального значения какого-либо из контролируемых параметров фиксируется изменением цвета сектора, отвечающего за соответствующую этому параметру подсистему НС КИС. В качестве примера приведем ситуацию, при которой наведение АС нуждается в автоматической поправке. Начальный угол приема НАП – 5°, следовательно, при отклонении АС на 6° теряется связь со спутником. В этом случае НС КИС не имеет возможности полноценно функционировать. На рисунке 2 приведено графическое отображение аномального состояния по точности наведения АС.

Рис. 2. Визуализация аномального состояния по точности наведения АС

На уровне подсистем происходит детализация ситуации. При щелчке клавишей мыши по одному из секторов появляются уточняющие образы, на которых можно просмотреть информацию о причинах некорректной работы НС КИС. Если какой-либо из параметров вышел из нормы, то оранжевый или красный цвет сектора дает знать о том, где произошли сбои в работе системы или возникли неблагоприятные условия. Рассмотрим детально отдельные компоненты контролируемой информации.

Метеоданные. На рисунке 3 представлен когнитивный графический образ метеоданных, приемлемых для нормальной работы НС КИС. В состав этого образа входит прямоугольник, разбитый на пять прямоугольных областей. В центральной зоне в один ряд расположены овалы, символизирующие значения метеопараметров (индикаторные фигуры). Числовые значения контролируемых метеоданных указаны на каждом индикаторном образе. Названия параметров и единицы их измерения находятся в нижней части образа. Для удобства будем называть образы такого типа прямоугольной диаграммой. Если параметры не влекут за собой сбои работы станции, то овалы лежат в пределах центральной зоны. Если значение параметра превысит норму, то соответствующий ему образ будет накладываться на область, расположенную выше центральной. Если значение параметра занижено, овал перекроет область, расположенную ниже центральной.

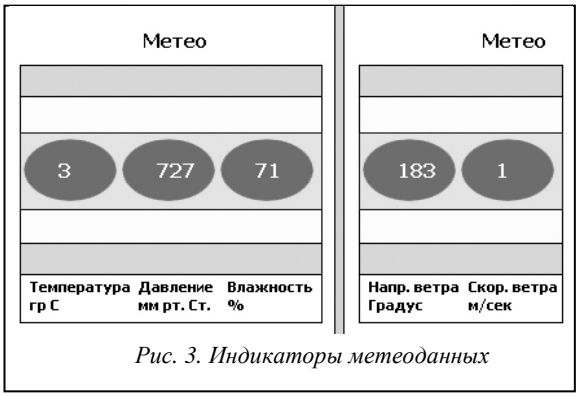


Рис. 3. Индикаторы метеоданных

На рисунке 4 показано, что при падении температуры до –45° соответствующий овал накладывается на зону, расположенную ниже всех остальных областей, предупреждая об опасности выхода из строя аппаратуры. Очень высокая скорость ветра, например 32 м/с, может привести к большим разрушениям. В таком случае соответст-

вующий образ достигает верхней области, граничащей с центральной, что свидетельствует о приближении к пределам нормы ветровых нагрузок

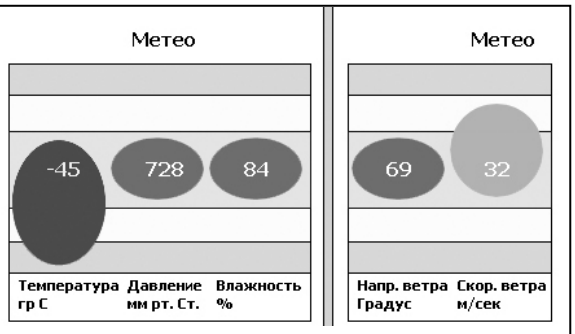


Рис. 4. Пример визуализации неблагоприятных метеоусловий

на НС КИС.

Прямоугольная диаграмма используется также для отображения параметров передатчика и установок из ЦУП.

Антенная система. При работе НС КИС важнейшим критерием является точность наведения АС. Для когнитивного отображения точности наведения предложен способ представления данных, который для удобства будем называть секторной диаграммой. Опишем отличительные особенности этого метода. Диаграмма представляет

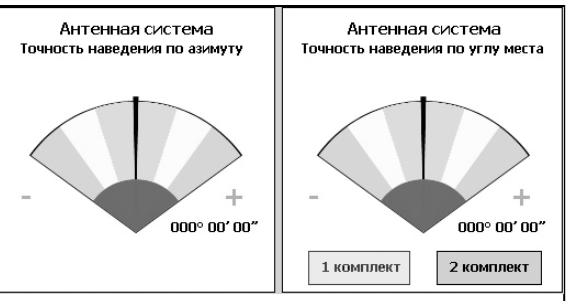


Рис. 5. Индикаторы состояния АС

собой совокупность секторов (рис. 5).

Можно выделить два сектора: сектор большего радиуса, разбитый на пять областей, и сектор меньшего радиуса. Центральную часть большего сектора занимает область, соответствующая значениям параметров, находящихся в пределах нормы. По обе стороны от нее расположены области, отвечающие за аномальные и приближающиеся к аномальным значения параметров. Стрелка указывает угол отклонения от цели. По тому, в какой области она находится, можно судить о допустимости величины этого отклонения. Сектор меньшего радиуса лежит поверх описанного сектора и меняет свой цвет в зависимости от значения контролируемого параметра подобно расположению стрелки. Для удобства будем называть этот однотонный сектор индикаторным сектором.

Справа от индикаторного сектора указано значение угла отклонения. Обязательным является наличие информации о точности наведения АС по азимуту и точности наведения АС по углу места. На рисунке 5 показан когнитивный образ состояния АС. Указатель отклонения в виде черной стрелки расположен в центральной зоне, что говорит о нормальном положении действующего комплекта АС.

Потеря связи со спутником является критической ситуацией для функционирования НС КИС. В случае отказа АС необходима быстрая реакция оперативно-диспетчерского персонала. Поэтому предлагается фиксировать время последнего отказа АС и время последнего восстановления АС в верхней части образа. В случае необходимости ручной или автоматической поправки АС информация об этом появляется в правой нижней части образа.

Секторная диаграмма используется для отображения точности наведения АС, расхождения во времени, уровня сигнала, мощности сигнала.

Для визуализации работоспособности и состояния приборов применяется цветовое матричное представление. Работающий прибор или узел подсвечен зеленым цветом, а неработающий – серым. Если прибор отказал, то соответствующий прямоугольник станет красным.

Разработка журнала событий

Важной составной частью интеллектуального интерфейса является журнал событий. Программа накапливает списки событий, на основе которых будет проводиться прогнозирование. Событие регистрируется в журнале сразу после его обнаружения. В журнале три категории записей: ошибки, предупреждения и сведения.

В каждой записи расположены данные, относящиеся к событию:

- дата и время, когда произошло событие;
- тип события (ошибка, предупреждение или сведение);
- источник события;
- все зафиксированные данные о состоянии НС КИС, сопровождающие событие.

К последнему пункту относятся:

1) информация о наличии и частоты поступления кадров телеметрии (ТМ);

2) метеоданные (Метео);

3) информация о точности наведения действующего комплекта АС (АС);

4) данные передатчика (ПРД);

5) информация о состоянии аппаратуры (АКИС);

6) данные уровня и мощности сигнала (Сигнал);

7) данные установок из ЦУП **(F(t))**;

8) информация о расхождении во времени НАП.

Для просмотра занесенных в журнал событий

необходимо нажать левой клавишей мыши на кнопку главной формы «Просмотреть события». В списке формы журнала событий располагаются сведения о произошедших сбоях в работе станции, предупреждения и информация о времени восстановления нормальной работы, сопровождающаяся значениями контролируемых параметров. Пример содержания списка журнала событий представлен в таблице.

Дата и время	13.11.2006 0:05:54	13.11.2006 0:05:55	13.11.2006 0:06:24	13.11.2006 0:06:25
Тип	Предупреждение	Сведение	Ошибка	Сведение
Событие	Сигнал: уровень=-214	Восстановление	АС: ОТКАЗ; Аппаратура: = АС 2 компл.	Восстановление
ТМ	8 кГц	8 кГц	8 кГц	8 кГц
Метео	0 729 87 146 2	0 729 87 146 2	0 729 87 147 1	0 729 87 147 1
АС	000° 00' 00" 000° 00' 00"	000° 00' 00" 000° 00' 00"	000° 00' 00" 000° 00' 00"	000° 00' 00" 000° 00' 00"
ПРД	50 55 108	50 55 108	50 54 108	50 54 108
АКИС	В норме	В норме	/ АС	В норме
Сигнал	-214 108	-179 108	-138 108	-138 108
F(t)	4 500 0 0 6 5	4 500 0 0 6 5	4 500 0 0 6 5	4 500 0 0 6 5
НАП	0	0	0	0

Кнопка «Очистить» позволяет удалить из дерева формы все зарегистрированные ранее события. После нажатия на нее журнал будет пуст, если не поступили новые сообщения, или отобразит следующие произошедшие события. Щелчок по кнопке «Показать все» приведет к занесению в дерево формы всех событий, зафиксированных за период просмотра файла информации функционального контроля.

В графе, отведенной для описания события, содержится текст, описывающий это событие. Предположим, в локальное время **t** произошли значительные отклонения в **n** контролируемых параметрах. Обозначим: **НКП** – название контролируемого пункта, в работе которого произошел сбой или для которого возникло предупреждение; **НПП** – название подпункта/параметра НКП; **ед**. – единица измерения НПП; **М** – значение, соответствующее НПП.

Тогда текст описания будет иметь следующую структуру:

$НКП_1$: $НПП_1$(ед.$_1$)=M_1;
$НКП_2$: $НПП_2$(ед.$_2$)=M_2;
…; $НКП_n$: $НПП_n$(ед.$_n$)=M_n.

Например, если вышел из строя блок аппаратуры, произошел отказ ПРД или АС, то в этом поле будут приведены перечень названий неработоспособных приборов и номера их комплектов.

Реализация когнитивного дополнения

Когнитивное дополнение интерфейса реализовано в среде разработки *CodeGear Delphi 2007 for Win32*. Необходимую скорость выполнения данного приложения обеспечивает использование пакета *Graphics32* с открытыми исходными кодами. Пакет содержит набор функций, классов, компонентов и элементов управления, разработанных для высокоэффективного программирования графическими средствами. Дерево журнала событий создано при помощи компоненты *Virtual Treeview* с открытыми исходными кодами, являющейся одним из самых гибких и доступных средств создания контроля элементов управления в виде дерева и/или списка на сегодняшний день.

Итак, интеллектуальный интерфейс опирается на применение методов когнитивного формализованного представления состояния предметной области в виде цветояркостных моделей и служит для поддержания режима работы, контроля, диагностики и прогнозирования состояний технических систем НС КИС. На основе интерфейса и его когнитивного приложения оператор визуально оценивает состояние НС КИС и окружающей среды. Целесообразность разработанных форм индикации состоит в обеспечении надежного и эффективного взаимодействия ЛПР с ЭВМ, в повышении оперативности за счет сокращения аналитической работы, компактности и наглядности геометрических форм представления информации. Экспериментальное программное обеспечение, реализующее когнитивное дополнение, встроено и испытано в составе НС КИС нового поколения [3].

Литература

1. Энциклопедия космонавтики. URL: http://kosmonavtica.ru/tag/kompleks-upravleniya/ (дата обращения: 23.03.2009).

2. Развитие наземного комплекса управления космическим аппаратом «KazSat» / Ю.М. Урличич [и др.]: матер. Междунар. конф. «…». Алматы. 2007. С. 19–20.

3. Смирнов С.В. Средства когнитивной графики для отображения текущего состояния наземных станций командно-измерительных систем // Авиакосмическое приборостроение. 2008. № 3. С. 47–57.

4. Вагин В.Н., Еремеев А.П. Базовые принципы конструирования интеллектуальных систем поддержки принятия решений реального времени для мониторинга и управления сложными техническими объектами: тр. Третьего расшир. сем. М.: Физматлит, 2003. С. 79–97.

5. Хачумов В.М., Ксенофонтова Е.В. Образный анализ и диагностика сложных процессов: докл. 11-й Всерос. конф. М., 2003. С. 201–204.

ГЕНЕТИЧЕСКАЯ НАСТРОЙКА ЦИФРОВОГО РЕГУЛЯТОРА ЧАСТОТЫ ВРАЩЕНИЯ С ДВУХРЕЖИМНОЙ ФИЛЬТРАЦИЕЙ

Ф.В. Калугин (ИПС им. А.К. Айламазяна РАН, г. Переславль-Залесский, teodor@dt.pereslavl.ru)

DIESEL ENGINE CONTROL SYSTEM WITH DUAL-MODE FILTERING ALGORITHM AND TUNING BY GENETICS ALGORITHM APPLYING

Kalugin Fedor V.

(Organization of Russian Academy of Sciences Ailamazyan, Pereslavl-Zalessky, teodor@dt.pereslavl.ru)

Abstract. The questions of the genetics algorithm applying for searching of the optimal diesel engine crankshaft PID-controller settings are considered. The validity of the dual-mode filtering algorithm applying is confirmed. The experimental results of the transient response of the controlling system are shown.

Key words: *dual-mode filtering, PID controller tuning, genetics algorithm, crankshaft.*

Рассмотрены вопросы применения генетического алгоритма для поиска оптимальных настроек ПИД-регулятора частоты вращения коленчатого вала дизельного двигателя. Показана обоснованность применения двухрежимного фильтра, учитывающего динамику частоты вращения коленчатого вала дизельного двигателя. Приведены результаты экспериментального исследования переходных характеристик системы управления.

Ключевые слова: *двухрежимный фильтр, настройка ПИД-регулятора, генетический алгоритм, коленчатый вал.*

Необходимость обеспечения устойчивости холостого хода автомобильных дизелей, оснащенных электронными системами управления, обусловливает создание специальных *систем управления* (СУ). Данная проблема решалась и при доводке электронной системы ЭСУ-1 на дизелях серий КАМАЗ-740.60-360, КАМАЗ-740.61-320, КАМАЗ-740.62-280, поскольку двухкаскадный *пропорционально-интегрально дифференциальный* (ПИД) регулятор автоматически не обеспечивал устойчивую работу дизеля при отключении нагрузки [1]. Алгоритмы управления получаются более эффективными, если вместо специальной СУ холостого хода применять универсальные, то есть обеспечивающие устойчивую работу дизеля на всех нагрузочно-скоростных режимах, включая и режимы холостого хода, и торможения двигателем. В настоящей работе данная проблема решается введением модифицированного цифрового фильтра оборотов частоты [2] с переменной структурой и универсальной генетической настройкой ПИД-регулятора.

Система управления дизельным двигателем

Принципиальная схема регулирования частоты вращения коленчатого вала показана на рисунке 1. Текущая частота вращения $\boldsymbol{\Omega}$ определяется

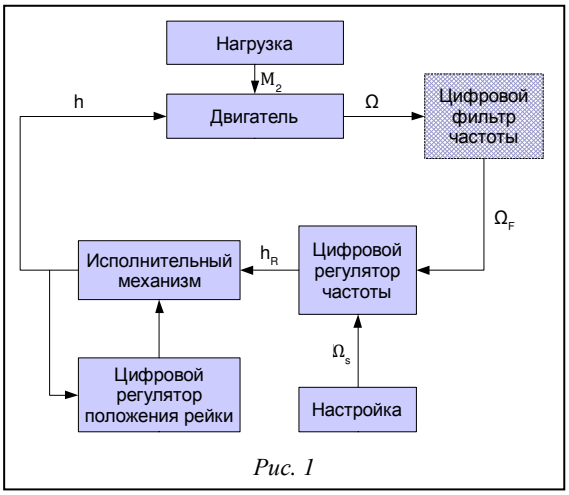


Рис. 1

величиной подачи топлива (положение рейки топливного насоса **h**), свойствами двигателя и нагрузкой (момент инерции нагрузки, момент сил внешнего сопротивления $\mathbf{M_2}$) и факторами окружающей среды.

Измеряемое значение $\boldsymbol{\Omega}$ подается на вход фильтра высокой частоты, затем отфильтрованная частота вращения $\boldsymbol{\Omega_F}$ передается в качестве входного сигнала в цифровой регулятор частоты вращения двигателя. Фильтр высокой частоты может отсутствовать в контуре управления двигателем. Цифровой регулятор задает требуемое положение рейки $\mathbf{h_R}$ для обеспечения стабильности скоростного режима, который определяется настройкой задаваемой частоты вращения $\boldsymbol{\Omega_S}$. *Исполнительный механизм* (ИМ) устанавливает физическое положение рейки **h**, близкое к задаваемому значению $\mathbf{h_R}$. Текущая величина **h** отличается от $\mathbf{h_R}$ вследствие инерции ИМ. Для стабилизации параметра **h** в ИМ используется обратная связь с цифровым регулятором положения рейки. Модификация цифрового двухрежимного фильтра оборотов частоты заключается в том, что не допускается большое отклонение отфильтрованного сигнала от реального: при разнице отфильтрованного сигнала от реального больше некоторого порога, характерного для двигателя, в качестве результата берется реальная частота.

Математическая модель системы регулирования

Уравнение частоты вращения двигателя имеет вид

$$\mathbf{J}\frac{\mathbf{d\Omega}}{\mathbf{dt}}=\mathbf{M_\Sigma}, \qquad (1)$$

где **J** – суммарный момент инерции двигателя и нагрузки, приведенной к коленчатому валу; **t** – время; $\mathbf{M_\Sigma}$ – суммарный момент приложенных сил. $\mathbf{M_\Sigma}$ можно представить в виде

$$\mathbf{M_\Sigma}=\mathbf{M_1}+\mathbf{M_2}+\mathbf{M_3}, \qquad (2)$$

где $\mathbf{M_1}$ – крутящий момент; $\mathbf{M_2}$ – момент сил внешнего сопротивления; $\mathbf{M_3}$ – момент сил внутреннего трения. Мгновенное значение крутящего момента можно представить в виде

$$M_1(t)=H_u\frac{i_d}{\pi\tau_d}g_c(t)\eta_e+M'(t), \quad (3)$$

где H_u – теплотворная способность топлива; i_d – число цилиндров в двигателе; τ_d – тактность двигателя; g_c – цикловая подача топлива; η_e – эффективный КПД; $M'(t)$ – флуктуационная составляющая («шум» двигателя), среднее по времени значение $M'(t)$ равно нулю [3]. Цикловая подача топлива зависит от частоты вращения Ω и координаты рейки топливного насоса h. В первом приближении будем считать g_c пропорциональной координате рейки и не зависящей от частоты:

$$g_c=k_g h, \quad (4)$$

где k_g – коэффициент пропорциональности. С учетом (4) формула (3) принимает следующий вид:

$$M_1(t)=K_1 h(t-\tau_1)+M'(t), \quad (5)$$

$$K_1=H_u\frac{i_d k_g}{\pi\tau_d}\eta_e,$$

где K_1 – коэффициент пропорциональности крутящего момента к координате рейки; τ_1 – запаздывание (время, необходимое на движение и сгорание топлива). Предположим, что момент сил внутреннего трения пропорционален частоте вращения $M_3=-\theta\Omega$, где θ – коэффициент сопротивления.

С учетом (2)–(5) уравнение момента импульса (1) принимает вид

$$J\frac{d\Omega(t)}{dt}+\theta\Omega(t)=K_1 h(t-\tau_1)+M'(t)+M_2(t). \quad (6)$$

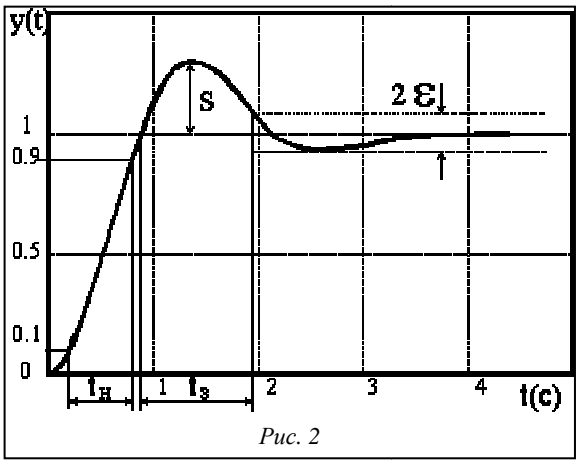


Рис. 2

Уравнение регулятора частоты вращения

Для регулирования частоты оборотов Ω используем классический ПИД-регулятор

$$h_{k+1}=K_P\Psi+K_I\sum_0^t\Psi+K_D\dot{\Psi}, \quad (7)$$

где нижний индекс k соответствует характеристике в момент времени $t=k\Delta t$; Δt – шаг квантования по времени; K_P, K_I, K_D – постоянные времени регулятора (пропорциональная, интегральная и дифференциальная); $\Psi=\Omega_S-\Omega$, где Ω_S – задаваемая угловая скорость (частота).

Прямое численное моделирование системы регулирования частоты вращения

На основе уравнений (6) и (7) строим систему уравнений, которая описывает замкнутую систему, состоящую из двигателя и регулятора (рис. 1). Эта система при единичном скачке на входе определяет кривую переходного процесса в системе. Численное моделирование проводим следующим образом:

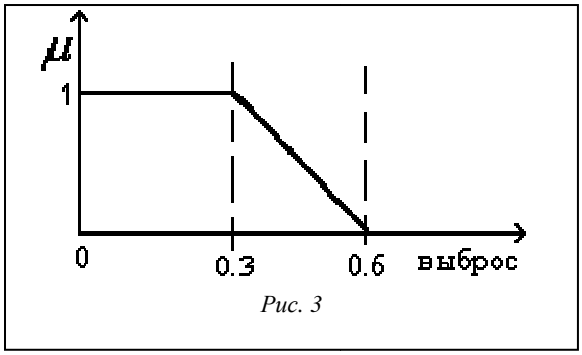


Рис. 3

1) устанавливаем первые $\Omega_1\ldots\Omega_{k-1}$ и $\Psi_1\ldots\Psi_{k_\tau}$, равные 0, где $k_\tau=\mathrm{round}\left(\frac{\tau}{T_0}\right)$;

2) вычисляем частоту на k-м шаге $\Omega_k=\Delta t\Psi_{k-1}+\Omega_{k-1}$;

3) вычисляем положение рейки на шаге $k+1$, $h_{k+1}=K_P\Psi+K_I\sum_0^t\Psi+K_D\dot{\Psi}$;

4) вычисляем приращение частоты на k-м шаге $\Psi_k=\frac{K_1 h_{k-k_\tau}+M_k+M_{2k}-\theta\Omega_k}{J}\Delta t$, где в качестве M_k берем случайные величины в интервале [–2, 2];

5) так как задачей нашего исследования является имитация условий холостого хода, то $M_k=0$ (внешняя нагрузка отключена);

6) увеличиваем шаг k, переходим к пункту 2.

Вследствие наложения на работу двигателя его физических характеристик (например, задано количество цилиндров) получается более сложный периодический сигнал с повторяющимся периодом в 264 зубца. Эти данные также учитываются при моделировании – они суммируются с вычисляемой частотой Ω_S в пункте 2.

Оценка качества переходного процесса

Для оценки используем 4 параметра [4] (вид переходной характеристики и ее параметров изображен на рисунке 2):

1) t_H – время нарастания, то есть время, за которое переменная $y(t)$ возрастает с 0,1 до 0,9 уста-

новившегося значения;

2) **S** – «выброс» (максимальное превышение сигналом **y(t)** единичного уровня);

3) $\mathbf{t_З}$ – время затухания переходного процесса (время между моментом первого достижения сигналом **y(t)** единичного уровня и моментом, начиная с которого значения **y(t)** остаются внутри интервала $[1\pm\boldsymbol{\varepsilon}]$; $\boldsymbol{\varepsilon}$ – некоторая постоянная);

4) $\mathbf{D_f}$ – среднеквадратичное отклонение от задаваемого сигнала, рассчитывается на отрезке затухания переходного процесса.

Первые 3 параметра являются стандартными для данной задачи. Автор статьи вводит четвертый параметр в силу использования данных о циклической неравномерности сигнала, обусловленной физическими характеристиками двигателя и моделирования флуктуационной составляющей.

Оценка переходной кривой по упомянутым выше параметрам $(\mathbf{t_H}, \mathbf{S}, \mathbf{t_З}, \mathbf{D_f})$ представляет собой многокритериальную задачу. Для ее решения воспользуемся предложенным Р. Беллманом и Л. Заде методом слияния целей и ограничений [5]. Введем лингвистические переменные «время нарастания», «время затухания», «выброс», «равномерность затухания» и определим на каждой из них терм «приемлемое значение» как нечеткое множество с функцией принадлежности $\boldsymbol{\mu}(\mathbf{x})$ трапецеидального вида. Пример функции принадлежности для терма «приемлемое значение» лингвистической переменной «выброс» приведен на рисунке 3.

Конкретные значения параметров переходной кривой $(\mathbf{t_H}, \mathbf{S}, \mathbf{t_З}, \mathbf{D_f})$ могут быть охарактеризованы степенью принадлежности $\boldsymbol{\mu}(\mathbf{x})$ каждого из параметров к терму «приемлемое значение» соответствующей лингвистической переменной. В рассматриваемом случае представляется естественным принять в качестве оценки переходного процесса

$$\mathbf{F(t_H, S, t_З, D_f) = \mu(t_H) + \mu(S) + \mu(t_З) + \mu(D_f)}\ . \quad (8)$$

Заданы следующие степени принадлежности:

$$\boldsymbol{\mu}(\mathbf{t_H}) = \begin{cases} 0, & \mathbf{t_H \le 200} \\ \dfrac{1}{400}\mathbf{t_H - 1.5}, & \mathbf{t_H \in (200, 600)} \\ 1, & \mathbf{t_H \ge 600} \end{cases},$$

$$\boldsymbol{\mu}(\mathbf{S}) = \begin{cases} 0, & \mathbf{S \le 0} \\ \mathbf{-0{,}01S - 1}, & \mathbf{S \in (0, 100)} \\ 1, & \mathbf{S \ge 100} \end{cases},$$

$$\boldsymbol{\mu}(\mathbf{t_З}) = \begin{cases} 0, & \mathbf{t_З \le 0} \\ \dfrac{1}{1350}\mathbf{t_З - 1}, & \mathbf{t_З \in (0, 1350)} \\ 1, & \mathbf{t_З \ge 1350} \end{cases},$$

$$\boldsymbol{\mu}(\mathbf{D_f}) = \begin{cases} 0, & \mathbf{D_f \le 0} \\ \dfrac{1}{45}\mathbf{D_f - 1}, & \mathbf{D_f \in (0, 45)} \\ 1, & \mathbf{D_f \ge 45} \end{cases}.$$

Исследуя работу реального фильтра частоты вращения, оценим переходный процесс при установлении величины частоты оборотов $\boldsymbol{\Omega}$, равной 600 об/сек.

Сигнал, снимаемый с двигателя, отфильтровываем от шумов и периодической составляющей, для этого в систему вне контура управления добавляем цифровой фильтр частоты, как это показано на принципиальной схеме (рис. 4).

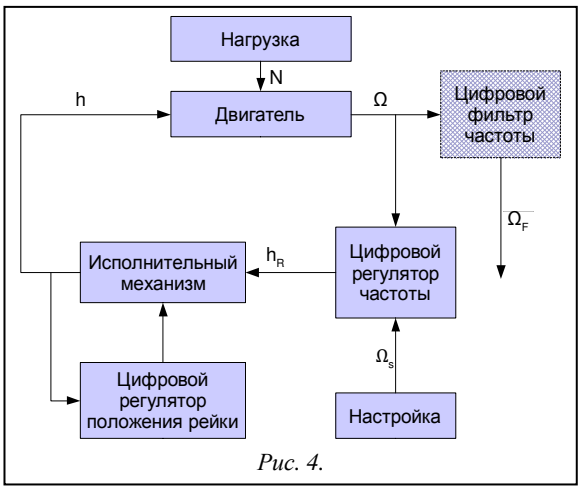


Рис. 4.

Необходимо автоматизировать поиск оптимальных настроек ПИД-регулятора.

Схема поиска оптимальной кривой переходного процесса

Для поиска оптимальных настроек, на которых оценочная функция (8) достигает максимума, необходимо уметь оценивать переходный процесс для определенных численных настроек $(\mathbf{K_P^0}, \mathbf{K_I^0}, \mathbf{K_D^0})$ ПИД-регулятора. Схема поиска оптимальной кривой переходного процесса выглядит следующим образом [4]:

1) эксперт задает функции принадлежности терма «приемлемое значение» для параметров $\mathbf{t_H}$, $\mathbf{S}$, $\mathbf{t_З}$, $\mathbf{D_f}$;

2) устанавливаются начальные величины коэффициентов регулятора $(\mathbf{K_P^0}, \mathbf{K_I^0}, \mathbf{K_D^0})$;

3) вводятся случайные значения поправок к начальным величинам коэффициентов ($\Delta\mathbf{K_P}$, $\Delta\mathbf{K_I}$, $\Delta\mathbf{K_D}$);

4) вычисляем новые значения коэффициентов $\mathbf{K_P = K_P^0 + \Delta K_P}$, $\mathbf{K_I = K_I^0 + \Delta K_I}$, $\mathbf{K_D = K_D^0 + \Delta K_D}$;

5) производится численное моделирование системы регулирования частоты вращения с коэффициентами, полученными в п. 4, при этом получается переходная характеристика, отображенная на рисунке 2;

6) вычисляются значения параметров $\mathbf{t_H}$, $\mathbf{S}$, $\mathbf{t_З}$, $\mathbf{D_f}$;

7) вычисляется степень принадлежности $\boldsymbol{\mu}(\mathbf{x})$ каждого из параметров к нечеткому множеству «приемлемое значение» соответствующей лин-

гвистической переменной;

8) вычисляется общая оценка переходной кривой по выражению (8);

9) если вычисленная оценка в п. 8 является максимальной, то алгоритм останавливается, иначе повторяется процедура для новых значений $(\mathbf{K_P^0}, \mathbf{K_I^0}, \mathbf{K_D^0})$.

Генетический подход

В качестве эффективного метода нахождения оптимальных значений коэффициентов регулятора используем генетический подход. *Генетические алгоритмы* (ГА) – это поисковая техника, имитирующая законы природной селекции и генетики [6]. ГА не касается проблема локального максимума.

Упрощенно схема ГА выглядит следующим образом:

1) на протяжении **k**-й итерации сохраняем популяцию потенциальных решений (хромосом) $\mathbf{P(k)=\{x_1^k,...,x_n^k\}}$, в начальный момент времени можно наполнить 1-ю популяцию любыми пригодными значениями, отдельной хромосомой $\mathbf{x_i^k}$ является набор коэффициентов ПИД-регулятора $(\mathbf{K_P^0}, \mathbf{K_I^0}, \mathbf{K_D^0})$;

2) каждое решение $\mathbf{x_i^k}$ оцениваем мерой пригодности $\mathbf{F(x_i^k)}$ по выражению (8);

3) формируем **k+1** поколение методом селекции решений в соответствии с мерой пригодности – каждое решение $\mathbf{x_i^k}$ отбирается в следующее поколение с учетом вероятности $\mathbf{p_i = \frac{F_i}{\sum_j F_j}}$;

4) некоторые решения из следующего поколения подвергаем действию генетических операторов (кроссовера и мутации) для образования новых решений [7];

5) алгоритм завершает работу при достижении заданных значений пригодности либо по исчерпании числа поколений.

При исследованиях используем модификацию ГА, которая сохраняет наилучшее решение [8].

Результат применения генетического подхода

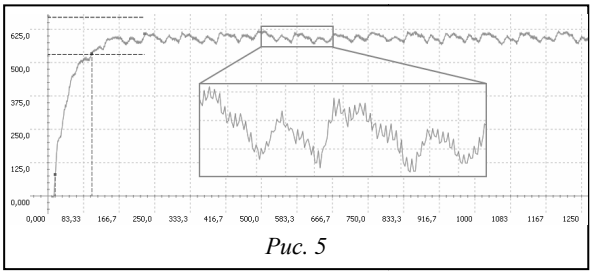


Рис. 5

В результате эксперимента при генетической настройке регулятора без применения фильтра частоты внутри контура управления и без учета флуктуационной составляющей были получены следующие параметры пропорционально-дифференциального регулятора: $\mathbf{K_P=7,74}$, $\mathbf{K_I=0}$, $\mathbf{K_D=0,14}$. Неравномерность сигнала, зависящая от цикловой подачи топлива (так называемые горбы с периодом 88 тактов), «срезана». Это отрицательно влияет на работу двигателя – происходит более быстрый его износ.

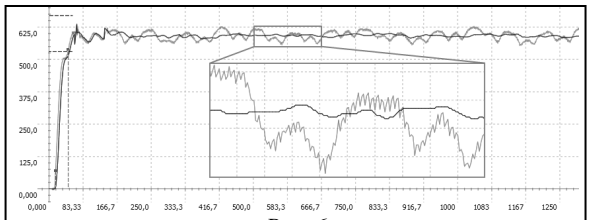


Рис. 6

В результате эксперимента при генетической настройке регулятора с применением фильтра частоты внутри контура управления и без учета флуктуационной составляющей были получены следующие параметры пропорционально-дифференциального регулятора: $\mathbf{K_P=5,4}$, $\mathbf{K_I=0}$, $\mathbf{K_D=0}$. Данные о неравномерности сигнала, зависящие от цикловой подачи топлива, «не срезаны». Это положительно влияет на работу двигателя – более оптимально регулируется рабочая частота оборотов коленчатого вала.

На рисунке 5 показан результат переходного процесса при генетической настройке регулятора без применения фильтра частоты внутри контура управления и с флуктуационной составляющей.

Получили ПИД-регулятор с параметрами $\mathbf{K_P=10,01}$, $\mathbf{K_I=2,81}$, $\mathbf{K_D=0,12}$. Недостатки те же, что и без флуктуационной составляющей.

На рисунке 6 показан результат переходного процесса при генетической настройке регулятора с применением фильтра частоты внутри контура управления и с флуктуационной составляющей.

Получен ПИД-регулятор с параметрами $\mathbf{K_P=14,12}$, $\mathbf{K_I=1,33}$, $\mathbf{K_D=0,07}$.

В заключение следует отметить, что в результате проведенных исследований, связанных с применением ГА, были найдены оптимальные настройки ПИД-регулятора для регулирования частоты оборотов коленчатого вала. Модифицированный двухрежимный алгоритм фильтрации частоты оборотов коленчатого вала дизельного двигателя [1] показал свою пригодность для включения в контур управления оборотами дизельного двигателя.

Литература

1. Хрящев Ю.Е., Кирик В.В., Третьяков А.А Использование аппарата fuzzy-логики в управлении дизелем: сб. тр. XX междунар. науч. конф. Ярославль: Изд-во ЯГТУ, 2007. С. 310–312. (Математические методы в технике и технологиях).

2. Калугин Ф.В. Двухрежимный алгоритм фильтрации частоты оборотов коленчатого вала дизельного двигателя // Мехатроника, Автоматизация, Управление. 2008. № 5. С. 35–41.

3. Крутов В.И. Автоматическое регулирование двигателей внутреннего сгорания. М.: Машиностроение, 1979. С. 616.

4. Денисенко В. ПИД-регуляторы: принципы построения и модификации // Современные технологии автоматизации. 2006. № 4. С. 66–74.

5. Giuseppe Munda. Multicriteria evaluation in a fuzzy environment. Physica-Verlag, 1995.

6. Chuck Karr. Genetic Algorithms for Fuzzy Controllers, AI Expert, 1991, Febriary, pp. 25–32.

7. Kim Chwee Ng, Yun Li. Design of Sophisticated Fuzzy Logic Controller Using Genetic Algorithms Proc. 3rd IEEE Int. Conf. On Fuzzy Systems, Orlando, FL, June, 1994. Vol. 3, pp. 1708–1712.

8. Goldberg D.E. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, 1989.

ПАРАЛЛЕЛЬНАЯ РЕАЛИЗАЦИЯ МОДЕЛИ MAP-REDUCE С ИСПОЛЬЗОВАНИЕМ ШАБЛОННЫХ КЛАССОВ C++

(Работа выполнена при поддержке Программы № 1 фундаментальных исследований Президиума РАН «Проблемы создания национальной научной распределенной информационно-вычислительной среды на основе развития GRID-технологий и современных телекоммуникационных сетей» и проекта РФФИ № 07-07-12038-офи)

А.А. Московский, к.х.н.; А.Ю. Первин; Е.О. Тютляева
(ИПС им. А.К. Айламазяна РАН, г. Переславль-Залесский, xgl@pereslavl.ru)

PARALLEL IMPLEMENTATION OF MAP-REDUCE MODEL USING C++ TEMPLATE CLASSES

Moskovsky Alexandr A., Ph.D; Pervin Artem Yu.; Tyutlyaeva Ekaterina O.
(Organization of Russian Academy of Sciences Ailamazyan Program Systems Institute of RAS, Pereslavl-Zalessky, xgl@pereslavl.ru)

Abstract. Approach to Map-Reduce model implementation using parallel programming skeletons (PPS) is considered. PPS based on C++ template classes provide a top-level of parallel program, while T-Sim parallel library is used as more low-level instrument. Programmer can affect to application performance by combination of policy classes, which execute PPS tuning in correlation to data processing algorithm or data amount. PPS ensure high scalability of application.

Keywords: *parallel programming skeletons, Map-Reduce, scalability.*

В работе рассматривается подход к реализации модели параллельного программирования Map-Reduce с помощью шаблонов параллельного программирования, реализованных как шаблонные классы C++. ШПП обеспечивают высокоуровневую конструкцию параллельных программ, используя в качестве низкоуровневого средства параллельного программирования библиотеку T-Sim. Программисту предоставляется возможность настройки поведения шаблонов в зависимости от специфики алгоритмов обработки данных и объема входных данных с помощью классов стратегий, что влияет на производительность параллельного приложения. Использование шаблона позволяет получать хорошо масштабируемые параллельные приложения для самых различных прикладных областей.

Ключевые слова: *шаблоны параллельного программирования, масштабируемые параллельные приложения, Map-Reduce.*

Задача создания эффективных параллельных программ не только не потеряла своей актуальности с ростом числа вычислительных узлов в высокопроизводительных установках, а, наоборот, приобрела еще большую остроту. Простое увеличение числа вычислителей не гарантирует рост производительности системы. Согласно закону Амдала, максимальный прирост производительности с увеличением количества вычислителей в системе (линейный) достижим только на алгоритме, который вообще не содержит последовательных вычислений (идеальное распараллеливание) [1]. Чем больше доля последовательных вычислений в алгоритме, тем меньший выигрыш можно получить от добавления вычислительных узлов. Например, если 5 % кода программы может выполняться только последовательно, то никакое увеличение числа процессоров не позволит добиться прироста производительности более чем в 20 раз. Поэтому большая часть усилий специалистов по параллельному программированию затрачивается на уменьшение доли последовательных вычислений в алгоритме.

Для прикладного программиста, основная сфера интересов которого ограничивается предметной областью, усилия по написанию эффективного параллельного кода являются досадными накладными расходами. К тому же эти усилия не всегда приводят к желаемому результату, поскольку создать действительно эффективную параллельную программу не так просто, часто это требует знания различных аспектов распределенных вычислений и определенных навыков программирования для многопроцессорных установок. Используя стандартные средства, программист вынужден задумываться о синхронизации доступа, балансировке нагрузки на вычислительных узлах, учитывать расходы на передачу данных по сети и т.п. Потребность в сокращении времени на проектирование и реализацию параллель-

ных программ обусловила появление на рынке программных продуктов различных систем, частично автоматизирующих процесс создания таких программ. Однако освоение этих систем, в свою очередь, может занимать существенное время.

Целью нашего исследования было создание программного инструмента, который сделает для программиста прозрачными все аспекты распараллеливания вычислений, не потребует от него дополнительных усилий по освоению новой среды программирования и обеспечит эффективный параллельный код приложения. В качестве такого инструмента авторы предлагают реализацию шаблонов на языке C++ для параллельных вычислений. Шаблоны обеспечивают высокоуровневую конструкцию параллельных программ [2]. В качестве более низкоуровневого средства параллельного программирования используется библиотека шаблонных классов C++ *T-Sim* [3], реализующая подход автоматического динамического распараллеливания программ. Пользовательский интерфейс предлагаемых шаблонов совместим с *Standard Templates Library* (STL) языка C++. Для компиляции используется стандартный компилятор с языка C++. Решение предназначено для кластеров и распределенных сетей (Грид).

Описание модели Map-Reduce

Популярная модель параллельного программирования *Map-Reduce* хорошо подходит для обработки огромных массивов данных на большом количестве компьютеров. Модель является крайне простой и при этом очень эффективной [4]. Она позволяет программисту отвлечься от проблем параллельного выполнения и сконцентрироваться на алгоритмах. Доля последовательного кода в алгоритме стремится к нулю. Алгоритм разбивается на два слоя параллельных операций и в итоге очень хорошо масштабируется. Стандартный алгоритм (имеющий последовательные участки) будет иметь предел, после которого добавление новых процессоров не дает выигрыша в производительности, с вариантом *Map-Reduce* этот предел почти недостижим. Данная особенность делает модель очень актуальной в условиях постоянного увеличения числа процессоров в современных кластерных установках. Большое количество реальных задач, естественно, выражается в терминах этой модели. Дополнительными плюсами являются независимость и возможность при сбое вычислить отдельный участок заново, не перезапуская весь процесс.

В этой модели функция *Map* (отображение) независимо обрабатывает входные данные, поделенные на блоки. Затем функция *Reduce* (свертка) формирует окончательный результат из предварительно обработанных данных. Рисунки 1 и 2 иллюстрируют работу функций *Map* и *Reduce*. Последовательным аналогом параллельного шаблона *Map* является шаблон *stl::transform* из библиотеки STL, который применяет функцию к диапазону элементов и сохраняет результаты. Последовательным аналогом параллельного шаблона *Reduce* является шаблон *stl::accumulate* из библиотеки STL, который осуществляет свертку списка или другой структуры в одну величину применяя бинарный оператор между всеми элементами списка. В нашем примере для наглядности взяты элементарные функции обработки – функция удвоения для шаблона *Map* и функция сложения для шаблона *Reduce*.

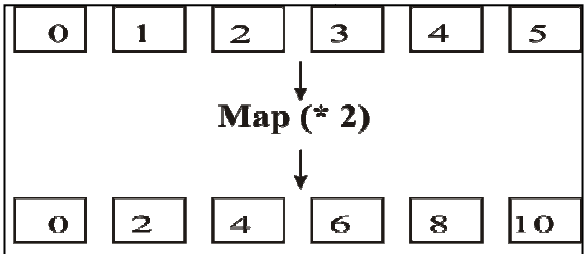


Рис. 1. Функция Map (отображение)

Реализация: классы стратегий

Шаблоны параллельного программирования (ШПП) *Map* и *Reduce* реализованы как шаблонные классы языка C++, по интерфейсу совместимые с библиотекой STL. При реализации использовалась технология классов стратегий (*policy classes*) [5]. Механизм стратегий основан на комбинации шаблонов C++ с множественным наследованием. Класс, использующий стратегии (главный класс), представляет собой шаблонный класс C++ с многими шаблонными параметрами, каждый из которых является стратегией, определяющей какой-либо один аспект поведения главного класса. Классы стратегий являются базовыми для главного класса (рис. 3). Когда на экземпляре главного класса происходит вызов функции *method*, вызывается та функция, которая определена в классе стратегии, переданном в главный класс в качестве параметра при инициации шаблона. В нашем примере это будут функции *Strategy1::method* для *Mainclass<Strategy1>* и *Strategy2::method* для *Mainclass<Strategy2>*. У пользователя есть возможность определить свой класс стратегии. Для приведенного примера этот класс обязательно должен содержать определение метода *method*.

Главный класс делегирует часть функцио-

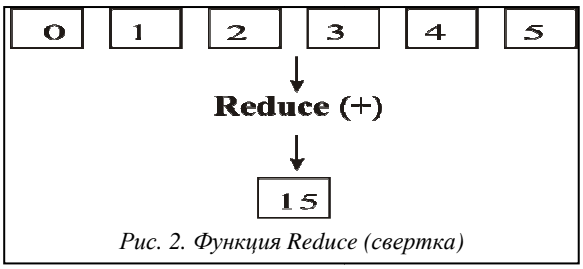


Рис. 2. Функция Reduce (свертка)

нальных возможностей стратегиям. Можно указывать стратегии по умолчанию. Комбинация различных стратегий дает пользователю набор решений, из которого можно выбрать наиболее удовлетворяющее его задаче. Таким образом, данная технология обеспечивает гибкий механизм настройки поведения главного класса.

В текущей реализации ШПП используются стратегии *LoopingTrait* и *AggregationTrait*, которые позволяют настраивать поведение шаблонов для повышения производительности приложения.

Стратегии типа LoopingTrait определяют способ фрагментации входных данных для их последующей обработки. По умолчанию пользователь может воспользоваться стратегией *DefaultLooping*, при которой шаблон обрабатывает все переданные ему данные в одном цикле (за один раз). Однако в случае, когда необходимо обработать очень большое количество данных, обработка их за один цикл может привести к потере производительности. Например, если операционной системе не удается загрузить в оперативную память все обрабатываемые данные, начинается использование механизма виртуальной памяти, что сразу приводит к резкой потере производительности шаблонов и приложения в целом. В этом случае лучше разбить исходное множество на несколько фрагментов и обработать их по очереди (за несколько циклов).

Специально для таких случаев разработана стратегия **EvenLooping<N>**, где **N** – количество разбиений. Входные данные будут разбиты на **N** независимых блоков, при этом размерность пользовательских данных необязательно должна быть кратной **N**. Независимо от размерности данных обработка будет произведена корректно. В таких случаях размер последнего обрабатываемого блока данных будет чуть больше остальных за счет остатка.

Кроме того, если ни одна из предлагаемых стратегий не удовлетворяет условиям задачи пользователя, он может написать свою стратегию разбиения. Например, если прикладному программисту понадобится разбить свои данные на неравные сложноструктурированные блоки.

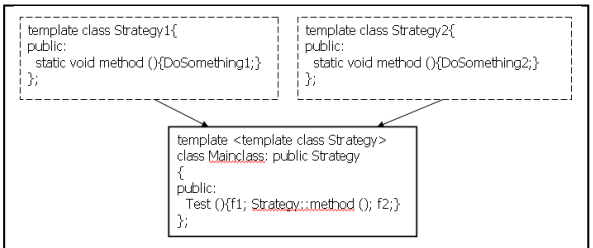


Рис. 3. Классы стратегий и главный класс с одной стратегией

Стратегии типа AggregationTrait определяют способ объединения элементов множества обрабатываемых данных в блоки.

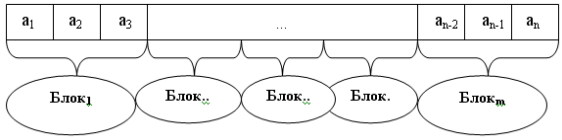


Рис. 4. Объединение элементов множества обрабатываемых данных в блоки

Использование различных стратегий типа *AggregationTrait* позволяет влиять на вес гранулы параллелизма. По умолчанию используются стратегия *DefaultAggregation*, при которой для каждого элемента из переданного пользователем множества будет порождаться отдельный параллельный поток исполнения. Такое поведение может быть эффективным, если определенная пользователем функция требовательна к ресурсам (процессорному времени). Однако при легковесной операции накладные расходы на распараллеливание исполнения (например, на передачу данных по сети) могут свести на нет выгоды от распараллеливания. Для этого случая следует воспользоваться стратегией **FixedAggregation<N>**, при которой параллельный поток исполнения будет порождаться не для одного элемента, а для блока из **N** элементов.

На рисунке 4 продемонстрировано применение класса стратегии **FixedAggregation<3>**. Использование стратегии разбиения *FixedAggregation* увеличивает тяжесть одной гранулы в **N** раз и может обеспечить прирост производительности параллельного приложения в случае легковесной операции обработки данных.

Библиотека T-Sim

В качестве низкоуровневого средства для организации параллельных вычислений ШПП используют библиотеку *T-Sim*, которая представляет собой набор шаблонов и макроопределений на языке C++. Библиотека реализует сетевой вызов функций, доставку посчитанного результата его потребителю, операции с неготовыми значениями, поддержку механизма неявной синхронизации процессов (приостановка и возобновление вычислений). *T-Sim* обеспечивает автоматическое динамическое распараллеливание приложения, автоматическую балансировку нагрузки на узлах многопроцессорной вычислительной установки. Механизм реализации планировщика заданий в *T-Sim* позволяет программисту не только использовать планировщик по умолчанию, но и легко подключать свой алгоритм планирования. Параллельная программа, базирующаяся на использовании библиотеки *T-Sim*, не содержит явных распараллеливающих конструкций, но все же она должна содержать описание данных и функций с использованием средств *T-Sim*. Применение ШПП позволяет программисту абстрагироваться от реализации параллельной программы и использования биб-

лиотеки *T-Sim*. Все необходимые для шаблонов *Map* и *Reduce* структуры данных и функции, содержащие алгоритм обработки этих данных, описываются на языке C++ обычным способом.

Таким образом, предлагаемое решение использует многоуровневый подход: на верхнем уровне программа оформляется с использованием ШПП, что гарантирует корректный параллельный код приложения; доступ к низкоуровневым средствам библиотеки *T-Sim* в случае необходимости позволяет программисту оптимизировать параллельное исполнение для специфических условий его приложения, добиваться приемлемых значений производительности и равномерной загрузки вычислительной установки.

Программный интерфейс шаблонов

Интерфейс шаблона *Map* имеет следующий вид:

```
template
<typename InIter, typename OutIter, typename Op,
 typename LoopingTrait, typename AggregationTrait>
OutIter Map(InIter first, InIter last, OutIter result, Op& in),
```

где **InIter** – тип входного итератора; **OutIter** – тип возвращаемого итератора; **Op** – класс, содержащий пользовательскую операцию обработки данных; **LoopingTrait** – стратегия разбиения на циклы; **AggregationTrait** – стратегия объединения обрабатываемых данных в блоки; **first** – итератор начала передаваемых данных; **last** – итератор конца передаваемых данных; **result** – итератор на начало результата; **in** – экземпляр класса, содержащего операцию обработки данных.

В случае использования стратегий по умолчанию можно воспользоваться упрощенным интерфейсом шаблона:

```
template
<typename InIter, typename OutIter, typename Op>
OutIter MapS(InIter first, InIter last, OutIter res, Op& op)
{
return Map <InIter, OutIter, Op, DefaultLooping, DefaultAggregation>
(first, end, res, op);
}
```

Интерфейс шаблона *Reduce* имеет аналогичный вид.

Для использования ШПП программист должен определить операцию обработки данных на языке C++.

Предлагаемые ШПП были использованы при реализации приложения по классификации мультиспектральных космических снимков с помощью метрики Махалонобиса и при создании экспериментального сервера приложений на основе библиотеки *T-Sim*.

Предлагаемые шаблоны параллельного программирования реализуют модель *Map-Reduce*, которая позволяет получать хорошо масштабируемые параллельные приложения.

Шаблоны *Map* и *Reduce* обеспечивают высокоуровневую конструкцию параллельных программ. В качестве более низкоуровневого средства параллельного программирования используется библиотека шаблонных классов C++ *T-Sim*. При программировании приложений с использованием шаблонов *Map* и *Reduce* типы данных, специфичные для *T-Sim*, скрыты от прикладного программиста за интерфейсом шаблонов. Пользовательский интерфейс предлагаемых шаблонов совместим с *Standard Templates Library* языка C++. Для компиляции шаблонов используется стандартный компилятор с языка C++.

Использование классов стратегий позволяет осуществлять настройку шаблонов в зависимости от специфики алгоритмов обработки данных и объема входных данных, влияя на производительность параллельного приложения.

Шаблоны *Map* и *Reduce* позволяют решать широкий класс различных прикладных задач. Например, решение дифференциальных уравнений – это итерационное применение функции *Map*; любой генетический алгоритм – это итерационное применение шаблона *Map* к популяции. Алгоритм *Reduce* может быть использован в задачах оптимизации.

Литература

1. Закон Амдала. URL: http://en.wikipedia.org/wiki/Amdahl%27s_law (дата обращения: 30.03.2009).

2. Московский А.А., Первин А.Ю., Сергеева Е.О. Первый опыт реализации шаблона параллельного программирования на основе Т-подхода: тр. Междунар. конф. М.: Наука. Физматлит. 2006.Т. 1. (Программные системы: теория и приложения).

3. Московский А.А.. T-Sim – библиотека для параллельных вычислений на основе подхода Т-системы: Там же.

4. Map Reduce. URL: /http://artamonov.ru/2008/09/16/map-reduce/#more-122 (дата обращения: 30.03.2009).

5. Alexandrescu. Modern C++ Design. Addison-Wesley, 2001.

АВТОМАТИЧЕСКОЕ ПЛАНИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ФОКУСИРОВКИ

(Работа выполнена при поддержке РФФИ, проект 07-01-00763)

И.В. Трофимов (ИПС им. А.К. Айламазяна РАН, г. Переславль-Залесский, itrofimov@km.ru)

AUTOMATIC PLANNING WITH FOCUSING

Trofimov Igor V. *(Organization of Russian Academy of Sciences Ailamazyan Program Systems Institute of RAS, Pereslavl-Zalessky, itrofimov@km.ru)*

Abstract. One of essential research directions in automatic planning is the development of methods for search space reduction. There have been numerous attempts to build such methods based on various abstraction techniques. The paper proposes the use of focusing as an abstraction technique and demonstrates experimental results.

Keywords:*automatic planning, abstraction, focusing, focalization, planner.*

Одним из важнейших направлений исследований в области автоматического планирования является разработка методов, позволяющих сократить пространство поиска решения. Неоднократно предпринимались попытки положить в основу таких методов какой-либо механизм абстрагирования. В работе рассматривается попытка использования фокусировки в качестве такого механизма, приводятся результаты экспериментов.

Ключевые слова: *автоматическое планирование, абстрагирование, фокусировка, планировщик.*

Традиционно в литературе по искусственному интеллекту абстрагирование определяется как частный случай изменения способа представления. По мнению современных исследователей проблемы абстрагирования [1], сложно дать понятию «абстрагирование» точную формулировку, которая была бы достаточно общей, чтобы включать все разнообразие методов, интуитивно подпадающих под эту категорию, и в то же время была бы достаточно специфичной, чтобы исключить другие формы изменения способа представления, интуитивно абстрагированием не являющиеся (например переформулировка). Мы будем придерживаться определения, предложенного Зюкером [2].

Методы абстрагирования можно классифицировать по механизму, который используется для скрытия несущественных деталей. По мнению автора, таких механизмов два.

1. Скрытие элементов предметной области. В качестве элементов могут выступать объекты, процессы, явления, действия и т.д. В литературе такой механизм называют фокусировкой, или скрытием домена, или проекцией.

2. Скрытие свойств элементов предметной области. Скрытие свойств может сделать отдельные элементы домена неразличимыми (уменьшается разнообразие), но количество элементов не изменяется. Скрытие свойств может выполняться не для всех элементов домена, а только для определенных групп или даже отдельных элементов. В литературе этот механизм называют также избирательностью, или скрытием содомена, или абстрагированием домена.

Абстрагирование в планировании

Существует множество планировщиков, в которых можно выделить общую методологическую основу: сначала осуществляется переход к абстрактной формулировке задачи, затем решается абстрактная задача, сложность которой меньше сложности исходной задачи, и, наконец, на базе абстрактного решения отыскивается решение исходной задачи.

Абстрагирование формулировки задачи в большинстве работ сводится к синтаксическому удалению элементов предусловия действий (типично – элементов конъюнкции в предусловиях, выраженных в форме конъюнкции литералов). К числу таких планировщиков относятся системы ABSTRIPS, ABTWEAK, PRODIGY/ALPINE. Абстрагирование также можно выполнять путем синтаксического удаления элементов эффекта действий. Этот принцип используют планировщики HSP и FF.

Что касается применения абстрактного решения, то здесь существуют два основных метода. Первый заключается в том, что длина абстрактного решения используется как эвристическая оценка расстояния до цели. На базе этого метода реализованы планировщики HSP и FF. Во втором методе само абстрактное решение выступает в качестве каркаса искомого решения, и необходимо найти лишь подходящую ему конкретизацию. На базе этого метода реализованы планировщики ABSTRIPS, ABTWEAK, PRODIGY/ALPINE. Для второго подхода важным условием является возможность создания эффективного алгоритма конкретизации абстрактного решения. При определенных ограничениях это условие выполняется. Однако если оно не выполняется, такой подход может оказаться гораздо менее эффективным, чем методы, не использующие абстрагирование.

Все упомянутые планировщики осуществляют абстрагирование путем скрытия свойств элементов предметной области, а именно, свойств действий. Однако не менее полезным для поиска решения проблемы вычислительной сложности является скрытие самих элементов предметной области (фокусировка) – объектов и действий, которые не могут иметь отношение к решению текущей задачи. Отсев нерелевантных объектов и действий может в значительной мере сокращать пространство поиска. Кроме того, важным свойством фокусировки является то, что *абстрактное решение – это также и конкретное решение*, поэтому нет необходимости в конкретизации. Автору неизвестны планировщики, осуществляющие абстрагирование таким способом.

Модель фокусировки для задачи планирования

Определимся с формализацией задачи планирования и механизма абстрагирования.

Обозначим **Schemas** конечное множество схем действий, описывающих все возможные виды деятельности для данного агента. Каждому элементу

множества **Schemas** соответствует целое неотрицательное число **k**, обозначающее количество объектов, вовлеченных в описание действий, соответствующих данной схеме. Назовем это число *арностью схемы действий*, или *количеством параметров схемы действий*. Арность схемы действий удобно моделировать при помощи функции

h: **Schemas** → $\mathbf{Z}^+$.

На письме для обозначения арности схемы действий используется верхний индекс. Например, запись $\mathbf{schema}^{\mathbf{k}}$ означает, что речь идет о такой **schema**, для которой **h**(**schema**) = **k**.

Каждому описанию схемы действий соответствует непустое множество описаний конкретных действий. Описание действия представляет собой кортеж вида: ‹$\mathbf{schema}^{\mathbf{k}}$, $\mathbf{obj}_1$,…, $\mathbf{obj}_k$›, где $\mathbf{obj}_i$ – имена объектов, вовлеченных в описание действия. Способы формирования описаний конкретных действий из схем могут быть различными. Множество всех действий, соответствующих некоторой схеме действий **schema**, обозначим **actions**[**schema**], а множество всех возможных действий агента – **Actions**.

$$\mathbf{Actions} = \bigcup_{\mathbf{schema} \in \mathbf{Schemas}} \mathbf{actions[schema]}.$$

Во всякий момент внешняя среда находится в каком-либо состоянии. Множество всех возможных состояний, в которых может пребывать внешняя среда, обозначим **States**. Для каждого состояния в **States** определено множество действий из **Actions**, которые выполнимы в данном состоянии. Другими словами, существует функция

g: **Actions**×**States**→{*выполнимо*, *невыполнимо*},

которая для каждого действия из **Actions** и каждого состояния среды из **States** определяет, выполнимо ли действие в данном состоянии. Функция **g** – *функция выполнимости*.

Выполнение всякого действия приводит к изменению состояния среды. Обозначим **AS** множество всех пар ‹**action**, **state**› (**action**∈**Actions**, **state**∈**States**), для которых выполняется условие (**x**∈**AS**→(**g**(**x**)=*выполнимо*)). Определим функцию

f: **AS** → **States**,

причем, если **f**(**action**, $\mathbf{state}_1$)=$\mathbf{state}_2$, то $\mathbf{state}_1 \neq \mathbf{state}_2$; **f** – *функция переходов*.

Определим *задачу планирования* как кортеж вида ‹**States**, **Actions**, **g**, **f**, $\mathbf{S}_0$, **SG**›, где $\mathbf{S}_0 \in \mathbf{States}$ – *начальное* состояние; **SG**⊂**States** – множество состояний, являющихся *целевыми* для данной задачи планирования (в этих состояниях выполняется целевое условие); $\mathbf{S}_0 \notin \mathbf{SG}$.

Решением задачи планирования (или *планом*) назовем последовательность вида ‹$\mathbf{S}_0$, $\mathbf{A}_0$, $\mathbf{S}_1$, $\mathbf{A}_1$, …, $\mathbf{S}_N$, $\mathbf{A}_N$, $\mathbf{S}_{N+1}$›, для которой выполняются следующие условия:

1. $\mathbf{S}_i \in \mathbf{States}$, $\mathbf{A}_j \in \mathbf{Actions}$, где $\mathbf{0} \le \mathbf{i} \le \mathbf{N+1}$, $\mathbf{0} \le \mathbf{j} \le \mathbf{N}$;
2. $\mathbf{S}_{N+1} \in \mathbf{SG}$;
3. $\mathbf{S}_i \notin \mathbf{SG}$, где $\mathbf{0} \le \mathbf{i} \le \mathbf{N}$;
4. $\mathbf{g}(\mathbf{A}_i, \mathbf{S}_i)$=*выполнимо*, где $\mathbf{0} \le \mathbf{i} \le \mathbf{N}$;
5. $\mathbf{f}(\mathbf{A}_i, \mathbf{S}_i)=\mathbf{S}_{i+1}$, где $\mathbf{0} \le \mathbf{i} \le \mathbf{N}$.

Пусть **P** – некоторая задача планирования, тогда **Solutions**(**P**) обозначим множество всех возможных решений задачи планирования **P**.

Определив таким образом постановку задачи планирования и результат ее решения, перейдем к формальному определению понятия фокусировки для задачи планирования.

Для данной задачи **P**=‹**States**, **Actions**, **g**, **f**, $\mathbf{S}_0$, **SG**› *результатом фокусировки* будет абстрактная задача $\mathbf{P}_1$=‹$\mathbf{States}_1$, $\mathbf{Actions}_1$, **g**, **f**, $\mathbf{S}_0$, $\mathbf{SG}_1$›, отвечающая следующим требованиям:

1. $\mathbf{States}_1 \subseteq \mathbf{States}$;
2. $\mathbf{Actions}_1 \subseteq \mathbf{Actions}$;
3. $\mathbf{a} \in \mathbf{Actions}_1$, $\mathbf{s} \in \mathbf{States}_1$, $\mathbf{g}(\mathbf{a}, \mathbf{s})$=*выполнимо* → $\mathbf{f}(\mathbf{a},\mathbf{s}) \in \mathbf{States}_1$;
4. $\mathbf{SG}_1 \subset \mathbf{States}_1$;
5. $\mathbf{SG}_1 \subseteq \mathbf{SG}$;
6. $\mathbf{Solutions}(\mathbf{P}_1) \subseteq \mathbf{Solutions}(\mathbf{P})$;
7. $|\mathbf{Solutions}(\mathbf{P})| \neq \mathbf{0} \rightarrow |\mathbf{Solutions}(\mathbf{P}_1)| \neq \mathbf{0}$.

Предложенная модель фокусировки предполагает, что абстрактная задача отличается от исходной содержанием множеств состояний, действий, целевых состояний. В данной работе рассматривается механизм фокусировки, сокращающий *только* множество **Actions**, то есть предполагается выполнение условий $\mathbf{States}_1$=**States**, $\mathbf{SG}_1$=**SG** и $\mathbf{Actions}_1 \subset \mathbf{Actions}$.

Согласно требованию 7, если существует решение исходной задачи, то множество $\mathbf{Actions}_1$ должно содержать набор действий, из которых можно получить хотя бы одно решение абстрактной задачи. Это абстрактное множество действий назовем *значимым контекстом* и обозначим **SCx**. Исключенные (абстрагированные) действия составляют множество, которое назовем *допустимым незначимым контекстом* и обозначим **NCx**. Эти два вида контекстов связаны соотношением **SCx**=**Actions****NCx**. Таким образом, чтобы осуществить абстрагирование, нам достаточно найти либо значимый контекст, либо допустимый незначимый контекст.

Чтобы воспользоваться предложенным механизмом абстрагирования, необходимо решить две подзадачи: найти адекватное *средство определения* для контекстов и *механизм выявления* **NCx** или **SCx** для текущей задачи планирования.

Способы определения контекстов

По определению, значимый и допустимый незначимый контексты являются множествами. Есть два основных способа определения множеств – перечисление элементов и определение посредством характеристического свойства. Рассмотрим второй способ.

В общем виде определение контекста **Cx** посредством характеристического свойства можно

записать следующим образом: **Cx**={**a** | **a**∈ **Actions**, **filter**}, где **filter** – требование к элементам **Cx** (характеристическое свойство). То есть контекст представляет собой такое множество элементов из **Actions**, которые отвечают требованию **filter**.

В силу того что действие моделируется как составная сущность, а именно как кортеж вида ‹**schema**$^{\mathbf{k}}$, **obj**$_1$,…, **obj**$_{\mathbf{k}}$›, можно накладывать ограничения на элементы, из которых состоит действие.

Во-первых, можно потребовать, чтобы элементами контекста были действия, порожденные от одной определенной схемы действий: **Cx**={**a** | **a**∈ **Actions**, **a**∈ **actions**[**schema**]}, где **schema**∈ **Schemas**.

Во-вторых, требование может постулировать, что **n**-м параметром действия является имя конкретного объекта **Obj**. Обозначим **actions**$_{\mathbf{i,Obj}}$[**schema**$^{\mathbf{k}}$] множество действий, порожденных от схемы **schema**$^{\mathbf{k}}$, у которых в качестве **i**-го параметра (и, следовательно, **i+1**-го элемента кортежа, описывающего действие) выступает имя объекта **Obj** (здесь **1≤i≤k**). Тогда множеством всех действий, у которых **i**-й параметр равен **Obj**, является объединение множеств **actions**$_{\mathbf{i,Obj}}$[**schema**$^{\mathbf{k}}$] по всем схемам действий с арностью, большей либо равной **i**. Обозначим это множество **actions**$_{\mathbf{i,Obj}}$:

$$\mathbf{actions}_{\mathbf{i,Obj}} = \bigcup_{\substack{\mathbf{schema}^{\mathbf{k}} \in \mathbf{Schemas} \\ \mathbf{i} \le \mathbf{k}}} \mathbf{actions}_{\mathbf{i,Obj}}[\mathbf{schema}^{\mathbf{k}}],\ \mathbf{1} \le \mathbf{i}.$$

Соответственно, контекст будет определяться как **Cx**={**a** | **a**∈ **Actions**, **a**∈ **actions**$_{\mathbf{i,Obj}}$}.

Разумеется, в качестве требования может выступать и принадлежность или непринадлежность **a** множеству **actions**$_{\mathbf{i,Obj}}$[**schema**$^{\mathbf{k}}$].

В-третьих, можно сформулировать требование, согласно которому контекст состоит из действий, оперирующих объектом **Obj**. Множество таких действий будем обозначать **actions**$_{\mathbf{Obj}}$:

$$\mathbf{actions}_{\mathbf{Obj}} = \bigcup_{\forall \mathbf{i}(\mathbf{1} \le \mathbf{i})} \mathbf{actions}_{\mathbf{i,Obj}}.$$

Определение соответствующего контекста: **Cx**={**a** | **a**∈ **Actions**, **a**∈ **actions**$_{\mathbf{Obj}}$}.

Такое условие определяет, что все действия, составляющие контекст, должны оперировать объектом **Obj**. Для значимого контекста требование непринадлежности такому множеству фактически исключает объект из рассмотрения. То есть требование **SCx** = {**a** | **a**∈ **Actions**, **a**∉ **actions**$_{\mathbf{Obj}}$} в действительности постулирует абстрагирование от одного определенного объекта предметной области (посредством абстрагирования от операций с ним).

В-четвертых, требование может ограничить контекст действиями, оперирующими объектом **Obj** и, кроме того, порожденными от одной конкретной схемы (а не от любой, как в предыдущем случае). Множество таких действий обозначим **actions**$_{\mathbf{Obj}}$[**schema**$^{\mathbf{k}}$]:

$$\mathbf{actions}_{\mathbf{Obj}}[\mathbf{schema}^{\mathbf{k}}] = \bigcup_{\forall \mathbf{i}(\mathbf{1} \le \mathbf{i} \le \mathbf{k})} \mathbf{actions}_{\mathbf{i,Obj}}[\mathbf{schema}^{\mathbf{k}}].$$

Соответствующее определение контекста: **Cx**={**a** | **a**∈ **Actions**, **a**∈ **actions**$_{\mathbf{Obj}}$[**schema**$^{\mathbf{k}}$]}.

Так как характеристическое свойство задано посредством операции принадлежности или непринадлежности какому-либо множеству, можно строить из предложенных множеств (а также из индивидуальных действий) более сложные требования к контексту, используя операции над множествами.

Способ задания контекста посредством характеристического свойства достаточно гибок. Переходя к более сложным моделям задачи планирования, можно составлять характеристические требования другого рода – оперирующие понятиями новой модели. Например, если рассматривать действия как сущности, обладающие предусловием и эффектом (именно так обычно определяются действия в планировании), то можно ввести ограничения и на эти элементы. Кроме этого, оперируя объектами, можно осуществлять отбор по их характеристикам.

Выявление значимого контекста на базе правил

Гибким способом выявления значимого контекста является создание эвристик, опирающихся на набор декларативных правил, которые определяют связь между формальной постановкой задачи и контекстом. Декларативное представление знаний позволяет без изменения алгоритмов настраивать планировщик на работу с новыми предметными областями и задачами. Такой подход предполагает наличие единого механизма интерпретации правил, а значит, и наличие некоторого формального языка описания правил.

Рассмотрим один из языков такого рода, названный *языком декларации принципа абстрагирования* (ЯДПА).

ЯДПА – язык сценариев. Значимый контекст выявляется при помощи выполнения инструкций и операций. Инструкции отвечают за управление последовательностью выполнения операций и именование результатов операций. Операции формируют значимый контекст, а также промежуточные вспомогательные множества.

ЯДПА содержит три инструкции: условие, переход и присваивание. Инструкция «условие» имеет вид *Если (идентификатор) инструкция* и интерпретируется следующим образом. Если соответствующее идентификатору множество не пусто, выполняется инструкция после скобок.

Инструкция «переход» имеет вид *Переход метка_перехода.*

Интерпретация данной инструкции: следующей будет выполнена инструкция, помеченная

меткой перехода.

Инструкция «присваивание» имеет вид *идентификатор = операция.*

Ее интерпретация: полученное в результате выполнения операции множество будет иметь имя, заданное идентификатором. Выполнение операций всегда связано с инструкцией присваивания.

Инструкции должны формировать сценарий таким образом, чтобы всякая последовательность выполнения операций приводила к формированию специального множества с именем *SCx*, соответствующего значимому контексту.

Инструкции «условие» и «переход» позволяют реализовать операцию транзитивного замыкания при построении множеств, а также осуществлять классификацию задач планирования.

Операции ЯДПА содержательно идентичны операциям реляционной алгебры. В язык включены операции выборки, проекции, декартова произведения, объединения, пересечения и разности. Дополнительно введены еще две операции: выборка одного произвольного элемента и (реляционная) группировка с выборкой одного произвольного элемента в каждой группе.

Операции можно выполнять над так называемыми базовыми множествами или множествами, полученными при помощи ранее выполненных операций.

Базовыми считаются следующие множества.

- Множество всех объектов (*Objects*) из постановки задачи планирования. Элементами этого множества являются кортежи из двух элементов <имя объекта, непосредственный тип объекта>.
- Множество всех конкретизированных действий (*Actions*) задачи планирования. Это множество состоит из кортежей, длина которых больше максимальной арности схем на единицу. Первый элемент кортежа содержит имя действия, остальные – аргументы (если количество аргументов у действия меньше длины кортежа, оставшиеся элементы кортежа принимают специальное значение «пусто»). Результирующее множество *SCx* должно являться подмножеством *Actions*.
- Для каждого предиката **P** арности **k** в определении домена существует пара множеств **InInit_P** и **InGoal_P**, элементами которых могут быть кортежи длины **k**. Элементами данных множеств являются кортежи имен объектов. Если начальное состояние задачи содержит атомарную формулу, производную от **P**, аргументы этой формулы составляют кортеж, который является элементом **InInit_P**. Аналогично для цели задачи. Такая формулировка налагает ограничение на спектр задач, к которым может быть применен метод – начальное состояние и цель задачи планирования должны определяться списком атомарных формул (или их конъюнкцией).

Все базовые множества могут быть построены автоматическими средствами на основании формулировки задачи планирования.

Элементы кортежей именованы. В **Objects** первый элемент именуется **obj**, второй – **type**. В **Actions** первый элемент именуется **aName**, остальные – **arg1, arg2, …, argN**, где **N** – максимальная арность схем действий. Во множествах вида **InInit_P** и **InGoal_P** элементам кортежей присваиваются имена аргументов предиката **P** согласно его описанию в домене. Во всех операциях, за исключением проекции, разрешается переименование элементов кортежа, формируемого в результате выполнения операции.

В качестве примера использования ЯДПА рассмотрим процесс отбора полезных действий **stack** из домена *Blocks World*. Если в цели задачи указан атом (**on x, y**), то действие **stack(x, y)** является полезным. Для получения множества всех таких действий сначала из **Actions** отбираются действия **stack**:

StackActions=Выборка(<aName, arg1, arg2> | AllActions | aName='stack').

Затем строится декартово произведение этого множества с множеством кортежей объектов для предиката **ON** в цели:

Stack_x_OnInGoal = ДекПроизв(<aName, arg1, arg2, x, y> | StackActions, InGoal_ON).

Далее из этого множества отбираются те записи, у которых **arg1=x** и **arg2=y**:

StackEx=Выборка(<aName, arg1, arg2, x, y> | Stack_x_OnInGoal | arg1=x, arg2=y).

Формируется окончательное множество действий **stack** (без лишних столбцов).

FinalStack=Проекция(<aName, arg1, arg2> | StackEx).

Результаты экспериментов

Чтобы убедиться в жизнеспособности подхода, были проведены эксперименты, в ходе которых оценивались следующие показатели: количество решенных задач, время поиска плана, длина плана.

Сопоставлялись показатели, полученные при помощи оригинального планировщика **FF** версии 2.3 [3] и этого же планировщика, дополненного модулем абстрагирования. В качестве тестовых задач взяты *strips*-задачи с соревнований планировщиков IPC. Эксперименты проводились на доменах *Blocks World* (мир блоков), *Depots* (склады), *Satellite* (спутник).

Blocks World (4 оператора)

На втором соревновании IPC в качестве тестов использовались задачи для мира кубиков. Трек состоял из 35 задач (конфигурации от 4 до 17 кубиков). Эксперимент также проводился на 67 внеконкурсных дополнительных задачах (от 17 до 50 кубиков).

После абстрагирования значимый контекст

составляли действия **stack(x, y)**, если цель задачи содержала атом **(on x, y)**; действия **unstack(x, y)**, если начальное состояние задачи содержало атом **(on x, y)**; все действия **pickup** и **putdown**.

В эксперименте задача считалась нерешенной, если планировщик работал более 10 минут или тратил более 1 Гб ОЗУ.

Оригинальный **FF** не решил 25 задач (причем

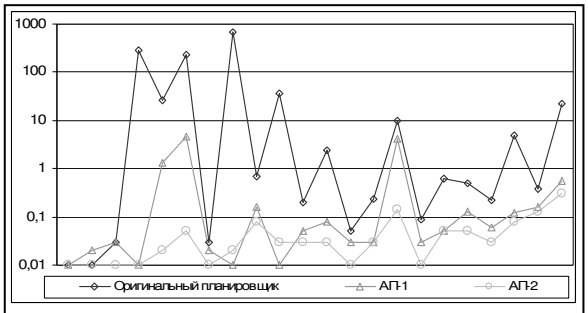


Рис. 1. Домен Depots. Время поиска решения планировщиками на тестовом треке

4 нерешенные задачи из основного трека). Абстрагирующий планировщик решил все задачи. Для совместно решенных задач планировщики получали планы почти одинаковой длины и (за редким исключением) схожее время работы.

Depots

Домен «склады» является объединением хорошо известных логистического домена (*Logistics*) и мира кубиков. Задачи домена «склады» использовались в качестве тестов на третьем соревновании IPC. Трек состоял из 22 задач.

В экспериментах были опробованы два принципа абстрагирования. Первый позволял выгружать и складировать ящики только в пункте их назначения (АП-1). Второй дополнял первый принцип тем, что грузить ящики позволялось только в месте их изначального нахождения и допускалось использование только одного грузовика (АП-2). Результаты представлены на рисунке 1. *(На рисунках 1 и 2: ось абсцисс – задачи, ось ординат (логарифмическая) – время поиска решения в секундах.)*

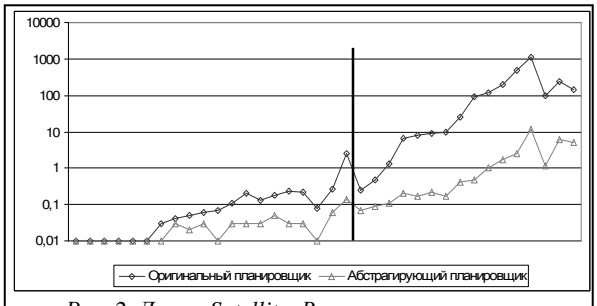


Рис. 2. Домен Satellite. Время поиска решения планировщиками на треках: тестовом – слева от вертикальной черты и внеконкурсном – справа

Планировщики с абстрагированием не только продемонстрировали выигрыш по времени, но и на многих задачах сгенерировали более короткие планы.

Эксперимент для домена «склады» был продолжен на наборе внеконкурсных задач (22 задачи из каталога *HandCoded*) значительно большего размера. Для этого блока задач был выставлен лимит в 10 минут и ограничение на память в 1 Гб. Из внеконкурсных задач оригинальному планировщику удалось решить только одну (первую). АП-1 решил 8 задач; АП-2 – 12 задач. Самый длинный план был получен АП-2 на 17-й задаче – 388 шагов (за 531 сек.).

Satellite

Задачи состоят в планировании наблюдений для группы спутников, имеющих различное оборудование. Домен «спутник» был тестовым треком на третьем соревновании IPC. Трек состоял из 20 задач. Дополнительно в экспериментах с абстрагированием использовались 16 внеконкурсных задач (каталог *HandCoded*).

Основу принципа абстрагирования составило ограничение множества поворотов спутников на цели и ограничение количества используемых инструментов. Инструменты выбирались по одному для каждого режима съемки, а повороты спутник осуществлял только на цели съемки, калибровки или глобальные цели задачи, указывающие окончательную ориентацию спутников.

И оригинальный **FF**, и абстрагирующий планировщик решили все задачи. Результаты экспериментов показаны на рисунке 2.

В заключение отметим: эксперименты показывают, что использование фокусировки в задачах планирования может давать значительный выигрыш по времени поиска решения за счет сокращения поискового пространства. Разработанный подход на базе правил, определяющих связь между формальной постановкой задачи и релевантными для ее решения действиями, может быть развит для поддержки широкого спектра элементов современного PDDL, что еще более усилит его дифференцирующие возможности. Предложенная модель фокусировки достаточно обща и допускает множество альтернативных путей определения и выявления значимого контекста.

Литература

1. Holte R.C., Choueiry B.Y. Abstraction and reformulation in artificial intelligence // Philosophical Transactions of the Royal Society B: Biological Sciences. 2003. 358(1435). pp. 1197–1204.

2. Zucker J.-D. A grounded theory of abstraction in artificial intelligence // Ibidem. 358(1435). pp. 1293–1309.

3. Hoffmann J., Nebel B. The FF Planning System: Fast Plan Generation Through Heuristic Search // Journal of Artificial Intelligence Research. 2001. № 14. pp. 253–302.

ТЕХНОЛОГИЯ ИЗВЛЕЧЕНИЯ ИНФОРМАЦИИ ИЗ ТЕКСТОВ, ОСНОВАННАЯ НА ЗНАНИЯХ

(Работа поддержана РФФИ, проект 09-07-00407, и программой фундаментальных исследований Президиума РАН № 3, проект «Высокопроизводительные масштабируемые средства работы с фактографическими базами большого объема»)

Д.А. Кормалев; Е.П. Куршев, к.т.н.; Е.А. Сулейманова; И.В. Трофимов
(ИПС РАН, г. Переславль-Залесский, dkormalev@acm.org)

KNOWLEDGE-BASED INFORMATION EXTRACTION TECHNOLOGY

Kormalev Dmitry A.; Kurshev Evgeny P., Ph.D.; Suleymanova Elena A.; Trofimov Igor V.
(Organization of Russian Academy of Sciences Ailamazyan Program Systems Institute of RAS, Pereslavl-Zalessky , dkormalev@acm.org)

Abstract. The paper deals with the development of methods to extract information from natural language text. Emphasis is made on knowledge representation and recognition of textual situations. The paper also gives a brief description of experimental implementation of the above techniques in ISIDA-T system.

Keywords: *information extraction, natural language processing, knowledge representation.*

Статья посвящена усовершенствованию методов извлечения информации из текста на естественном языке. Основной акцент сделан на представлении знаний и распознавании текстовых ситуаций. Приводится описание экспериментальной реализации в системе ИСИДА-Т.

Ключевые слова: *извлечение информации, анализ естественного языка, представление знаний.*

Значительная доля информации, доступной в электронном виде, представлена текстами на естественном языке. Заключенная в них полезная информация неструктурирована, а значит, ее невозможно обработать и проанализировать классическими вычислительными методами и средствами.

Технология извлечения информации (ТИИ) из текстов на естественном языке позволяет автоматически просматривать относительно большой объем текстов, содержащих сравнительно небольшое количество искомой информации. Обнаруженная в тексте информация преобразуется в структурированный формат: выявляются целевые факты, объекты, отношения в виде, пригодном для дальнейшей автоматической обработки (статистической обработки, визуализации, поиска закономерностей в данных и др.).

Иногда ТИИ рассматривают как специфическую разновидность информационного поиска. Отличия ТИИ от информационного поиска заключаются в том, что запросы должны быть известны заранее, результатом же является не набор ссылок на документы, а построенные структуры данных, описывающие релевантные факты из набора документов.

Приведем некоторые области применения ТИИ:

- расширение возможностей информационного поиска (поиск не по ключевым словам, а по фактам, ситуациям, объектам, отношениям);
- построение досье на персон или организации из открытых текстовых источников;
- мониторинг сообщений СМИ (примеры событий, которые могут представлять интерес: слияние и поглощение компаний, появление новых игроков на рынке, выпуск новой продукции, теракты);
- извлечение специфической метаинформации из коллекций документов большого объема (например, построение по текстовой базе муниципальных нормативно-правовых актов, связанных с недвижимостью; реляционной БД с информацией о типах событий, объектах и субъектах).

Первоначально задача ТИИ формулировалась как выделение фрагментов текста, содержащих релевантную информацию, и, возможно, преобразование их в реляционную форму. Для решения задачи в такой постановке часто достаточно анализировать локальный контекст, используя ограниченный набор знаний предметной области. Назовем такую технологию *извлечением информации в слабом смысле*. Результаты извлечения информации в слабом смысле и характер их представления несколько ограничивают возможности дальнейшего использования добытых из текста данных. *Извлечением информации в сильном смысле* назовем переход от базы текстовых фактов к такому их представлению, которое можно было бы использовать как интеллектуальный информационный ресурс, своего рода базу текстовых знаний.

Исследования авторов были направлены на усовершенствование методов и расширение возможностей ТИИ, что позволило бы вплотную подойти к решению задачи извлечения информации в сильном смысле. Полигоном для экспериментальной проверки идей и практического воплощения разработанных подходов стала система ИСИДА-Т (*интеллектуальная система извлечения данных и их анализа (для обработки текстов)*), над которой ведется работа в течение нескольких лет.

Чтобы получить информацию из прочитанно-

го фрагмента текста (понять текст), человек должен знать язык, на котором написан текст, и располагать некоторым объемом «фоновых» знаний. Аналогично система извлечения информации из текста должна располагать средствами анализа естественного языка и некоторым объемом знаний предметной области.

Общая организация системы

Краеугольным камнем системы ИСИДА-Т является точная настройка на предметную область и конкретную задачу извлечения. Это достигается за счет редактирования лингвистических ресурсов, ресурсов знаний, правил извлечения и правил трансформации. Настройка может потребовать также включения в процесс обработки дополнительных специализированных методов обработки текста. Кроме того, для каждой задачи необходимо подобрать наиболее подходящие алгоритмические средства анализа из набора имеющихся. Эти аспекты требуют создания такой архитектуры, при которой легко могут добавляться и замещаться алгоритмические компоненты процесса извлечения.

Конфигурирование на алгоритмическом уровне потребовало создания модульной архитектуры и декларативного подхода к определению процесса извлечения. Модули получили название обрабатывающих ресурсов в противовес лингвистическим ресурсам и ресурсам знаний. В конфигурации декларируются порядок обработки документа аналитическими модулями, потоки данных между ними, а также параметры их работы.

Обрабатывающие ресурсы можно разделить на следующие группы.

- Ресурсы предобработки. К ним относятся средства определения кодировки документа, извлечения текста и стилевой разметки из документа, предварительной фильтрации.
- Ресурсы лингвистического анализа. Осуществляют разбор текста на отдельные слова, морфологический анализ (в том числе специализированные варианты для различных категорий имен собственных), поверхностный синтаксический анализ и определение границ предложений.
- Ресурсы извлечения. Осуществляют поиск в документе целевой лексики и синтаксических конструкций, а также первичное структурирование информации.
- Ресурсы унификации знаний и вывода. Осуществляют унификацию и отождествление элементов знаний, вывод производных знаний.
- Ресурсы подготовки результата. Осуществляют приведение извлеченной информации к определенному формату и передачу за пределы последовательности обработки (в БД, глобальный ресурс знаний, файл, приложение).

Средства анализа естественного языка

Используемые в ТИИ средства для анализа естественного языка, можно разделить на две большие категории: средства общего лингвистического анализа и предметно-ориентированные методы распознавания текстовых ситуаций.

Средства общего лингвистического анализа включают графематический, морфологический и синтаксический анализ. Эти средства применимы практически во всех предметных областях, существует ряд реализаций с довольно высокими показателями качества, поэтому останавливаться на них подробно нет необходимости.

Распознавание текстовых ситуаций состоит в выделении фрагментов текста, описывающих объекты, и содержательных связей между этими фрагментами, в той или иной мере основанных на синтаксисе естественного языка. Можно рассматривать распознавание ситуаций как ориентированный на предметную область частичный, но точный синтактико-семантический анализ.

Распознавание основывается на сопоставлении с образцом, заданным при помощи правил на специализированном формальном языке. Правила определяют не только образец, но и действия, которые должны быть выполнены при успешном сопоставлении. Правила работают не с текстом как последовательностью символов, а со структурами, построенными над текстом и выражающими лингвистическую и предметную информацию о нем.

Для упрощения конфигурирования системы желательно, чтобы все модули использовали одинаковый способ представления информации о тексте (разметки текста). В системе ИСИДА-Т все модули, в том числе средства общего лингвистического анализа, используют структуры данных.

Разметка текста и структуры данных

В различных системах обработки текста на естественном языке используется широкий спектр средств для представления лингвистической и предметно-ориентированной информации о тексте в целом или его фрагментах. Единого подхода к представлению разметки текста и информации о нем не существует.

В последнее десятилетие довольно широко используется способ представления информации о тексте, основанный на так называемых *аннотациях*, отличающийся простотой и высокой степенью универсальности [1]. Сегодня многие системы обработки текста в той или иной степени используют идеи модели аннотаций.

Аннотация – объект, который приписывается фрагменту текста (например, слову, словосочетанию, предложению, ссылке на сущность предметной области и т.д.) и описывает свойства этого фрагмента. Аннотации разбиты на конечное множество классов. Каждый класс аннотаций описывает текст в определенном аспекте. Информация о фрагменте представлена значениями именованных

атрибутов аннотации. Наборы классов и атрибутов аннотаций намеренно не специфицированы, чтобы можно было использовать произвольный набор обрабатывающих модулей и представлять необходимую лингвистическую и предметную информацию. Обмен данными между модулями тоже идет в терминах аннотаций: новые аннотации могут строиться на основании полученных на предыдущих этапах анализа.

Представление информации с помощью аннотаций дает возможность разрабатывать средства анализа текста, компоненты которых слабо связаны между собой. Не отражающееся на функциональных характеристиках сложной системы уменьшение числа зависимостей между ее составляющими облегчает ее понимание, разработку и поддержку. Слабая связность – это существенное преимущество, так как она повышает возможность повторного использования компонентов и снижает риск критических сбоев, вызванных неправильным взаимодействием компонентов (например из-за того, что в цепочке обработки какой-то компонент ошибочно не был зарегистрирован или же частично нарушился порядок обработки).

Впрочем, базовая модель аннотаций не лишена недостатков. В частности, в ней не предусмотрены средства проверки соответствия атрибутов и их значений. Атрибуты могут быть только атомарными. Отсутствует возможность установления связей между отдельными аннотациями. Нет средств для контроля расположения границ аннотаций разных классов, в то время как для большинства классов аннотаций можно задать условия, описывающие их взаимное расположение. Например, аннотации, описывающие синтаксис предложения в терминах системы составляющих, не могут пересекаться – для них возможно только отношение строгого вхождения или совпадения.

В реализации системы ИСИДА-Т модель аннотаций дополнена некоторыми полезными средствами. В частности, было снято ограничение на атомарность атрибутов и добавлена возможность устанавливать ссылки между аннотациями.

Язык правил распознавания текстовых ситуаций

Для распознавания текстовых ситуаций используется набор правил, описывающих характерные для конкретной задачи способы выражения ситуации в тексте. Эти правила задают образец для сопоставления и действия, которые должны быть выполнены после успешного сопоставления. Качество работы (полнота и точность) ТИИ тесно связано с возможностями языка правил. Ряд современных систем извлечения информации (в том числе система ИСИДА-Т) берут за основу различные диалекты языка CPSL [2]. Использование этого языка подразумевает разметку текста при помощи аннотаций.

Единицей трансляции языка правил является фаза. Правила, входящие в одну фазу, применяются в недетерминированном порядке. Результаты фазы – изменения, внесенные в набор аннотаций после работы правил, – фиксируются после применения всех правил и становятся доступными в последующих фазах. Поэтому правило не может использовать результаты работы другого правила из этой же фазы. Можно рассматривать фазу как модуль для специфического анализа текста. Работа фаз может перемежаться применением произвольных обрабатывающих ресурсов.

Правило – основная единица языка. Правила представляются в виде «образец→действие». Здесь «образец» – образец для поиска в терминах высказываний о взаимном расположении и значениях атрибутов аннотаций разных классов (левая часть правила); «действие» – набор действий, выполняемых при успешном сопоставлении (правая часть правила). По структуре левая часть правила во многом схожа с регулярным выражением, но существенное отличие состоит в том, что роль символов в правиле играют тесты. Тест представляет собой конъюнкцию высказываний (элементарных тестов) о значениях атрибутов аннотаций разных классов. Из тестов могут образовываться сложные конструкции с использованием следования, альтернативы, квантификаторов и скобок. Чтобы обозначить границы фрагментов текста, сопоставленных подвыражениям, используются метки. Метка – это идентификатор, которым помечается образец. В дальнейшем (при выполнении действий в правой части правила) можно использовать метку для ссылки на фрагмент текста, сопоставленный подвыражению.

Язык правил, используемый в системе ИСИДА-Т, является расширением CPSL. Предлагаемые нами расширения преследуют две цели: обеспечить возможность описывать более сложные контексты, в которых встречается целевая информация, и снизить объем рутинной работы при создании системы правил за счет более компактного описания контекста. Расширения включают в себя поддержку дополнительных типов данных, большего числа квантификаторов и метасимволов. Помимо этого, реализована поддержка переменных и списков значений, существуют гибкие возможности проверки взаимного расположения аннотаций.

Общая проблема средств распознавания текстовых ситуаций (при расширении функциональных возможностей этих средств) – резкое снижение производительности. Для решения проблемы использовались два основных способа оптимизации интерпретатора правил: предобработка правил путем анализа потоков управления и сокращение перебора кандидатов при выполнении тестов [3]. Внедрение этих модификаций позволило ускорить интерпретацию правил в среднем в 6 раз

в зависимости от конфигурации системы и качества входных данных (в отдельных случаях наблюдался прирост производительности до двух порядков). В большинстве случаев повышение производительности сопровождалось снижением расхода памяти на 20–40 %.

Ресурс знаний

Практически в любой предметной области для точного извлечения требуются априорные знания о ней – знания о понятиях, объектах и отношениях, связанных с целями извлечения или являющихся целями. В свою очередь, извлеченная из текстов информация может содержать новые знания о предметной области и быть полезной для дальнейшей автоматической обработки текста. Тесная связь между априорной и извлеченной информацией, а также между предметными и лингвистическими знаниями сформировала потребность в унификации средств представления.

Представление знаний. Интегрированный *ресурс знаний* (РЗ) системы ИСИДА-Т объединяет в себе базу априорных предметных знаний, хранилище фактографической информации и словарь. Предметные знания хранятся в РЗ в структурах, называемых *элементами знаний*. Элементы знаний делятся на 4 категории: концепты (**CT**), экземпляры концептов (**CI**), типы предметных отношений (**RT**), экземпляры отношений (**RI**). В данной работе в подходе к представлению знаний используются элементы семантических сетей и систем фреймов.

Концепты и типы отношений служат для представления онтологической информации о предметной области и задаются априорно. Экземпляры концептов и отношений составляют базу фактов предметной области и могут быть как априорными, так и извлеченными из текстов.

Для каждого элемента знаний задается набор атрибутов. В списках атрибутов **CT** и **RT** хранятся пары «имя–ограничения на значение», в списках атрибутов **CI** и **RI** – пары «имя–значение». В терминах системы фреймов **CT** и **RT** выражались бы прототипами фреймов, а **CI** и **RI** – экзофреймами. Неявно определены два специальных (служебных) типа отношений: **ISA** и **AKO**. Их интерпретация такая же, как в системах фреймов.

Лингвистическая составляющая ресурса знаний – словарь. Словарь связан с базой предметных знаний посредством ссылок от дескрипторов к элементам знаний: дескрипторы словаря базовой лексики ссылаются на концепты, а дескрипторы словаря собственных имен – на априори известные экземпляры концептов из базы фактов. В отличие от тезауруса дескрипторы в словаре базовой предметной лексики не связаны друг с другом никакими парадигматическими отношениями (последние выражаются с помощью отношений между соответствующими элементами базы предметных знаний).

Словарь предоставляет возможность указывать дополнительные ограничения на все словоформы, входящие в состав дескриптора и синонимов, чтобы увеличить точность распознавания словарных единиц в тексте.

Унификация априорных и извлеченных из текстов знаний удобна тем, что позволяет использовать одни и те же средства для работы с обоими типами знаний. Объединение лингвистических и предметных знаний в одном ресурсе, во-первых, облегчает первичное наполнение и последующую поддержку, а во-вторых, дает возможность использовать предметные знания уже на этапе первичной обработки текста правилами извлечения информации. Благодаря специально разработанному языку запросов к РЗ правила могут не ограничиваться словарной информацией, а обращаться в онтологию и базу фактов для проверки различных условий, требующих навигации по отношениям.

Трансформации. После извлечения информации из текста и помещения ее в хранилище фактографической информации часто требуется дополнительная обработка для ее унификации и уточнения. На основе такой обработки может решаться целый спектр задач:

- навигация по связанным объектам, фактам и ситуациям;
- определение и объединение тождественных элементов;
- кластеризация сходных сюжетов;
- вывод имплицитной фактографической информации;
- генерация текстовых описаний фрагментов фактографической базы.

Для проведения экспериментов по преобразованию извлеченной фактографической информации был разработан язык трансформаций и выполнена экспериментальная программная реализация интерпретатора этого языка.

Трансформацию элементов ресурса знаний можно рассматривать как особый вид немонотонного вывода на знаниях. При трансформации происходят поиск образца ситуации в ресурсе знаний и выполнение указанных действий. Для описания ситуации можно задавать ограничения на типы элементов знаний, их атрибуты, наличие или отсутствие отношений того или иного типа между ними. Попытка выполнить действия производится для каждого набора элементов знаний, для которых выполняются условия, указанные в посылке правила трансформации. Набор действий включает создание, удаление, модификацию элементов знаний, манипулирование их атрибутами.

Особенностью языка правил трансформации является сочетание декларативных и императивных элементов.

Для эффективного выполнения трансформаций были разработаны оригинальные алгоритмы предобработки правил трансформации и подготовки вспомогательных структур в ресурсе знаний, с которым будет идти работа.

Глобальный РЗ (ГРЗ). Информация, извлеченная из одного документа, помещается в РЗ, ассоциированный с рабочим процессом, – так называемый локальный РЗ. Если требуется выполнять параллельную, асинхронную обработку большой коллекции документов, извлеченная информация помещается в специальное хранилище – ГРЗ. ГРЗ обеспечивает целостность извлеченной информации. Он не содержит промежуточных результатов извлечения, появление которых допустимо в локальных РЗ. Помимо этого, ГРЗ по специальным правилам устанавливает тождественность сущностей и отношений, извлеченных из разных документов.

Структурными элементами ГРЗ являются все те же элементы знаний. Однако архитектурное устройство ГРЗ кардинально отличается от локального варианта. Во-первых, ГРЗ рассчитан на хранение больших объемов информации и использует дисковые накопители. Во-вторых, объем информационных потоков не должен быть ограничен как со стороны параллельно работающих систем извлечения, так и со стороны пользователей извлеченной информации. Поэтому архитектура ГРЗ допускает масштабирование по модулям обслуживания запросов со стороны внешних систем. Масштабироваться также должен процесс отождествления сущностей, извлеченных из разных документов. В противном случае ГРЗ будет иметь ограничение по объему входных данных, причем с их ростом в самом ГРЗ это ограничение будет становиться все более жестким.

В заключение отметим, что описанные в статье методы и подходы могут найти применение в технологических цепочках хранилищ знаний, для построения и наполнения ресурсов знаний разного рода, для повышения точности и обогащения результатов работы поисковых машин. Методы обработки текста и работы со знаниями, реализованные в системе ИСИДА-Т, создают основу для средств извлечения информации в сильном смысле. Такие средства не ограничиваются разметкой текста, они подразумевают переход от корпуса текстов к такому представлению фактографической информации, которое можно было бы использовать как интеллектуальный информационный ресурс – своего рода базу текстовых знаний.

Литература

1. Grishman R. TIPSTER Text Architecture Design. Version 3.1. New York: NYU, 1998.

2. Appelt D.E. The Common Pattern Specification Language: Technical report / SRI International, Artificial Intelligence Center. 1996.

3. Кормалев Д.А. Повышение производительности при распознавании текстовых ситуаций: тр. Одиннадцатой нац. конф. по искусствен. интел. с междунар. участием. М.: ЛЕНАНД, 2008. Т. 2. С. 192–200.

МОРФОЛОГИЧЕСКИЙ АНАЛИЗ НЕЗНАКОМЫХ ФАМИЛИЙ В РУССКОЯЗЫЧНОМ ТЕКСТЕ

(Работа поддержана РФФИ, проект 09-07-00407, и программой фундаментальных исследований Президиума РАН № 3, проект «Высокопроизводительные масштабируемые средства работы с фактографическими базами большого объема»)

Е.А. Сулейманова, К.А. Константинов

(ИПС им. А.К. Айламазяна РАН, г. Переславль-Залесский, yes@helen.botik.ru)

MORPHOLOGICAL ANALYSIS OF UNKNOWN SURNAMES IN RUSSIAN-LANGUAGE TEXTS

Suleymanova Elena A.; Konstantinov Konstantin A.

(Organization of Russian Academy of Sciences, Pereslavl-Zalessky, yes@helen.botik.ru)

Abstract. The paper suggests a method for morphological analysis of unknown surnames in Russian-language texts. The method has been implemented as part of the Isida-T system for intelligent text analysis. The method consists in generating a redundant set of hypotheses which is then being filtered using rules and additional data from the text.

Keywords: *morphological analysis of proper names, hypotheses filtering.*

В статье описывается подход к морфологическому анализу незнакомых фамилий в русскоязычном тексте, реализованный в специальном модуле системы интеллектуального анализа текста «Исида-Т». Метод состоит в построении заведомо избыточного множества вариантов-гипотез, которое затем подвергается фильтрации с помощью правил и с привлечением данных текста.

Ключевые слова: *морфологический анализ собственных имен, фильтрация гипотез.*

Любая система, работающая с текстом и использующая морфологический анализатор со словарем основ, сталкивается с проблемой обработки незнакомых слов – лексем, по тем или иным причинам отсутствующих в словаре. Классический труд А.А. Зализняка [1], лежащий в основе боль-

шинства современных машинных морфологий русского языка, содержит описания около 100 тысяч слов. Размеры морфологических словарей, используемых в различных прикладных системах, варьируются довольно широко: от 90 тысяч [2] до более 161 тысячи лемм [3]. Однако и при увеличении объема словаря проблема анализа незнакомых слов сохраняется. Одна сторона этого явления связана со словообразовательными возможностями языка, другая – с существованием принципиально открытых категорий лексики (термины, диалектизмы, профессионализмы, разные виды слэнга, разговорная лексика и, наконец, имена собственные).

В контексте задачи извлечения информации из текстов последние представляют наибольшую важность, поскольку Ф.И.О. лиц, названия всевозможных организаций, названия географических и административных объектов служат для идентификации участников фактов.

Современные морфологические анализаторы обычно снабжаются специальными механизмами, позволяющими определять морфологию незнакомой словоформы с приемлемым качеством. Эти механизмы основываются либо на некоторой вероятностной модели, либо на правилах [2–4].

В системе интеллектуального анализа текста «Исида-Т», разрабатываемой в ИПС РАН, используется морфологический анализатор «Диалинг» (www.aot.ru). По данным [3], предсказание в нем обеспечивает до 87 % правильно распознанных незнакомых словоформ для русского языка, из которых большинство составляют существительные и прилагательные (такой уровень вполне сопоставим с результатами других исследователей, полученными для английского и французского языков). Однако, что касается фамилий, то для корректного морфологического предсказания в этой области как минимум требовалось бы, чтобы словарь располагал весьма представительным массивом фамилий всех возможных словоизменительных типов. На практике приходится сталкиваться с тем, что фамилии при распознавании часто подводятся под похожую форму существующего или гипотетического существительного или прилагательного, а при отсутствии таковых воспринимаются как неизменяемые существительные. Так, фамилии на *-ов* в именительном падеже часто распознаются как форма родительного падежа множественного числа некоторого существительного (для *Чемезов* выводится каноническая форма *Чемез*), в то время как с формами косвенных падежей проблем почти не бывает; для фамилий типа *Малых*, *Седых* не строится вариант с правильной канонической формой (все они приводятся к прилагательным – *малый*, *седой*). Иногда случаются курьезы: финская фамилия *Хуовинен* распознается как краткая форма прилагательного *хуовинный*.

Нестабильность результатов работы встроенного модуля морфологического предсказания на фамилиях послужила поводом для разработки собственных средств.

Описание подхода

Общий алгоритм морфологического анализа фамилий делится на несколько этапов.

1. ***Первичный анализ.*** Порождение заведомо избыточного множества гипотез – вариантов разбора для фамилий, встретившихся в тексте вместе с именами (и, возможно, отчествами). В качестве данных используется таблица словоизменительных классов фамилий. Гипотезы строятся с учетом грамматических характеристик имени (и отчества) на основании совпадения правого конца словоформы фамилии с содержимым ячейки таблицы. Гипотез заведомо больше, чем правильных вариантов: наряду с регулярными неоднозначностями морфологический шум создают случайные совпадения. Каждой гипотезе приписывается индекс совпадения (число букв в ячейке, с которой успешно сопоставилась словоформа). Индекс совпадения можно использовать в качестве крайнего средства для выбора из множества вариантов в отсутствие дополнительных текстовых данных.

2. ***Исключение некорректных вариантов с помощью правил.*** Правила позволяют исключить небольшое число запланированных ошибочных совпадений (фактически проверка определенных условий внесена во второй этап, чтобы не усложнять первый).

3. ***Фильтрация результатов путем сравнения данных из одного текста.*** Этот этап несет основную нагрузку в отсеивании морфологического шума и разрешении регулярных (и часто не разрешимых другими средствами) неоднозначностей.

4. ***Построение для анализируемых фамилий новых аннотаций*** класса *Morpho* взамен построенных штатной морфологией.

5. ***Морфологический анализ изолированных фамилий***, не охваченных предыдущими этапами.

На вход алгоритму поступают аннотации класса ANE, построенные *psl*-правилами для встретившихся в тексте цепочек вида [5]:
<имя> (возможно, <отчество>) <слово с заглавной буквы>.

На выходе каждой текстовой форме фамилии приписано множество аннотаций *Morpho*, содержащих в значениях соответствующих атрибутов каноническую форму, число, род, падеж, имя словоизменительного класса.

Каждая такая аннотация представляет собой вариант морфологического разбора фамилии. Остановимся подробнее на каждом этапе.

Первичный анализ

Данные. Предположим, что подавляющее большинство фамилий, встречающихся в русскоя-

зычном тексте, можно описать сравнительно небольшим числом словоизменительных классов. В текущей реализации используется 59 классов. При построении системы классов учитывались как закономерности общей модели субстантивного и адъективного словоизменения в русском языке, так и особенности склонения фамилий. Грамматический род фамилии считаем классифицирующей категорией (как род у существительного, в отличие от прилагательного, где род – словоизменительная категория). Таким образом, фамилиям *Иванов* и *Иванова* соответствуют разные классы.

В качестве данных алгоритм использует таблицу словоизменительных классов фамилий. Таблица редактируема. Строки таблицы соответствуют классам. Столбцы – клеткам стандартной парадигмы существительного. Каждая ячейка, элемент парадигмы, содержит буквы русского алфавита и маркеры (скобки разных видов). Смысл содержимого ячейки – описать альтернативные условия, необходимые для соотнесения словоформы с данной ячейкой. Если правый конец словоформы совпадет с одной из альтернатив, словоформе будут приписаны соответствующая ячейке морфологическая информация (координаты клетки парадигмы) и имя класса, указанное в соответствующей строке. Пример содержимого двух ячеек для класса «м 3* ок/к муж» (обозначения типов основ и склонений отчасти заимствованы из [1]):

{б,в,д,з,л,м,н,п,р,с,т,ш,ж,ч}<ок>[] (ед. ч., м. р., им. п.)

{б,в,д,з,л,м,н,п,р,с,т,ш,ж,ч}<к>[а] (ед. ч., м. р., род. п.)

В фигурных скобках указываются альтернативные буквы. В угловых – чередование, в квадратных – окончание (в первом случае нулевое). Маркеры используются для генерации основы и канонической формы.

Примеры словоформ, которые сопоставятся с первой ячейкой: *Лубок*, *Божок*, *Черток, Павшок* и т.п. Класс «м 3* ок/к муж» описывает фамилии, у которых в основе происходит чередование «о» с нулем. Очевидно, что не каждая словоформа, успешно сопоставившаяся с первой ячейкой, на самом деле принадлежит упомянутому классу: она вполне может не иметь беглого «о» в других формах. Причем даже человек не всегда может правильно определить словоизменительное поведение фамилии по одной лишь словоформе (например, *Черток – Чертка*, *Чертку*, ... или *Чертока*, *Чертоку*, …?). Словоформы *Лубок*, *Божок*, *Черток, Павшок* сопоставятся также и с ячейкой класса без чередований «м 3 муж» с основой на {к, г, х}.

Общее описание алгоритма первичного анализа

На вход поступают текстовые Ф.И.О. с распознанным именем и, возможно, отчеством (точнее, построенные для них аннотации ANE). О фамилии неизвестно ничего, кроме ее буквенного состава.

На выходе строится текстовый файл, содержащий для каждой цепочки Ф.И.О. набор вариантов морфологического разбора входящей в нее фамилии (набор гипотез).

Вариант морфологического разбора должен включать в себя следующее: неизменяемую основу, каноническую форму, имя класса, идентификатор словоформы (в тексте), набор грамматических характеристик (число–род–падеж), число букв в ячейке класса, с которой успешно сопоставилась словоформа.

Имя и отчество уже распознаны морфологическим словарем. Предполагая, что грамматические характеристики (число, род, падеж) фамилии совпадают с характеристиками имени (и отчества), алгоритм использует эти данные для сужения области поиска в таблице и, соответственно, сокращения числа гипотез. В настоящее время анализ выполняется для цепочек Ф.И.О., включающих не более одного имени, – значит, фамилия, предположительно, стоит в единственном числе.

Проиллюстрируем работу алгоритма на примере цепочки *Евгения Гришковца*. Для нее построены три аннотации ANE, соответствующие трем грамматическим наборам омонимичного имени: «ед. ч., м. р., род. п.», «ед. ч., м. р., вин. п.» и «ед. ч., ж. р., им. п.».

1. Берем грамматический набор из аннотации ANE. Он задает пространство поиска в таблице. В пространстве поиска находим ячейку, цепочка в которой успешно сопоставляется с концом фамилии; маркеры при сопоставлении игнорируются, но при успешном сопоставлении копируются в словоформу; получаем *размеченную словоформу*.

2. *Имя класса* содержится в строке на пересечении со столбцом «Имя класса».

3. Строим *неизменяемую основу* для этого варианта разбора – берем цепочку от начала размеченной словоформы до правой фигурной скобки.

4. Строим *каноническую форму*. Тут возможны два случая в зависимости от того, какой падеж входит в рассматриваемый грамматический набор. Если падеж именительный, то каноническая форма фамилии совпадает с размеченной словоформой (если убрать из последней все маркеры). Иначе каноническая форма генерируется путем прибавления к неизменяемой основе (построенной на шаге 3) чередования и окончания из ячейки именительного падежа.

Описанные шаги выполняются по всему пространству поиска для данного грамматического набора, затем аналогично для остальных грамматических наборов. В результате получаем все возможные варианты разбора.

Если словоформу не удалось сопоставить ни с одной ячейкой в пространстве поиска, то фамилии приписывается класс «неизм муж» или «неизм жен» в зависимости от рода. В качестве неизменяемой основы и канонической формы берется

сама словоформа.

Результаты анализа Ф.И.О. *Евгения Гришковца*, полученные на этом этапе:

{Гришковц}, {ГРИШКОВЦА}, «ж 1-5b жен», (17); (Singular-F-I); (2).
{Гришковц}, {ГРИШКОВЦА}, «ж 5a жен», (17); (Singular-F-I); (2).
{Гришковц}, {ГРИШКОВЦ}, «м 1-5b муж», (17); (Singular-M-R); (2).
{Гришковц}, {ГРИШКОВЦ}, «м 1-5b муж», (17); (Singular-M-V); (2).
{Гришков}, {ГРИШКОВЕЦ}, «м 5*a ец/ц муж», (17); (Singular-M-R); (3).
{Гришков}, {ГРИШКОВЕЦ}, «м 5*a ец/ц муж», (17); (Singular-M-V); (3).
{Гришков}, {ГРИШКОВЕЦ}, «м 5*b ец/ц муж», (17); (Singular-M-R); (3).
{Гришков}, {ГРИШКОВЕЦ}, «м 5*b ец/ц муж», (17); (Singular-M-V); (3).
{Гришковц}, {ГРИШКОВЦ}, «м 5a муж», (17); (Singular-M-R); (2).
{Гришковц}, {ГРИШКОВЦ}, «м 5a муж», (17); (Singular-M-V); (2).
{Гришковц}, {ГРИШКОВЦ}, «м 1-5b муж», (17); (Singular-M-V); (2).
{Гришковц}, {ГРИШКОВЦ}, «м 1-5b муж», (17); (Singular-M-R); (2).
{Гришковц}, {ГРИШКОВЦ}, «м 5a муж», (17); (Singular-M-V); (2).
{Гришковца}, {ГРИШКОВЦА}, «неизм жен», (17); (Singular-F-I); (1).
{Гришковца}, {ГРИШКОВЦА}, «неизм муж», (17); (Singular-M-R); (1).

Исключение некорректных вариантов с помощью правил

Очевидно, что в общем случае число вариантов разбора, полученных по буквенному составу конца словоформы, значительно больше числа правильных. Цель этапа – используя дополнительные формальные критерии, исключить заведомо ошибочные варианты (исключив при этом и риск потери возможных правильных альтернатив).

В правилах делается попытка смоделировать некоторые когнитивные фильтры формальными средствами. Примеры используемых правилами функций:

let(i) – **i**-я буква *слова* от правого конца;

left(i, j) – цепочка длины **i** букв в *слове* слева от **let(j)**;

end(j) – правый конец *слова* длиной **j** букв;

trunc(k) – остаток *слова* за вычетом **k** букв справа;

length – число букв в *слове*.

Слово – переменная, которая означивается в каждом конкретном случае. Чаще всего в качестве *слова* выступает каноническая форма.

Приведем (содержательно) несколько случаев, в которых вероятен риск исключить с помощью правил варианты с классом «ов-ин муж» (к этому классу относятся фамилии типа *Иванов* и *Калинин*):

Если

(1) каноническая форма фамилии оканчивается на *-ин* и

(2) длина ее меньше 4 букв или она не содержит ни одной гласной, не считая *и* в *-ин*,

то вариант с классом «ов-ин муж» удаляется (*Бин, Скин* – в отличие от *Якин*).

Если

(1) каноническая форма фамилии оканчивается на *-ов* и

(2) длина ее меньше 4 букв или она оканчивается на *-ьов*,

то вариант с классом «ов-ин муж» удаляется (*Шольов*).

Заметим, что критерий *отсутствия гласной*, используемый в первом случае, к фамилиям на *-ов/-ев* мы не применяем, так как такое правило удалило бы правильный вариант разбора фамилий *Лбов* и *Мнев*.

Типичные дилеммы, с которыми приходится сталкиваться и которые не разрешимы формально (а часто и не только формально):

-ин – по типу *Калинин* или *Толкин*?

-ина – по типу *Калинина* или *Скори́на*? Впрочем, даже для *Ско́рина* формально нельзя исключить тип *Калинина:* это вполне допустимо при ударении на первый слог: *Ско́рина*, *у Ско́риной*, *для Ско́риной*.

Такие регулярные случаи неоднозначности отвечают лишь за часть морфологического шума, порождаемого первым этапом. Большую его часть составляют случайные совпадения словоформ с ячейками таблицы. Кое-что можно было бы побороть эвристическими правилами, но написание таких правил вручную слишком трудоемко. Поэтому удобнее разрешать подобные неоднозначности с привлечением дополнительных текстовых данных. Что касается упомянутых выше дилемм, то в таких случаях дополнительные текстовые данные – это фактически единственное средство выбрать правильный вариант.

В крайнем случае в отсутствие последних можно отсеять варианты с наименьшим индексом совпадения (в большинстве случаев это не приводит к ошибкам).

Фильтрация результатов путем сравнения данных из одного текста

Идея этапа состоит в том, чтобы обнаружить все случаи употребления одной и той же фамилии в тексте и сравнить между собой множество вариантов разбора (с точностью до имени класса, основы и канонической формы).

Первый шаг – кластеризация текстовых форм фамилий. Объединение фамилий в кластеры выполняется *алгоритмом частичного сопоставления* по файлу результатов предыдущего этапа.

Суть алгоритма очень проста: две словоформы побуквенно сравниваются слева направо. Как только находится несовпадение, сравнение прекращается. Пусть **M** – длина словоформы (наибольшая из двух), **N** – количество букв, совпавших при сравнении. Тогда две словоформы считаются успешно сопоставленными, если $\mathbf{N/M \geq K}$, где **K** – некоторый эвристически подобранный коэффициент (в нашем случае **0,6**). Например, рассмотрим фамилии *Садовничий* и *Садовничему*. **M** в данном случае будет равно **11**, **N** – **8**. Высчитаем коэффициент: **8 / 11=0,72**. Поскольку **0,72** больше, чем **0,6**, считаем, что эти фамилии успешно сопоставлены.

При кластеризации фамилий учитывается исключительно буквенное сходство их форм. Имена и отчества игнорируются. Следовательно, допус-

кается, что в один кластер попадут «одноименные» фамилии, принадлежащие лицам разного пола, то есть, в нашем понимании, разные фамилии. Это допущение позволяет использовать для фильтрации случаи омонимичного (с нейтрализацией рода) употребления.

Приведем фрагмент файла результатов, поступившего на вход третьему этапу.

Файл построен по тексту, в котором упоминаются *Валентина Распутина* и *Валентин Распутин*. Следующие текстовые формы были объединены в один кластер методом частичного сопоставления:

Валентина Распутина {mf}.
m {Распутин}, {Распутин}, «м 1-5b муж», (1); (Singular-M-R); (2).
m {Распутин}, {Распутин}, «м 1-5b муж», (1); (Singular-M-V); (2).
m {Распутина}, {Распутина}, «неизм муж», (1); (Singular-M-R); (1).
m {Распутин}, {Распутин}, «ов/ин муж», (1); (Singular-M-R); (3).
m {Распутин}, {Распутин}, «ов/ин муж», (1); (Singular-M-V); (3).
f {Распутин}, {Распутина}, «ж 1-5b жен»,(1); (Singular-F-I); (2).
f {Распутина}, {Распутина}, «неизм жен»,(1); (Singular-F-I); (1).
f {Распутин}, {Распутина}, «ова/ина жен»,(1); (Singular-F-I); (3).

Валентине Распутиной {f}.
f {Распутин}, {Распутиная}, «ж <п 1> жен», (5); (Singular-F-D); (3).
f {Распутин}, {Распутина}, «ова/ина жен», (5); (Singular-F-D); (4).

Валентином Распутиным {m}.
m {Распутин}, {Распутиный}, «м <п 1a> муж», (13); (Singular-M-T); (3).
m {Распутин}, {Распутиной}, «м <п 1b> муж», (13); (Singular-M-T); (3).
m {Распутин}, {Распутин}, «ов/ин муж», (13); (Singular-M-T); (4).

Элемент кластера, содержащий текстовую цепочку Ф.И.О. и приписанные ей варианты, назовем *разделом*.

Второй шаг. Множество строк-вариантов, входящих в каждый раздел, разобьем на подмножества по значению приписанного грамматического рода – **m** или **f** – и пометим каждую строку соответствующей буквой. Сам раздел пометим, в зависимости от наличия в нем подмножеств разного вида, буквами **m**, **f** или **mf** (в приведенном примере это уже сделано). Считаем, что у **m**-разделов **f**-подмножества не пусты, а просто отсутствуют (как и **m**-подмножества у **f**-разделов).

Часть строки, включающую основу, каноническую форму и имя класса, назовем «*классифицирующая тройка*» (далее – просто *тройка*).

По каждому подмножеству раздела построим *классифицирующее множество (К-множество)* – множество содержащихся в его строках различных троек. В соответствии с типом подмножества будем различать ***mK***- и ***fK***-*множества*.

Третий шаг. На этом шаге делается попытка построить пересечение одноименных **К**-множеств во всем кластере.

Случай А. Если в результате не получено ни одного пустого множества, то каждая строка, содержащая элемент пересечения **К**-множеств, отмечается знаком «**+**», как в нашем примере:

Валентина Распутина {mf}.
m {Распутин}, {Распутин}, «м 1-5b муж», (1); (Singular-M-R); (2).
m {Распутин}, {Распутин}, «м 1-5b муж», (1); (Singular-M-V); (2).
m {Распутина}, {Распутина}, «неизм муж», (1); (Singular-M-R); (1).
+m **{Распутин}**, **{Распутин}**, **«ов/ин муж»**, (1); (Singular-M-R); (3).
+m **{Распутин}**, **{Распутин}**, **«ов/ин муж»**, (1); (Singular-M-V); (3).
f {Распутин}, {Распутина}, «ж 1-5b жен»,(1); (Singular-F-I); (2).
f {Распутина}, {Распутина}, «неизм жен»,(1); (Singular-F-I); (1).
+f **{Распутин}**, **{Распутина}**, **«ова/ина жен»**,(1); (Singular-F-I); (3).
[…]

Случай Б. Если в результате получено хотя бы одно пустое множество и при этом в кластере встречаются как «однобуквенные», так и смешанные (**mf**-) разделы, алгоритм усложняется. Дело в том, что наличие «разнополых» вариантов в разделе может быть вызвано просто омонимичной формой имени, следовательно, одно из подмножеств раздела может быть фиктивным. Пример – формально возможные варианты с женским родом для Ф.И.О. *Евгения Гришковца*:

Евгения Гришковца {mf}.
f {Гришковц}, {Гришковца}, «ж 1-5b жен», (17); (Singular-F-I); (2).
f {Гришковц}, {Гришковца}, «ж 5a жен», (17); (Singular-F-I); (2).
f {Гришковца}, {Гришковца}, «неизм жен», (17); (Singular-F-I); (1).
[…]

Опуская подробности, суть алгоритма для этого случая можно описать следующим образом. Сначала из рассмотрения исключаются все **mf**-разделы кластера и делается попытка найти пересечение **К**-множеств исключительно по однобуквенным разделам. В случае успеха с полученным таким образом пересечением поочередно сравниваются одноименные **К**-множества из **mf**-разделов – для возможной фильтрации самого пересечения. Последнее в конечном итоге используется, как и в случае А.

На этом шаге возможны разнообразные нештатные ситуации. В таких случаях нынешняя реализация ограничивается записью в файл протокола, обеспечивая разработчикам возможность анализа различных ситуаций для развития алгоритма.

Четвертый шаг. Определение окончательных вариантов для вывода в аннотацию *Morpho*:

- если в разделе есть отмеченные плюсом строки, то все они, и только они, выводятся в результаты.
- если в разделе ни одна строка не отмечена плюсом, то в результат выводятся все строки.

Приведем в заключение результаты морфологического анализа Ф.И.О. *Евгения Гришковца* в вымышленном тестовом примере, где содержались также цепочки *Евгению Гришковец* и *Евгений Гришковец* (для сравнения – приведенные ранее результаты первого этапа для этого случая содержали 15 вариантов):

Евгения Гришковца.
{Гришков}, {ГРИШКОВЕЦ}, «м 5*a ец/ц муж», (17); (Singular-M-R); (3).
{Гришков}, {ГРИШКОВЕЦ}, «м 5*a ец/ц муж», (17); (Singular-M-V); (3).
{Гришков}, {ГРИШКОВЕЦ}, «м 5*b ец/ц муж», (17); (Singular-M-R); (3).
{Гришков}, {ГРИШКОВЕЦ}, «м 5*b ец/ц муж», (17); (Singular-M-V); (3).

Заметим, что неоднозначность окончательных результатов здесь вполне корректна:

- два возможных варианта словоизменения (с ударным и безударным окончанием), выбрать из которых один можно было бы только при наличии в тексте диагностической формы творительного падежа: *Гришковцем* или *Гришковцом*;
- омонимия родительного и винительного па-

дежей, которую в такой ситуации не смогла бы разрешить и словарная морфология.

Описание этапа 4 можно опустить, поскольку он решает чисто техническую задачу построения аннотаций *Morpho* по результатам предыдущего этапа.

В ближайшей перспективе планируется наряду с отладкой алгоритма и оценкой его работы на значительных объемах реальных текстов дополнить модуль морфологического анализа фамилий средствами анализа изолированных фамилий, возможно, с учетом контекста (этап 5). Более отдаленная задача – подключить морфологический анализ собственных имен разной природы.

Литература

1. Зализняк А.А. Грамматический словарь русского языка: Словоизменение. М.: Русский язык, 1987.

2. Сегалович И., Маслов М. Русский морфологический анализ и синтез с генерацией моделей словоизменения для незнакомых слов: тр. Междунар. сем. Казань: ООО «Хэтер», 1998. Т. 2. С. 547–552.

3. Сокирко А.В. Морфологические модули на сайте www.aot.ru: Там же. С. 559–564.

4. Ермаков А.Е., Плешко В.В. Компьютерная морфология в контексте анализа связного текста: тр. Междунар. конф. М.: Наука, 2004. С. 185–190.

5. Исследование методов извлечения информации из текстов с использованием автоматического обучения и реализация исследовательского прототипа системы извлечения информации / Куршев Е.П. [и др.]: сб. докл. 13-й Всерос. конф. М.: МАКС Пресс, 2007. С. 602–605.

ПРЕОБРАЗОВАНИЕ ОБЪЕКТНО-ОРИЕНТИРОВАННЫХ ПРОГРАММ В ИМПЕРАТИВНЫЕ МЕТОДОМ ЧАСТИЧНЫХ ВЫЧИСЛЕНИЙ

(Работа поддержана проектами РФФИ № 08-07-00280-а и № 09-01-00834-а)

Ю.А. Климов (ИПМ им. М.В. Келдыша РАН, г. Москва, yuklimov@keldysh.ru)

TRANSFORMING OBJECT-ORIENTED PROGRAMS INTO IMPERATIVE ONES BY PARTIAL EVALUATION

Klimov Yuri A. *(Keldysh Institute of Applied Mathematics RAS, Moscow, yuklimov@keldysh.ru)*

Abstract. Object-oriented languages are considered to be superior to imperative ones in their expressive power. On the other hand, programs written in imperative style tend to be more efficient as compared to ones written in object-oriented style. The paper shows that the partial evaluator CILPE can be used as a tool for transforming object-oriented programs into more efficient imperative programs.

Keywords: *specialization, partial evaluation, object-oriented programming, C#, CILPE.*

Объектно-ориентированные языки обладают большей выразительностью, чем императивные. Но в некоторых случаях эффективность программ на императивных языках программирования заметно выше, чем на объектно-ориентированных. Как показано в статье, частичный вычислитель CILPE способен преобразовывать объектно-ориентированные программы в более эффективные императивные.

Ключевые слова: *специализация, частичные вычисления, объектно-ориентированные программирование, C#, CILPE.*

Для самых разных задач часто используемые модули можно оформить в отдельные библиотеки. Повторное использование таких библиотек заметно повышает как эффективность программирования, так и последующее сопровождение исходного кода. Отдельные модули можно отлаживать и оптимизировать независимо от кода основной программы, что повышает надежность и производительность программ.

В настоящее время наиболее широко распространены объектно-ориентированные языки программирования. Поэтому библиотеки стандартных подпрограмм часто оформляются в виде классов.

В некоторых случаях использование классов может заметно сказаться на производительности программ. Например, в случае численных расчетов затраты на обработку объектов оказываются достаточно большими [1].

Для широкого использования объектно-ориентированных языков в численных расчетах необходимы средства преобразования программ, которые позволяют преобразовывать высокоуровневые объектно-ориентированные программы в низкоуровневые императивные. Для таких преобразований можно использовать метод *частичных вычислений* [2], адаптированный для объектно-ориентированных программ [3].

На основе метода частичных вычислений создан специализатор CILPE [4], способный выполнять описанные преобразования [5].

Частичные вычисления

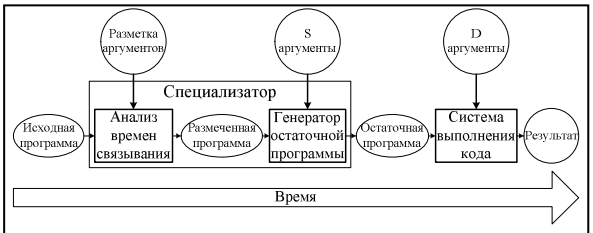
Оптимизация программ на основе использования априорной информации о значении части переменных называется *специализацией*. Рассмотрим программу **f(x, y)** от двух аргументов **x** и **y** и значение одного из ее аргументов **x=a**. Результатом специализации программы **f(x, y)** по известному аргументу **x=a** является новая программа одного аргумента **g(y)**, обладающая следующим свойством: **f(a, y)=g(y)** для любого **y**.

Одним из широко используемых методов специализации является метод ч*астичных вычислений* (*Partial Evaluation*, PE) [2]. Данный метод за-

ключается в получении более эффективного кода на основе использования априорной информации о части аргументов и однократного выполнения той части кода, которая зависит только от известной части аргументов.

В процессе частичных вычислений операции над известными данными исполняются, а над неизвестными переносятся в остаточную программу. Остаточная программа зависит только от неизвестной (на стадии специализации) части аргументов и будет исполняться только тогда, когда значения этих аргументов станут известны. Цель частичных вычислений – генерация остаточной программы.

Метод частичных вычислений основан на разделении операций и других программных конструкций на *статические* **(S)** и *динамические* **(D)**. (При этом понятие «статические операции и конструкции» не следует путать с понятием «статические методы и классы», *static*, которое используется в объектно-ориентированных языках программирования, например, C# и *Java*.) Статические операции будут выполнены во время специализации программы, а динамические перейдут в



Структура специализатора, основанного на методе частичных вычислений

остаточную программу.

Часть метода специализации, отвечающая за разделение операций и данных, называется *анализом времен связывания* (*Binding Time Analysis*, BTA, BT-анализ) (см. рис.). Вторая часть метода специализации, отвечающая за вычисление статической части программы и выделение динамической части в отдельную программу, называется *генератором остаточной программы* (*Residual Program Generator*, RPG). В этой части, собственно, и происходят *частичные вычисления*.

Метод частичных вычислений хорошо развит для функциональных языков [2]. Для объектно-ориентированных языков существуют специализаторы, основанные на методе частичных вычислений [3]. Но они, в отличие от специализатора CILPE, не решают в полной мере задачу по преобразованию объектно-ориентированных программ в императивные.

Специализатор CILPE

На основе метода частичных вычислений создан специализатор CILPE [4] для внутреннего языка *Common Intermediate Language* (CIL) платформы *Microsoft.NET*.

Язык CIL является объектно-ориентированным стековым языком. Он не предназначен для ручного написания программ. Но все языки платформы *Microsoft.NET*, такие как C#, J#, SML.NET, компилируются в язык CIL. Наличие единого внутреннего языка позволяет реализовать специализатор только для этого языка, а затем использовать его для специализации программ, написанных на различных языках программирования.

Основным языком платформы *Microsoft.NET* является высокоуровневый объектно-ориентированный язык C#. Язык CIL, хотя и не обладает всеми выразительными средствами языка C#, поддерживает все его понятия. Это позволяет обрабатывать результат компиляции программ с языка C# на язык CIL без потери необходимой для специализатора информации о программе.

Специализатор CILPE поддерживает ограниченное, но широко используемое подмножество операций языка CIL. Не поддерживаются только исключения, передача аргументов по ссылке и структуры.

Специализатор CILPE разрабатывался с учетом особенностей объектно-ориентированных языков [3]. В результате специализатор CILPE способен выполнить операции над объектами или массивами, создание и использование которых прослеживается во время специализации программы [5]. Во многих случаях в результате специализации получается программа, в которой отсутствуют объекты, то есть происходит преобразование объектно-ориентированной программы в императивную.

Используемые в специализаторе CILPE техники и идеи могут быть легко адаптированы для других объектно-ориентированных языков, например, для языка *Java Byte Code* (JBC) платформы *Java*.

Пример исходной программы

Рассмотрим пример из работы [1], демонстрирующий *задержанные* вычисления над массивами. При реализации идеи задержанных вычислений порождается много вспомогательных объектов. Специализатор CILPE на основе частичных вычислений успешно выполняет все операции над такими объектами. В результате получаем программу в виде одного цикла без объектов.

Пример также демонстрирует, что при использовании специализатора нужно развивать новые нетривиальные приемы программирования, которые позволяют сочетать удобства компонентного программирования с высокой эффективностью результирующей программы.

Рассмотрим программу на языке C#, в которой поэлементно перемножаются два массива и к результату прибавляется третий.

При программировании на императивном

языке явно указываются операции, которые необходимо выполнить с каждым элементом массива. В реальных программах циклы по элементам массива могут занимать значительную часть программы.

Пример части программы на императивном языке:

```
…
// w=x+y*z
for (int i=0; i<n; i++)
  w[i]=x[i]+y[i]*z[i];
…
```

Использование классов на объектно-ориентированным языке C# позволяет записать операции над массивами, не обращаясь явно к элементам массивов:

```
…
// w=x+y*z
w.Assign(x.Plus(y.Times(z)));
…
```

Для описания операций в таком виде используем абстрактный класс **Expr**, описывающий интерфейс доступа к элементам массива – метод **Get**. А также методы **Plus** и **Times** для поэлементного сложения и произведения массивов:

```
abstract class Expr {
  [Inline] public Expr () {}
  [Inline] public abstract double Get (int i);
  [Inline] public Expr Plus (Expr e)
  { return new BinaryOpExpr(this,e,new Plus()); }
  [Inline] public Expr Times (Expr e)
  { return new BinaryOpExpr(this,e,new Times()); }
}
```

Атрибут метода **[Inline]** в описании классов показывает специализатору CILPE, что данный метод должен быть раскрыт. То есть вместо вызова метода необходимо подставить его тело.

Для представления массива языка C# в виде наследника класса **Expr** определим производный класс **Array**. Его конструктор описывает создание массива, а метод **Get** обращение к элементам массива:

```
class Array : Expr {
  double[] data;
  [Inline] public Array (int n)
  { this.data = new double[n]; }
  [Inline] public override double Get (int i)
  { return data[i]; }
  private static int D (int n) { return n; }
  [Inline] public void Assign (Expr e) {
    for (int i = D(0); i < data.Length; i++)
      data[i] = e[i];
  }
}
```

Для присваивания значения элементам массива используется метод **Assign**. В нем описан цикл копирования значений из элементов аргумента в элементы массива. Отметим, что это единственный метод, в котором описан такого рода цикл. В дальнейшем при использовании классов **Expr** и **Array** такие циклы описывать не приходится.

Следует обратить внимание на методы **Plus** и **Times** класса **Expr**. Они предназначены для поэлементного сложения и умножения массивов. Но вместо явного вычисления, что потребовало бы заведения дополнительного массива для промежуточного результата, создается объект класса **BinaryOpExpr**, описывающий необходимые действия над элементами массива. То есть вместо того чтобы в этом методе непосредственно вычислять массив, операция над массивами запоминается и будет выполнена при обращении к элементам массива-результата.

Для этого используется класс **BinaryOpExpr**. Он хранит в себе два объекта класса **Expr** и операцию, которую необходимо произвести с элементами этих массивов, – элемент класса **BinaryOp**:

```
class BinaryOpExpr : Expr {
  Expr a, b; BinaryOp op;
  [Inline] public BinaryOpExpr (Expr a, Expr b,
    BinaryOp op) {this.a = a; this.b = b; this.op = op;}
  [Inline] public override double Get (int i)
  { return op.Apply(a.Get(i), b.Get(i)); }
}
```

Для описания операций над элементами массивов используется абстрактный класс **BinaryOp**. В данном классе определен единственный метод **Apply** – операция над двумя числами. А сами операции описываются классами **Plus** и **Times** – сложения и умножения соответственно:

```
abstract class BinaryOp {
  [Inline] public abstract double Apply (double x,
    double y);
}
class Plus : BinaryOp {
  [Inline] public abstract double Apply (double x,
    double y) { return x + y; }
}
class Times : BinaryOp {
  [Inline] public abstract double Apply (double x,
    double y) { return x * y; }
}
```

Специализация примера

Допустим, что в теле программы используются описанные классы для вычисления поэлементного произведения двух массивов и суммирование с третьим; **w**, **x**, **y**, **z** – объекты класса **Array**, объявленные по исходной программе. Часть исходной программы на языке C#:

```
…
// w=x+y*z
w.Assign(x.Plus(y.Times(z)));
…
```

Методы **Plus** и **Times** не производят вычисления, они порождают объекты, описывающие необходимые операции с элементами массивов. Затем в методе **Assign** созданные объекты используются и никуда более не передаются.

В реальных вычислительных программах таких промежуточных объектов может быть очень много. Их создание и обработка могут заметно сказаться на производительности программы. По-

этому нужен механизм, позволяющий преобразовывать эти программы в программы без создания промежуточных объектов.

Таким механизмом является специализатор CILPE, использующий метод частичных вычислений.

При специализации программы указанный ранее участок кода преобразуется специализатором CILPE в цикл на стековом языке CIL. Для читаемости приведем его эквивалент на языке C#:

```
…
// w=x+y*z
for (int i=0; i<n; i++)
  w[i]=x[i]+y[i]*z[i];
…
```

В результате специализации переменные **w**, **x**, **y**, **z** – уже не объекты класса **Array**, а массивы языка C#.

Специализатор CILPE полностью выполнил операции над объектами классов **BinaryOp**, **Plus**, **Times**, **Expr**, **Array** и **BinaryOnExpr**, и в остаточной программе от них не осталось и следа. Таким образом, в результате специализации получилась императивная программа.

Отметим, что использование объектно-ориентированного языка и специализатора, основанного на частичных вычислениях, например, языка C# и специализатора CILPE, позволяет эффективно программировать вычислительные задачи.

При использовании объектов программа получается более короткой и выразительной, что сокращает время ее разработки и упрощает отладку.

В то же время после специализации исходной программы получается программа, по эффективности сравнимая с императивной.

Для достижения уровня специализации, требуемого в таких задачах, разработанный специализатор CILPE обладает поливариантностью по коду, по переменным, по методам и по классам, обрабатывает изменяемые объекты и массивы [5].

По перечисленным свойствам специализатор CILPE превосходит известные специализаторы для объектно-ориентированных языков.

Литература

1. Jones N.D., Gomard C.K., Sestoft P. Partial Evaluation and Automatic Compiler Generation // C.A.R. Hoare Series, Prentice-Hall, 1993.

2. Todd L. Veldhuizen. Just When You Thought Your Little Language Was Safe: «Expression Templates» in Java. URL: www.springerlink.com/content/p152067k1213k606/ (дата обращения: 06.04.2009).

3. Partial Evaluation for Common Intermediate Language / A.M. Chepovsky [et al.]. URL: www.springerlink.com/content/r4rr30ajjn449q1a/ (дата обращения: 06.04.2009).

4. Климов Ю.А. Особенности применения метода частичных вычислений к специализации программ на объектно-ориентированных языках. URL: www.library.keldysh.ru/preprint.asp?id=2008-12 (дата обращения: 06.04.2009).

5. Климов Ю.А. Возможности специализатора CILPE и примеры его применения к программам на объектно-ориентированных языках. URL: www.library.keldysh.ru/preprint.asp?id=2008-30 (дата обращения: 06.04.2009).

SPSC: СУПЕРКОМПИЛЯТОР НА ЯЗЫКЕ SCALA

(Работа поддержана проектами РФФИ № 08-07-00280-а и № 09-01-00834-а)

И.Г. Ключников; С.А. Романенко, к.ф.-м.н.
(ИПМ им. М.В. Келдыша РАН, г. Москва, ilya.klyuchnikov@gmail.com)

SPSC: A SUPERCOMPILER WRITTEN IN SCALA

Klyuchnikov Ilya G.; Romanenko Sergeiy A., Ph.D.
(Keldysh Institute of Applied Mathematics RAS, Moscow, ilya.klyuchnikov@gmail.com)

Abstract. Supercompilation is a sophisticated technique of program analysis and optimization. The paper presents the complete text of a simple, yet fully operational supercompiler. Thus a working programmer is given an opportunity to become acquainted with the basic principles and ideas underlying supercompilation, formulated in his familiar language: in terms of concrete and ready-to-use programs.

Keywords: *program transformation, program analysis, program optimization, metacomputation, supercompilation, supercompiler, program specialization, Scala, functional programming, object-oriented programming.*

Суперкомпиляция является мощным и довольно сложным методом анализа и оптимизации программ. В данной статье представлен *полный* текст простого, но реально функционирующего суперкомпилятора. Это дает возможность программисту-практику ознакомиться с основными идеями и принципами суперкомпиляции, сформулированными на понятном ему языке, в виде конкретных работоспособных программ.

Ключевые слова: *преобразования программ, анализ и оптимизация программ, метавычисления, суперкомпиляция, суперкомпилятор, специализация программ, Scala, функциональное и объектно-ориентированное программирование.*

Суперкомпиляция [1–3] – метод преобразования программ, предложенный В.Ф. Турчиным в 70-х годах прошлого века. Несмотря на то, что суперкомпиляции посвящено немало литературы, практически везде алгоритмы суперкомпиляции описаны либо неполностью и фрагментарно, либо чрезмерно абстрактно (например, в виде каких-то спецификаций или правил вывода). Поэтому трудно проверить работоспособность предлагаемых методов и алгоритмов. Если информация не-

полная, читателю предлагается самому восстановить недостающие части (и нет гарантии, что он сделает это именно так, как подразумевал автор статьи). Если материал представлен слишком абстрактно, превращение его в работающие программы – сложный и неоднозначный процесс.

С точки зрения программистов-практиков, суперкомпиляция – нечто доступное только узкому кругу посвященных, поскольку ее якобы очень трудно реализовать. А без реализации нельзя оценить возможности суперкомпиляции и потенциальные области ее применения.

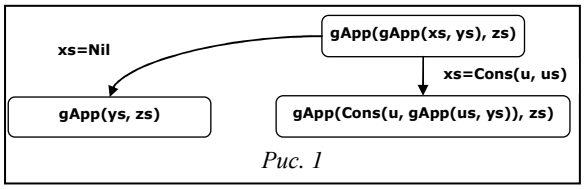


Рис. 1

Цель данной статьи – представить *полный* текст простого, но работоспособного суперкомпилятора. Это даст возможность программисту-практику ознакомиться с суперкомпиляцией, представленной на понятном ему языке – в виде конкретных программ. Причем при желании можно проверить представленные алгоритмы на деле, взяв текст суперкомпилятора прямо из статьи.

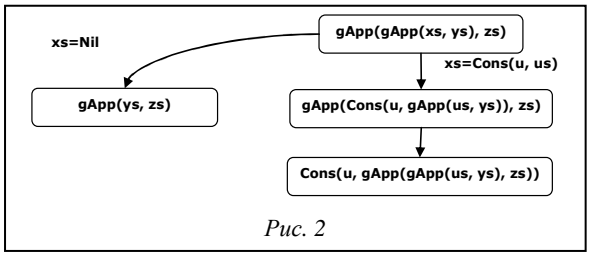


Рис. 2

В качестве языка реализации суперкомпилятора используется язык *Scala* [4]. Это объясняется тем, что для описания одних аспектов суперкомпиляции хорошо подходит функциональный стиль программирования, для описания других – объектно-ориентированный, а особенностью языка *Scala* является то, что в нем удалось органично соединить возможности как функционального, так и объектно-ориентированного программирования.

При этом *Scala* – вполне практичный язык программирования: все, что может быть написано на языке *Java*, легко может быть записано и на языке *Scala* (причем программа при этом становится короче). В настоящее время имеется реализация языка *Scala*, основанная на компиляции в виртуальный код JVM (*Java Virtual Machine*) и обеспечивающая возможность создавать программы, различные части которых написаны на *Scala* и на *Java*.

В статье рассматривается простейший суперкомпилятор SPSC для *ленивого* функционального языка первого порядка [5].

Рассмотрим программу на простом функциональном языке, конкатенирующую списки:

gApp(Nil(), vs)=vs;

gApp(Cons(u, us), vs)=Cons(u, gApp(us, vs));

Допустим, требуется получить результат конкатенации трех списков. Это можно сделать, вычислив выражение **gApp(gApp(xs, ys), zs)**. Однако при этом элементы списка **xs** будут анализироваться два раза (сначала внутренним вызовом **gApp**, а потом наружным). С помощью суперкомпиляции можно получить более эффективную программу. Интерпретируем выражение (конфигурацию) **gApp(gApp(xs, ys), zs)**. Сначала раскроем внутренний вызов **gApp(xs, ys)**. Поскольку **gApp** проверяет, какой вид имеет ее первый аргумент – **Nil()** или **Cons(u, us)**, рассмотрим два случая суперкомпиляции, то есть расщепим конфигурацию (рис. 1).

Правую конфигурацию теперь можно однозначно развернуть, заменить наружный вызов **gApp** (рис. 2).

Новое выражение содержит конструктор на верхнем уровне, с которым в данный момент ничего полезного сделать нельзя. Поэтому переходим к символическому вычислению аргументов конструктора (рис. 3).

Конфигурация в нижнем правом узле полу-

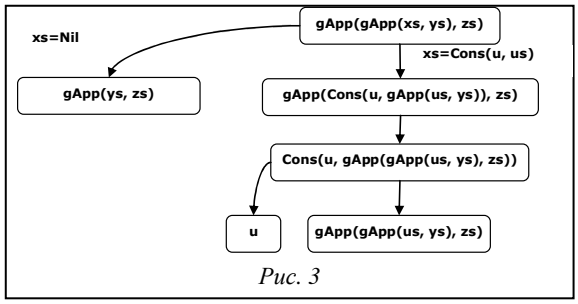


Рис. 3

чившегося дерева (рис. 3) совпадает с конфигурацией в корне дерева (с точностью до переименования переменных). Понятно, что нет смысла анализировать ее второй раз. Оставив **gApp(gApp(us, ys), zs)**, символически вычислим **gApp(ys, zs)**, получив через два шага решение, изображенное на рисунке 4.

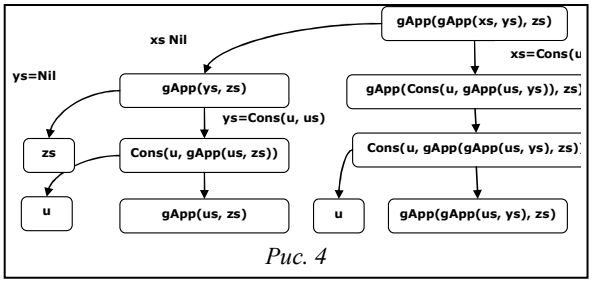


Рис. 4

Построение дерева конфигураций завершено: каждый лист дерева содержит ранее встречавшуюся конфигурацию (с точностью до переиме-

нования переменных), либо переменную, либо нуль-арный конструктор. Можно сказать, что построенное таким образом дерево представляет все возможные пути вычисления выражения, находящегося в его корне. Ветвления в дереве получаются из разбора возможных значений переменных.

Чтобы построить программу из полученного дерева конфигураций, мы, условно говоря, генерируем для каждого узла *alpha* с дочерним узлом *beta* определение функции, где левая и правая части определения выводятся из *alpha* и *beta* соответственно. В этом примере переименуем выражение **gApp(gApp(xs, ys), zs)** в **gApp1(xs, ys, zs)** и получим следующую остаточную программу:

```
gApp1(Nil(), y, z)=gApp2(y, z);
gApp1(Cons(v1, v2), y, z)=Cons(v1, gApp1(v2, y, z));

gApp2(Nil(), z)=z;
gApp2(Cons(v3, v4), z)=Cons(v3, gApp2(v4, z));
```

Видно, что при вычислении выражения **gApp1(xs, ys, zs)** элементы списка **xs** анализируются только один раз, в то время как при вычислении исходного выражения **gApp(gApp(xs, ys), zs)** они анализировались дважды.

Преобразования проходили в три этапа: символические вычисления, поиск закономерностей и построение остаточной программы, причем первые два этапа перемежались. На первом этапе была развернута конфигурация с добавлением узлов к дереву конфигураций. На втором этапе необходимо было следить за тем, чтобы конфигурации, совпадающие (с точностью до переименования переменных) с ранее обработанными конфигурациями, не развертывались далее. Это продолжалось до тех пор, пока дерево не становилось «завершенным». На последнем этапе была сгенерирована программа из завершенного дерева конфигураций.

Входной язык

Входным языком суперкомпилятора SPSC, рассмотренного в статье, является SLL – ленивый функциональный язык первого порядка. В языке SLL используется перечислимое множество символов для представления переменных **x∈X**, конструкторов **c∈C** и имен функций **f∈F** и **g∈G**. Все символы имеют фиксированную арность. SLL-программы обрабатывают данные, представляющие собой конечные деревья, построенные с помощью конструкторов. Синтаксис языка SLL состоит из набора определений функций. Причем функции делятся на два класса: **f**-функции и **g**-функции. Определение **g**-функции состоит из одного или нескольких правил, при этом в каждом правиле присутствует образец, с помощью которого анализируется структура первого аргумента. Определение **f**-функции состоит из одного правила, в котором все аргументы являются переменными.

Пример синтаксиса языка:

```
p ::= d1 .. dn                (программа)
d ::= f(x1,...,xn) = e;       (f-функция)
  | g(q1, x1,...,xn) = e1;    (g-функция)
    ...
    g(qm, x1,...,xn) = em
e ::= x                       (переменная)
  | c(e1,...,en)              (конструктор)
  | f(e1,...,en)              (вызов f-функции)
  | g(e1,...,en)              (вызов g-функции)
q ::= c(x1,...,xn)              (образец)
```

Для упрощения некоторых (несущественных) технических деталей в суперкомпиляторе потребуем, чтобы имена переменных начинались со строчной буквы, конструкторов – с прописной, а **f**- и **g**-функций – с **f** и **g** соответственно.

Семантика вычислений на языке SLL ленивая. Поэтому язык SLL является в сущности небольшим подмножеством языка *Haskell*.

В приведенной ранее программе определены два конструктора для представления списков – *Nil* и *Cons*. **G**-функция **gApp** конкатенирует два списка.

Абстрактный синтаксис

Представим реализацию абстрактного синтаксиса SLL на языке *Scala* (начало очень похоже на *Java*, но есть некоторые существенные отличия).

Сначала объявляется абстрактный класс *Term*, который используется для представления выражения языка SLL. В теле класса определены два абстрактных метода: *defs* и *args*, которые абстрактными становятся автоматически, так как у них отсутствует реализация.

```
abstract class Term{def name: String; def args: List[Term]}
case class Ctr(name: String, args: List[Term]) extends Term
case class FCall(name: String, args: List[Term]) extends Term
case class GCall(name: String, args: List[Term]) extends Term
case class Var(name: String) extends Term {val args = null}
case class Let(term: Term, bs: List[(Var, Term)]) extends Term
  {val (name, args) = (null, Nil)}
case class Pattern(name: String, args: List[Var])
case class FFun(name: String, args: List[Var], term: Term)
case class GFun(name: String, p: Pattern, args: List[Var], term: Term)
case class Program(defs: List[Either[FFun, GFun]]){
  def f(f: String) = (List.lefts(defs) find {f == _.name}).get
  def gs(g: String) = List.rights(defs) filter {g == _.name}
  def g(g: String, p: String) = (gs(g) find {p == _.p.name}).get
}
```

Далее объявляется класс **Ctr**, который изображает данные языка SLL и является подклассом класса **Term**. В языке *Scala* каждый класс обязан иметь основной (**primary**) конструктор, аргументы которого указываются сразу после имени класса, причем аргументы основного конструктора превращаются в поля класса. Конструктор **Ctr** имеет аргумент **name** (типа строка), поэтому **name** является членом класса **Ctr**. Использование ключевого слова **case** означает, что экземпляры класса можно сопоставлять с образцом. Например, если значением переменной **t** является объект класса **Ctr**, то напечатается значение поля **name** этого объекта, иначе ничего не произойдет:

```
t match {
  case Ctr(name, args) => print(name)
  case _ =>
}
```

В *Scala* значения (переменные, поля) бывают модифицируемые и немодифицируемые. Первые объявляются с помощью ключевого слова **var**, вторые – **val**. Методы объявляются с помощью ключевого слова **def**.

Класс **Ctr** имеет также поле **args** – список аргументов конструктора, представленных объектами класса **Term**. Тип списка **arg** записывается как **List[Term]**. То есть квадратные скобки в *Scala* не обозначают обращение к элементам массива, а используются для записи аргументов у конструкторов типов. (В *Java* для этого же применяются угловые скобки.) Тело класса пустое, поэтому оно (в отличие от *Java*) может быть опущено.

В определении метода **name** в классе **Term** отсутствует список параметров, в результате чего вызов метода **name** внешне выглядит как обращение к полю класса. Это позволяет в *Scala*-программах заменять реализацию свойств объекта (поле на метод или наоборот), не затрагивая части программы, использующие метод.

В подклассах **Ctr**, **FCall**, **GCall** есть поля **name** и **args**, и это приводит к тому, что автоматически появляются реализации методов **name** и **args**, объявленных в базовом классе **Term**.

В классе **Program**, помимо списка определений **defs** (определен в конструкторе), используются три метода – **f, g** и **gs**, где **f** выдает **f**-функцию по имени; **g** выдает **g**-функцию по имени и образцу; **gs** выдает список всех определений **g**-функции с данным именем.

При определении абстрактного синтаксиса языка SLL мы придерживались функционального стиля: все синтаксические объекты являются неизменяемыми значениями.

Let-выражение, не соответствующее какой-либо конструкции в языке SLL, может порождаться в процессе работы суперкомпилятора.

Операции над выражениями

Подстановкой называется набор пар вида **(v, t)**, где **v** – переменная; **t** – терм.

В терминах *Scala* для представления подстановок удобно использовать словари типа **(Map[Var, Term])**, у которых ключом является переменная, а значением – выражение.

Процесс применения подстановки к терму интуитивно понятен: находим в терме все переменные из области определения подстановки и заменяем их на соответствующие термы, что и реализовано в методе **sub**. Метод **findSub** находит (если это возможно) для двух выражений **t1** и **t2** подстановку **m**, такую что **sub(t1, m)=t2**. Идея заключается в том, что мы начинаем с пустой подстановки и помещаем в нее новые значения, обходя оба выражения одновременно методом **walk**. Метод **inst** тестирует, является ли второе выражение частным случаем первого. Метод **vars** возвращает список использованных переменных в данном терме. Здесь использованы высокоуровневые методы коллекций *Scala*: **map, /: (синтаксический сахар для foldLeft), filter, forall**, локальные частичные функции для *case*-выражений. Также удобна синтаксическая конструкция для «закарривания» функций: **sub(_, map)** эквивалентно **(x) => sub(x, map)**:

```
    object Algebra {
  def sub(term: Term, map: Map[Var, Term]): Term =
term match {
    case v: Var => map.getOrElse(v, v)
    case Ctr(n, vs)  => Ctr(n,  vs.map(sub(_, map)))
    case FCall(n, vs) => FCall(n, vs.map(sub(_, map)))
    case GCall(n, vs) => GCall(n, vs.map(sub(_, map)))
  }
  def inst(t1: Term, t2: Term) = findSub(t1, t2) != null
  def findSub(t1: Term, t2: Term) = {
    val map = scala.collection.mutable.Map[Var, Term]()
    def walk(t1: Term, t2: Term): Boolean = t1 match {
      case v: Var => map.getOrElse(v, t2) == (map+(v ->
t2))(v)
      case _ => t1.getClass == t2.getClass && t1.name ==
t2.name &&
        List.forall2(t1.args, t2.args)(walk)
    }
    if (walk(t1, t2)) Map(map.toSeq:_*).filter{case (k, v)
=> k != v} else null
  }
  def vars(t: Term): List[Var] = t match {
    case v: Var => (List(v))
    case _ => (List[Var]() /: t.args) {_ union vars(_)}
  }
}
```

Дерево процесса

Интерпретатор для языка SLL может рассматриваться как некая машина, упрощающая шаг за шагом заданный терм в соответствии с некоторыми правилами (семантикой языка). Отметим, что обычный интерпретатор способен упрощать только термы, не содержащие переменных, причем на каждом шаге редукций правило, которое необходимо применить, определяется однозначно. Вычисление заканчивается, когда текущий терм не содержит в себе вызовов функций.

В случае суперкомпиляции преобразуемые термы (конфигурации) могут содержать переменные. Из-за этого возникают ситуации, когда очередное преобразование терма нельзя выполнить однозначно и требуется разбор случаев.

Если программа написана на языке SLL, нельзя однозначно упростить вызов **g**-функции, первым аргументом которого является переменная. Требуется рассмотреть различные образцы из определения **g**-функции, расщепить текущую конфигурацию и «раскручивать» полученные конфигурации дальше. В результате символических вы-

числений (иначе – метавычислений [5]) строится дерево процесса – ориентированное дерево, в корне которого находится начальная конфигурация. В прямых потомках узла находятся конфигурации, получающиеся за один шаг метавычислений. Если произошло расщепление конфигурации, то дуге, связывающей родительский узел с дочерним, приписывается сужение – выбранный образец.

Для описания дерева процесса используются 3 класса. **Node** – узел дерева, ссылающийся на входящие и выходящие дуги графа и на содержащееся в нем выражение. **Edge** – дуга, связывающая родителя и потомка. Поле **pat** описывает сужение (выбранный образец при расщеплении конфигурации). **Tree** – дерево узлов. Класс **Tree** имеет несколько вспомогательных методов: **leaves** выдает список листьев дерева (узлов без потомков), **replace** заменяет выражение в данном узле и удаляет поддерево, **addChildren** конструирует из выражений узлы дерева и добавляет их к данному узлу:

```
    class Edge(val in: Node, var out: Node, val pat: Pat-
tern)
case class Node(expr: Term, in: Edge, var outs:
List[Edge], var fnode: Node) {
 def ancestors: List[Node] = if (in == null) Nil else in.in
:: in.in.ancestors
 def leaves: List[Node] = if (outs.isEmpty) List(this) else
children.flatMap(_.leaves)
 def children : List[Node] = outs map {_.out}
 def isProcessed: Boolean = expr match {
  case Ctr(_, Nil) => true
  case v: Var => true
  case _ => fnode != null
 }
}
class Tree(var root: Node) {
 def leaves = root.leaves
 def replace(node: Node, exp: Term) =  node.in.out =
Node(exp, node.in, Nil, null)
 def addChildren(node: Node, children: List[(Term,
Pattern)]) =
  node.outs = for ((term, p) <- children) yield {
   val edge = new Edge(node, null, p)
   edge.out = Node(term, edge, Nil, null)
   edge
  }
}
```

Суперкомпилятор

В общем случае дерево процесса, построенное в результате метавычислений, будет бесконечным. В основе суперкомпиляции лежит идея, что при построении дерева процесса будут встречаться конфигурации, похожие на ранее встречавшиеся. Например, **conf2** – это **conf1** с точностью до переименования переменных. Значит, в дальнейшем **conf2** будет развиваться так же, как и **conf1** – ее предок, и можно не строить поддерево для **conf2**, а зациклить **conf2** на **conf1**. Узел с **conf1** называется базовым узлом, с **conf2** – повторным. Может оказаться, что существует подстановка **s**, которая, будучи примененной к ранее встретившейся конфигурации **conf1**, даст текущую конфигурацию **conf2**. В этом случае говорят, что **conf2** – частный случай **conf1**, а значит, **conf2** далее будет развиваться, как **conf1** (с учетом найденной подстановки **s**): опять можно зациклить, только необходимо также построить соответствующие деревья для термов из **s** (для этого используется *let*-выражение). Таким образом, получается частичное дерево процесса.

Из такого дерева можно сгенерировать программу, которая будет эквивалентна исходной, причем некоторые «излишки» исходной программы могут быть устранены. В случае суперкомпиляции метавычисления называют прогонкой [3–5].

Опишем алгоритм построения частичного дерева процесса. Для представления зацикливания используем поле **fnode** в классе **Node**, для представления выбранного при расщеплении конфигураций образца – поле **pat** класса **Edge**.

Выражение называется тривиальным, если оно является конструктором, переменной или *let*-выражением. Узел называется обработанным, если он ссылается на функциональный узел либо на узел с переменной или нуль-арным конструктором.

Построение частичного дерева процесса начинается с помещения в корень дерева исходного выражения. Затем (пока есть хотя бы один необработанный лист **b**):

- если лист тривиальный, прогоняем выражение в этом узле;
- если среди предков **b** есть узел **a**, такой, что выражения в этих узлах совпадают с точностью до переименования, делаем узел **a** базовым узлом **b**;
- если среди предков **b** есть узел **a**, такой, что выражение в **b** является экземпляром выражения в **a**, то заменяем выражение в **b** на *let*-выражение, где **bs** – найденная подстановка, а к **b** в качестве потомков прикрепляем значения подстановок;
- в противном случае применяем прогонку.

Особенностью прогонки является прогонка вызова **g**-функции с переменной в качестве первого (образцового) аргумента. В этом случае конфигурация расщепляется соответственно образцам **g**-функции. При описании прогонки помогают сложные образцы вроде **case GCall(name, Ctr(cname, cargs) :: args)**. Кроме того, *Scala* дает изобразительные средства для использования методов (и конструкторов) в качестве бинарных операторов, например **::** является конструктором списков: **:: (head, tail)** эквивалентно **head :: tail**.

```
import Algebra._
class SuperCompiler(p: Program){
 def driveExp(expr: Term): List[(Term, Pattern)]=expr match {
   case gCall @ GCall(n, (v : Var) :: _) =>
    for (g <- p.gs(n); val pat = fPat(g.p); val ctr = Ctr(pat.name, pat.args))
      yield (driveExp(sub(gCall, Map(v -> ctr)))(0)._1, pat)
   case Ctr(name, args) => args.map((_,null))
  case FCall(n, vs)  => List((sub(p.f(n).term, Map()++p.f(n).args.zip(vs)), null))
   case GCall(name, Ctr(cname, cargs) :: vs) =>
```

```
    val g = p.g(name, cname)
    List((sub(g.term, Map((g.p.args:::g.args) zip (cargs ::: vs): _*)), null))
  case GCall(n, f :: vs) => driveExp(f) map {case (v, p) => (GCall(n, v :: vs), p)}
  case Let(term, bs) => (term, null) :: bs.map {case (_, x) => (x, null)}
 }

 def buildProcessTree(e: Term) = {
  val t = new Tree(Node(e, null, Nil, null))
  def split(a: Node, b: Node) =
   t.replace(a, Let(a.expr, findSub(a.expr, b.expr).toList))
  def step(b: Node) = if (trivial(b.expr)) t.addChildren(b, driveExp(b.expr))
    else b.ancestors.find(a => inst(a.expr, b.expr)) match {
     case Some(a) => if (inst(b.expr, a.expr)) b.fnode = a else split(b, a)
      case None => t.addChildren(b, driveExp(b.expr))
    }
  while (t.leaves.exists{!_.isProcessed}) step(t.leaves.find(!_.isProcessed).get)
  t
 }
 def trivial(t: Term)=t match{case _:FCall=>false;case _:GCall=>false;case _=>true}
 private var i = 0
 private def fPat(p: Pattern)=Pattern(p.name, p.args.map {_ => i+=1; Var("v" + i)})
}
```

Следует отметить, что при описании суперкомпилятора и частичного дерева процесса использовались как функциональные (метод **driveExp**), так и объектные средства: *Scala* позволяет смешивать различные парадигмы программирования. Написать суперкомпилятор в чисто функциональном стиле, конечно, можно, но получившийся при этом код был бы заметно более громоздким.

Генератор остаточной программы

Принцип генерации остаточной программы из частичного дерева процесса прост: каждое зацикливание порождает рекурсивную функцию, каждое расщепление конфигурации порождает функцию.

При генерации остаточной программы из дерева суперкомпиляции это дерево обходится сверху вниз методом **walk**. При посещении базового узла генерируется определение функции: если в этом узле произошло расщепление конфигурации, генерируется **g**-функция, в противном случае – **f**-функция. Определение функции состоит из сигнатуры (названия и порядка аргументов) и тела функции. Арность функции определяется исходя из количества переменных в данной конфигурации. При обходе базового узла генерируется сигнатура. При обходе узла, ссылающегося на функциональный узел, генерируется рекурсивный вызов функции, определенной в базовом узле. При обходе узла с конструктором генерируется конструктор, аргументы которого являются результатом обхода дочерних узлов. В остальных случаях итогом является результат обхода дочернего узла (транзитный шаг). Отметим, что при определении функций генерируются синтаксически корректные имена: **f**-функции начинаются с **f** и **g**-функции – с **g**. Генератор остаточной программы выдает пару – новое выражение и список определений.

```
import Algebra._
class ResidualProgramGenerator(val tree: Tree) {
 var (sigs, defs) = (Map[Node, (String, List[Var])](),List[Either[FFun, GFun]]())
 lazy val result = (walk(tree.root), Program(defs))
 private def walk(n: Node): Term = if (n.fnode == null) n.expr match {
  case v: Var => v
  case Ctr(name,args) => Ctr(name, n.children.map(walk))
  case Let(_,b) =>
   sub(walk(n.children(0)), Map()++b.map{_._1}.zip(n.children.tail map walk))
  case c: Term =>
   if (n.outs(0).pat != null) {
    sigs += (n -> ("g" + c.name.drop(1) + sigs.size, vars(c)))
    for (e <- n.outs)
     defs = Right(GFun(sigs(n)._1, e.pat, vars(c).tail, walk(e.out)))::defs
    GCall(sigs(n)._1, vars(c))
   } else if (tree.leaves.exists(_.fnode == n)) {
    sigs += (n -> ("f" + c.name.drop(1) + sigs.size, vars(c)))
    defs = Left(FFun(sigs(n)._1, sigs(n)._2, walk(n.children(0)))) :: defs
    FCall(sigs(n)._1, vars(c))
   } else walk(n.children(0))
  } else if (n.fnode.outs(0).pat == null)
    sub(Function.tupled(FCall)(sigs(n.fnode)), findSub(n.fnode.expr, n.expr))
  else sub(Function.tupled(GCall)(sigs(n.fnode)), findSub(n.fnode.expr, n.expr))
}
```

Парсер входного языка

Опишем *парсер*, который преобразует SLL-программы, представленные в *конкретном* синтаксисе (в виде последовательности литер), в SLL-программы, представленные в *абстрактном* синтаксисе (в виде деревьев).

Это позволяет продемонстрировать некоторые интересные возможности языка *Scala*.

Стандартная библиотека *Scala* включает в себя пакет для написания парсеров в комбинаторном стиле. Идея комбинаторного парсирования заключается в построении сложных парсеров из более простых путем их объединения с помощью *комбинаторов*, то есть функций, принимающих парсеры в качестве аргументов и возвращающих новые парсеры. Благодаря тому, что *Scala* позволяет использовать объектно-ориентированные средства, парсеры естественным образом реализуются в виде классов. А именно, каждый парсер является подклассом абстрактного класса:

```
abstract class Parser[+T] extends (Input => ParseResult[T]) {...}
```

При этом класс *Parser* является функцией, поскольку имеет функциональный тип **Input => ParseResult[T]**. Здесь уместно напомнить, что функции в *Scala* являются объектами, то есть могут не только применяться к аргументам, но и содержать дополнительные поля и методы.

Input – тип того, что подается парсеру на вход: в нашем случае это поток токенов (поток получается благодаря стандартному лексеру). **T** – тип того, что в итоге получается на выходе, в данном случае необходимо из текста получить объекты дерева абстрактного синтаксиса языка SLL, а именно программу. **ParseResult[T]** – абстрактный класс, подклассами которого являются успех **(Success[T])** и неудача **(Failure)**. Как было отмечено, парсер – это одновременно функция и объект с соответствующими методами. Стандартная реализация **(StandardTokenParsers)** предоставляет элементарные парсеры, разбивающие текст на строки – идентификаторы, ключевые слова, разделители и пр. Таким образом, примитивные стандартные парсеры в результате потока токенов выдают список строк, а дерево абстрактного синтак-

сиса можно построить с помощью комбинаторов – методов, получающих парсеры в качестве аргументов и выдающих парсеры. Стандартный класс **Parsers** определяет необходимые базовые комбинаторы:

- **x | y** – «или» (если **x** успешно завершается, то выдает результат работы **x**, если **y** успешно завершается, то выдает результат работы **y**, иначе выдает ошибку);
- **x ~ y** – «и затем» (выдает результат работы **x** и **y** в виде двухэлементного кортежа, если **x**, а затем и **y** успешно завершаются);
- **x+** – многократное повторение (от единицы), результат – список;
- **x ^^ f** – применение к результату работы **x** функции **f**;
- **x ^? f** – применение к результату работы **x** частичной функции **f** (успешен, если **f** определена на результате работы **x**);
- **x ~> y** – аналогичен ~, возвращает только результат работы **y**;
- **x <~ y** – аналогичен ~, возвращает только результат работы **x**;
- **repsep(x, del)** – повтор с разделителями **del(строка)**, результат – список из результатов работы **x**.

Поскольку *Scala* позволяет использовать для названий классов и методов практически любые строки, реализация парсера на скале очень близка к BNF-грамматике языка, а названия комбинаторов передают их смысл. Перейдем к модулю парсинга языка SLL. В первой строке определим разделители, далее – парсер **fid**, который соответствует идентификатору, начинающемуся с буквы **f**. Здесь воспользуемся стандартным парсером **ident**, распознающим последовательность букв и цифр, начинающуюся с буквы. С помощью комбинатора **^?** получим новый парсер. Уместно вспомнить, что последовательность *case*-выражений – это не просто функция, а *частичная* функция, определенная на перечисленных образцах. Аналогично конструируются парсеры **gid**, **uid** и **lid** (идентификатор, начинающийся с **g**, идентификатор, начинающийся со строчной буквы, и идентификатор, начинающийся с прописной буквы, соответственно). Используются *case*-выражения с ограничениями – *case*-выражение срабатывает, если срабатывает образец и выполняется записанное после него условие. Следующие парсеры почти очевидны – парсер **variable** принимает строку и выдает объект абстрактного синтаксиса – переменную. Обратим внимание, что конструктор может рассматриваться как функция соответствующей арности, то есть для унарного конструктора выражение **Ctx** эквивалентно (**x => Ctx x**).

```
import scala.util.parsing.combinator.ImplicitConversions
import scala.util.parsing.combinator.syntactical.StandardTokenParsers
import scala.util.parsing.input.{CharSequenceReader => Reader}
object SParsers extends StandardTokenParsers with ImplicitConversions {
  lexical.delimiters += ("(", ")", ",", "=", ";")
  def defs = (fFun ^^ {Left(_)} | gFun ^^ {Right(_)})+
  def term: Parser[Term] = fcall | gcall | ctr | vrb
  def uid = ident ^? {case id if id.charAt(0).isUpperCase => id}
  def lid = ident ^? {case id if id.charAt(0).isLowerCase => id}
  def fid = ident ^? {case id if id.charAt(0) == 'f' => id}
  def gid = ident ^? {case id if id.charAt(0) == 'g' => id}
  def vrb = lid ^^ Var
  def ptr = uid ~ ("(" ~> repsep(vrb, ",") <~ ")") ^^ Pattern
  def fFun = fid ~ ("(" ~> repsep(vrb, ",") <~ ")") ~ ("=" ~> term <~ ";") ^^ FFun
  def gFun=gid ("(" ~> ptr) ~ ((","  ~> vrb)*) <~ ")") ~ ("=" ~>term <~ ";")^^GFun
  def ctr = uid ~ ("(" ~> repsep(term, ",") <~ ")") ^^ Ctr
  def fcall = fid ~ ("(" ~> repsep(term, ",") <~ ")") ^^ FCall
  def gcall = gid ~ ("(" ~> repsep(term, ",") <~ ")") ^^ GCall
  def parseProgram(s: String)=Program(defs(new lexical.Scanner(new Reader(s))).get)
  def parseTerm(s: String) = term(new lexical.Scanner(new Reader(s))).get
}
```

Главные методы, которыми следует пользоваться:

- **parseProgram: String => Program** – преобразует (корректное) текстовое представление программы в саму программу;
- **parseTerm: String => Term** – преобразует (корректное) текстовое представление терма в дерево абстрактного синтаксиса.

Пример

Рассмотрим описанную программу, в которой выражение **gApp(gApp(x, y), z)** конкатенирует 3 списка, однако делает это неэффективно: получается, что список **x** анализируется на предмет соответствия образцу 2 раза. Первый раз – при вызове **gApp(x, y)**. Этот вызов создает промежуточную структуру данных – список, который затем будет проанализирован еще раз. Список **x** анализируется еще раз при обходе промежуточного списка.

Суперкомпилятор для этой программы на языке SLL вызывается следующим образом:

```
val programText =
"""
gApp(Nil(), vs1) = vs1;
gApp(Cons(u, us), vs) = Cons(u, gApp(us, vs));
"""
val p = SParsers.parseProgram(programText)
val sc = new SuperCompiler(p)
val pt = sc.buildProcessTree(SParsers.parseTerm("gApp(gApp(x, y), z)"))
val (expr, p1) = new ResidualProgramGenerator(pt).result
println(expr)
println(p1)
```

В результате на консоли выведутся новое выражение и остаточная программа, будет видно, что список **x** анализируется один раз и что остаточная программа содержит первоначальное определение конкатенации (**gApp1**) и специализированное (**gApp0**) – для конкатенации трех списков.

```
gApp0(x, y, z)
gApp0(Cons(v1, v2), y, z) = Cons(v1, gApp0(v2, y, z));
gApp0(Nil(), y, z) = gApp1(y, z);
gApp1(Cons(v3, v4), z) = Cons(v3, gApp1(v4, z));
gApp1(Nil(), z) = z;
```

В этой статье мы постарались описать простейший суперкомпилятор для ленивого функционального языка первого порядка. При этом преследовалась цель представить *полный* код суперкомпилятора.

Однако построение частичного дерева процесса не всегда завершается так, как это описано. Си-

туации, когда дерево конфигурации начинает опасно разрастаться, определяются с помощью специального критерия, называемого в суперкомпиляции *свистком*. В этом случае одна из конфигураций (которая приводит к бесконечному росту) обобщается – некоторые части конфигураций заменяются на переменные. Суперкомпиляция со свистком и обобщением завершается всегда.

Подробнее о более совершенной версии суперкомпилятора можно ознакомиться на сайте проекта в Интернете (*http://spsc.googlecode.com*).

Литература

1. Turchin V.F. The Language Refal: The Theory of Compilation and Metasystem Analysis. Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, 1980.

2. Turchin V.F. The concept of a supercompiler. ACM Transactions on Programming Languages and Systems (TOPLAS), 1986. № 8(3). pp. 292–325.

3. Абрамов С.М., Парменова Л.В. Метавычисления и их применение. Суперкомпиляция: учебник. – Переславль-Залесский: Изд-во «Университет города Переславля имени А.К. Айламазяна», 2006. 76 с.

4. Odersky M., Spoon L. and Venners B. Programming in Scala: A Comprehensive Step-by-step Guide. Artima Inc. 2008.

5. Sørensen M.H. Convergence of program transformers in the metric space of trees. In Mathematics of Program Construction. Vol. 1422 of Lecture Notes in Computer Science. Springer, 1998.

ПРОГРАММНАЯ ПОДДЕРЖКА ПОСТРОЕНИЯ ОБЛАСТИ РЕАЛИЗУЕМОСТИ ТЕРМОДИНАМИЧЕСКИХ СИСТЕМ

А.М. Цирлин, д.т.н.; И.Н. Григоревский (ИПС РАН, г. Переславль-Залесский, tsirlin@sarc.botik.ru, ivan@baby.botik.ru)

SOFTWARE SUPPORT FOR BUILDING REALIZABLE AREA OF THERMODYNAMIC SYSTEMS

Tsirlin Anatoly M., Ph.D.; Grigorevsky Ivan N.

(Program Systems Institute RAS, Pereslavl-Zalessky, tsirlin@sarc.botik.ru, ivan@baby.botik.ru)

Abstract. General features of a reliability domain of irreversible thermodynamic systems are investigated. It is shown how one can construct the domain and an example is made using Maple.

Keywords: *irreversible thermodynamics, systems, reliability domain, software, dissipation, heat engine.*

Исследованы общие особенности области реализуемости необратимых термодинамических систем, возможности ее построения и исследования качественного характера с использованием программного пакета Maple.

Ключевые слова: *необратимая термодинамика, системы, область реализуемости, программное обеспечение, диссипация, тепловая машина.*

Значительную часть технологических процессов можно охарактеризовать термодинамическими закономерностями. Тепловые и холодильные машины, процессы разделения, сушки, кристаллизации, химические реакторы и другое характеризуются такими переменными, как внутренняя энергия, концентрация тех или иных компонент, температура и энтропия.

Параметры входных и выходных потоков в таких системах связаны между собой уравнениями энергетического, материального и энтропийного балансов [1, 2]. Для открытой стационарной системы эти уравнения имеют вид:

$$\sum_j g_j h_j + \sum_i q_i - p = 0\,,\ \sum_j g_j x_{kj} + \sum_\nu \alpha_{k\nu} W_\nu = 0\ \forall k\,,$$

$$\sum_j g_j s_j + \sum_i \frac{q_i}{T_i} - \sigma = 0\,, \qquad (1)$$

где g_j – интенсивность j-го материального потока; h_j – его удельная энтальпия; x_{kj} – концентрация в нем k-го вещества; q_i – интенсивность i-го потока тепла; T_i – температура этого потока на контрольной границе системы; p – потребляемая механическая работа (мощность); W_ν – скорость ν-й химической реакции; $\alpha_{k\nu}$ – стихиометрический коэффициент, с которым k-я компонента входит в уравнение ν-й реакции ($\alpha_{k\nu} > 0$ для образующихся и $\alpha_{k\nu} < 0$ для расходующихся веществ); σ – производство энтропии (диссипация) в системе, которая зависит от потоков энергии и вещества. Коэффициенты в уравнениях (1) зависят от внешних факторов и от параметров самой системы – коэффициентов тепло- и массопереноса, интенсивных переменных (температуры, давления, концентрации и др.).

В оптимизационной термодинамике [3] решают задачу о такой организации процесса, при которой для тех или иных ограничений производство энтропии достигает своей нижней границы σ_{min}. Для любых условий $\sigma \geq \sigma_{min}$, а значит, при подстановке в уравнения (1) σ_{min} эти уравнения выделяют в пространстве потоков границу области, которую будем называть областью реализуемых значений параметров системы, или областью реализуемости. Если интенсивность стремится к нулю или коэффициенты тепло- и массопереноса сколь угодно велики, то $\sigma_{min} = 0$ и граница области реализуемости соответствует обратимым процессам. При этом ее граница всегда линейна. Если интенсивность одного или нескольких потоков за-

дана, $\sigma_{min} \geq 0$ и область реализуемости сужается. Однако может измениться и ее качественный характер, так как функция σ_{min} нелинейно зависит от интенсивности потоков.

Общие особенности границы области реализуемости

Области реализуемости термодинамических систем с ненулевой производительностью (интенсивностью целевого потока) и ограниченными размерами имеют ограниченные коэффициенты переноса. Говорить об общих особенностях можно потому, что множество реализуемости в пространстве потоков вещества и энергии определяется уравнением энтропийного баланса, в которое с постоянными коэффициентами входят потоки энтропии на входе и выходе системы, линейно зависящие от потоков вещества и энергии. Минимально возможное производство энтропии в системе $\sigma_{min}(\mathbf{g}, \mathbf{p}, \mathbf{q})$, которое является положительно-определенной, строго выпуклой вниз функцией потоков вещества и энергии, зависит от коэффициентов переноса и кинетики химических реакций, стремится к нулю при интенсивности потоков, стремящейся к нулю. Во многих случаях кинетика такова, что потоки пропорциональны движущим силам (законы теплопереноса Фурье, массоперенос, пропорциональный разнице химических потенциалов, и пр.), а в окрестности равновесия это всегда так. В таком случае производство энтропии является положительно-определенной квадратичной формой. При этом потоки связаны друг с другом уравнениями энергетического и материального балансов (см. формулу (1)).

Предположим, что число компонентов во входных и выходных потоках равно **n**. Тогда общее число уравнений (1) равно **n+2**, так как к уравнениям материального баланса добавлены балансы по энергии и энтропии. Число переменных (потоки вещества и энергии на входе и на выходе системы) обычно гораздо больше, часть из них может быть фиксированной.

Для исследования общих особенностей области реализуемости обозначим материальные и энергетические потоки через $\mathbf{x_i}$. Выделим *целевой поток*, характеризующий возможности системы (мощность тепловой машины, хладопроизводительность обратного цикла, расход разделяемой смеси и пр.), и обозначим его через $\mathbf{x_1}$, а поток затрат, который может быть одним из входящих в балансовые соотношения или линейной комбинацией $\mathbf{x_i}$, обозначим через $\mathbf{x_2}$. Минимальное производство энтропии представляет собой квадратичную функцию потоков, если они пропорциональны движущим силам, в противном случае оно может быть аппроксимировано в таком виде.

Перепишем уравнения (1) в общей, канонической форме:

$$\sum_i \mathbf{a_{ij}x_i = 0}, \quad \mathbf{j = 1, 2, ..., n+1},$$

$$\sum_i \mathbf{b_{ij}x_i - \sigma_{min}(x) \geq 0}. \qquad (2)$$

Границе области реализуемости соответствует замена в последнем соотношении знака неравенства на знак равенства. Из системы линейных уравнений для невырожденного случая можно исключить **n+1** переменных (будем исключать переменные с большими значениями индексов) и подставить в условие энтропийного баланса. Последнее для точек границы области реализуемости примет форму

$$\mathbf{c_1x_1 + c_2x_2 + ... - \sigma_{min}(x) =}$$
$$\mathbf{= c_1x_1 + c_2x_2 + ... - \sum_{ij} d_{ij}x_ix_j = 0.} \qquad (3)$$

При этом $\sigma_{min}(\mathbf{x})$ зависит только от переменных, оставшихся после исключения, но, как и ранее, представляет собой выпуклую вниз положительно-определенную квадратичную форму этих переменных.

Построим область реализуемости в плоскости с координатами $\mathbf{x_1}$ и $\mathbf{x_2}$, считая, что значения всех остальных переменных, оставшихся после исключения переменных с большими значениями индексов, фиксированы. В случае отсутствия потоков вещества этих оставшихся переменных может и не быть.

Общие свойства границы области реализуемости следующие.

• При обращении всех потоков, кроме целевого, в нуль $\mathbf{x_1}$ тоже равен нулю, так что граница области проходит через начало координат.

• Производная $\dfrac{\partial \mathbf{x_1}}{\partial \mathbf{x_2}}$ в начале координат представляет собой коэффициент эффективности в обратимом процессе. Она положительна и равна

$$\boldsymbol{\eta^0} = \left(\frac{\partial \mathbf{x_1}}{\partial \mathbf{x_2}}\right)_{x \to 0} = \left(\frac{\partial \sigma_{min}(\mathbf{x})/\partial \mathbf{x_2} - \mathbf{c_2}}{\partial \sigma_{min}(\mathbf{x})/\partial \mathbf{x_1} - \mathbf{c_1}}\right)_{x \to 0} = \frac{\mathbf{c_2}}{\mathbf{c_1}}. \qquad (4)$$

• В точке максимума целевого потока (если этот максимум существует) наклон границы области реализуемости равен нулю. Условие максимума целевого потока совместно с уравнением границы области реализуемости определяет предельное значение $\mathbf{x_1}$:

$$\partial \sigma_{min}(\mathbf{x})/\partial \mathbf{x_2} = \mathbf{2d_{22}x_2 + 2d_{12}x_1 = c_2},$$

$$\mathbf{c_1x_1 + c_2x_2 - d_{22}x_2^2 - d_{11}x_1^2 - 2d_{12}x_1x_2 = 0}. \qquad (5)$$

Если система (5) не имеет действительных корней, то область реализуемости не ограничена сверху (рис. 1а), в противном случае область реализуемости имеет иную форму (рис. 1б), при этом величина целевого потока ограничена значением $\mathbf{x_1^{max}}$, которому соответствует значение $\mathbf{x_2^0}$.

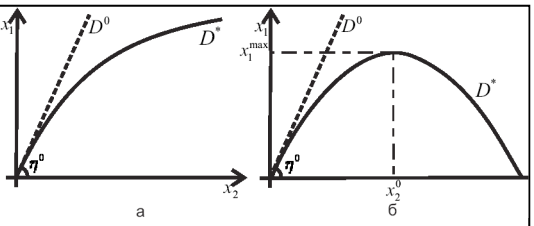


Рис. 1. Примеры области реализуемости термодинамических систем

• Область реализуемости выпукла вверх в силу свойств функции $\sigma_{min}(\mathbf{x})$.

Область реализуемости для $\sigma \geq 0$ обозначим $\mathbf{D^0}$ (ее граница соответствует обратимым процессам), а для случая $\sigma \geq \sigma_{min}$ – $\mathbf{D^* \subset D^0}$ и покажем на примере **тепловой машины** последовательность построения этих областей.

Рассмотрим тепловую машину, состоящую из горячего резервуара с температурой $\mathbf{T_+}$, холодного – с температурой $\mathbf{T_-}$ и рабочего тела, поочередно контактирующего с каждым из резервуаров и производящего механическую работу с интенсивностью **p**. Машина отбирает поток тепла $\mathbf{q_+}$ от горячего источника, отдает поток $\mathbf{q_-}$ холодному источнику и вырабатывает мощность **p**. Если процесс циклический, то под упомянутыми потоками будем понимать среднее значение потоков подводимого, отводимого тепла и механической работы (мощности) за цикл. Обмена веществом в данном случае не происходит, **n=0**.

Отобразим термодинамические балансы через исходные переменные:

$$\mathbf{p - q_+ + q_- = 0}, \quad \frac{\mathbf{q_-}}{\mathbf{T_-}} - \frac{\mathbf{q_+}}{\mathbf{T_+}} - \sigma = \mathbf{0}. \qquad (6)$$

Введем обозначения: $\mathbf{x_1 = p}$, $\mathbf{x_2 = q_+}$, $\mathbf{x_3 = q_-}$ и запишем термодинамические балансы для тепловой машины в форме (2), приняв во внимание, что σ_{min} или ее аппроксимация квадратично зависят от параметров системы:

$$\mathbf{a_1 x_1 + a_2 x_2 + a_3 x_3 = 0}, \qquad (7)$$

$$\mathbf{c_1 x_1 + c_2 x_2 + c_3 x_3 - \left[d_{11} x_1^2 + d_{22} x_2^2 + d_{33} x_3^2 + d_{12} x_1 x_2 + d_{23} x_2 x_3 + d_{13} x_1 x_3 \right] \geq 0}. \qquad (8)$$

Для определения границы области реализуемости неравенство в (8) заменим на равенство. Из формулы (7) выразим $\mathbf{x_3}$ через $\mathbf{x_1}$ и $\mathbf{x_2}$ и подставим в (8). Получим для границы области реализуемости квадратичное выражение вида $\mathbf{F(x_1, x_2) = 0}$ с двумя переменными. Функция найдена с помощью пакета *Maple* (см. уравнение (24)).

Чтобы найти точку экстремума границы области реализуемости, воспользуемся уравнением (5), которое линейно относительно $\mathbf{x_1}$ и $\mathbf{x_2}$. Из решения этого уравнения совместно с условием

$$\mathbf{F(x_1, x_2) = 0} \qquad (9)$$

определяем $\mathbf{x_1^{max}}$ и $\mathbf{x_2^0}$. Так как переменную $\mathbf{x_2}$ в силу линейности (5) легко выразить через $\mathbf{x_1}$, приходим к квадратному уравнению

$$\mathbf{\varphi(x_1) = F(x_1, x_2(x_1)) = 0}. \qquad (10)$$

Это уравнение имеет действительный корень, если его дискриминант **D** неотрицателен. Условие неотрицательности **D**, выраженное через коэффициенты системы, записанной в канонической форме, получено с использованием пакета и дано ниже, задача 4. Если это условие выполнено, область реализуемости имеет форму, как на рисунке 1б. Если дискриминант отрицателен, область реализуемости имеет вид, как на рисунке 1а. При известных коэффициентах канонической формы можно вычислить значение дискриминанта и определить вид области достижимости.

Пример расчета

Продолжим рассмотрение примера с тепловой машиной. После исключения $\mathbf{x_3}$ запишем систему в виде (2):

$$-\frac{\mathbf{x_1}}{\mathbf{T_-}} + \mathbf{x_2}\left(\frac{\mathbf{1}}{\mathbf{T_-}} - \frac{\mathbf{1}}{\mathbf{T_+}}\right) - \sigma = \mathbf{0}. \qquad (11)$$

Таким образом, коэффициенты $\mathbf{c_1} = -\frac{\mathbf{1}}{\mathbf{T_-}}$, $\mathbf{c_2} = \left(\frac{\mathbf{1}}{\mathbf{T_-}} - \frac{\mathbf{1}}{\mathbf{T_+}}\right)$.

Обратимый КПД $\eta_k = -\frac{\mathbf{c_2}}{\mathbf{c_1}} = \mathbf{1} - \frac{\mathbf{T_-}}{\mathbf{T_+}}$ (12)

равен КПД Карно и не зависит от законов теплообмена.

Для построения границы области реализуемости воспользуемся решением задачи о минимальной диссипации в цикле тепловой машины [4], приняв заданными коэффициенты теплообмена рабочего тела с источниками α_+ и α_- соответственно и закон теплопередачи в форме

$$\mathbf{x_2 = \alpha_+ (T_+ - T_1)}, \quad \mathbf{x_3 = \alpha_- (T_2 - T_-)}. \qquad (13)$$

Здесь $\mathbf{T_1}$ и $\mathbf{T_2}$ – температуры рабочего тела при контакте с источниками. Отметим, что для принятой формы закона теплообмена (его называют ньютоновским) поток непропорционален движущей силе, равной разности обратных значений температур, поэтому производство энтропии не является квадратной функцией потока и нахождение коэффициентов требует его аппроксимации квадратичной формой.

Для этих ньютоновских законов теплообмена задача о минимальной диссипации решена в [4], в которой найден предельный КПД тепловой машины заданной мощности:

$$\eta(x_1) = \frac{1}{2}\left(\frac{x_1}{\bar{\alpha} T_+} + \eta_k\right) + \sqrt{\frac{1}{4}\left(\frac{x_1}{\bar{\alpha} T_+} + \eta_k\right)^2 - \frac{x_1}{\bar{\alpha} T_+}}, \qquad (14)$$

где $\bar{\alpha}=\frac{\alpha_+\alpha_-}{\alpha_+ +\alpha_-}$.

В свою очередь, $\sigma_{min}(x_1)=\frac{x_1}{T_-}\left(\frac{\eta_k}{\eta(x_1)}-1\right)$. (15)

Ясно, что $\sigma_{min}=0$ при $x_1=0$ и $\sigma_{min}>0$ при $x_1>0$, причем при малом значении мощности зависимость $\sigma(x_1)$ квадратична.

Чтобы воспользоваться условием (5), необходимо найти значение коэффициента d_{22}. Для этого выразим σ_{min} через x_2. Так как $x_1=x_2\eta(x_1)$,

$$x_2(x_1)=\frac{x_1}{\eta(x_1)} \quad (16)$$

и σ_{min} перепишется в виде

$$\sigma_{min}(x_1,x_2)=\frac{x_1}{T_-}\left(\frac{x_2\eta_k}{x_1}-1\right)=\frac{1}{T_-}(x_2\eta_k-x_1). \quad (17)$$

Для тепловой машины поток x_2 связан с x_1 соотношением [3]:

$$x_2=\bar{\alpha}x_1\left(T_+-T_-\frac{1}{1-x_1/x_2}\right). \quad (18)$$

После преобразования получим

$$x_1=\frac{x_2^2-\bar{\alpha}T_+x_2+\bar{\alpha}T_-x_2}{\bar{\alpha}T_+-x_2}. \quad (19)$$

Подставим результат в (17) и получим $\sigma_{min}(x_2)$:

$$\sigma_{min}(x_2)=\frac{1}{T_-}\left(x_2\eta_k-\frac{x_2^2-\bar{\alpha}T_+x_2+\bar{\alpha}T_-x_2}{\bar{\alpha}T_+-x_2}\right). \quad (20)$$

Аппроксимация $\tilde{\sigma}_{min}(x_2)$ имеет вид:

$$\tilde{\sigma}_{min}(x_2)=\frac{1}{2}\left(\frac{d^2\sigma_{min}(x_2)}{dx_2^2}\right)_{x_2=0}x_2^2=d_{22}x_2^2. \quad (21)$$

С помощью *Maple* (см. ниже, задача 1) дважды продифференцировав (20) по x_2 и подставив $x_2=0$, найдем значение коэффициента d_{22}:

$$d_{22}=\frac{1}{\bar{\alpha}T_+^2}. \quad (22)$$

Так как $\sigma_{min}(x_2)$ не зависит от x_1, значения коэффициентов аппроксимации d_{11} и d_{12} равны **0**.

Уравнения (5) примут вид

$$\partial\sigma_{min}(x)/\partial x_2=2\frac{1}{\bar{\alpha}T_+^2}x_2^0=c_2,\ x_1^{max}=T_-d_{22}x_2^{0^2}; \quad (23)$$

они определяют максимальную мощность и соответствующий ей поток теплоты от горячего источника.

Программы построения области реализуемости необратимых термодинамических систем в пакете Maple

Задача 1.

#Расчет коэффициентов аппроксимации $\mathbf{sigma_{min}(x)=d_{11}*x_1^2+2*d_{12}*x_1*x_2+d_{22}*x_2^2}$

```
restart;
```

#Вид функции $\mathbf{sigma_{min}(x)}$ может быть произвольным и зависит от решаемой задачи. В данном случае мы оставляем $\mathbf{sigma_{min}(x)}$ в общем виде и проводим расчет. Тем самым даем возможность читателю подставить свое значение функции $\mathbf{sigma_{min}}$ в нашу программу и автоматически рассчитать значения коэффициентов аппроксимации для своего случая.

#Ввод конкретной функции $\mathbf{sigma_{min}}$:

```
sigma_min(x)=d11*x1^2+2*d12*x1*x2+d22*x2^2;
```

#Для нахождения коэффициента $\mathbf{d_{11}}$ необходимо дважды продифференцировать функцию $\mathbf{sigma_{min}}$ по $\mathbf{x_1}$, взять значение производной при $\mathbf{x_1=0}$ и домножить на ½:

```
koeff11:=subs(x1=0,diff(sigma_min,x1,x1))/2;
```

#Для нахождения коэффициента $\mathbf{d_{22}}$ необходимо дважды продифференцировать функцию $\mathbf{sigma_{min}}$ по $\mathbf{x_2}$, взять значение производной при $\mathbf{x_2=0}$ и домножить на ½:

```
koeff22:=subs(x2=0,diff(sigma_min,x2,x2))/2;
```

#Для нахождения коэффициента $\mathbf{d_{12}=d_{21}}$ необходимо взять смешанную производную функции $\mathbf{sigma_{min}}$ по $\mathbf{x_1}$ и $\mathbf{x_2}$ и домножить ее значение на ½ при $\mathbf{x_1=0}$ и $\mathbf{x_2=0}$:

```
koeff12:=subs(x1=0,x2=0,diff(sigma_min,x1,x2))/2;
```

Задача 2.

#Вид функции **F**, определяющей границу области реализуемости

#Приложение для выражения функции $\mathbf{F(x_1, x_2)}$ в квадратичной форме для тепловой машины

#Выпишем уравнения термодинамических балансов в общем виде для тепловой машины (7), (8)

```
tb1:=a1*x1+a2*x2+a3*x3;=0
tb2:=c1*x1+c2*x2+c3*x3-
-(d11*x1^2+d22*x2^2+d33*x3^2+d12*x1*x2+
+d23*x2*x3+d13*x1*x3)≥0;
```

#Для определения границы области реализуемости неравенство в (8) заменим на равенство

```
tb2:=c1*x1+c2*x2+c3*x3-
-(d11*x1^2+d22*x2^2+d33*x3^2+d12*x1*x2+
+d23*x2*x3+d13*x1*x3)=0;
```

#Из уравнения (7) выразим $\mathbf{x_3}$:

```
x3:=solve(tb1,x3);
```

#Подставим полученное выражение в уравнение (8): `F:=tb2;`

#Преобразуем полученное соотношение, собрав все коэффициенты при $\mathbf{x_1}$, $\mathbf{x_2}$. Это и будет искомое соотношение $\mathbf{F(x_1, x_2)=0}$ в квадратичной форме

```
F:=collect(F,x1,x2);
```

$$F:=\left(-d_{11}-\frac{d_{33}a_1^2}{a_3^2}+\frac{d_{13}a_1}{a_3}\right)x_1^2+$$
$$+\left(\left(\frac{d_{13}a_2}{a_3}-\frac{2d_{33}a_2a_1}{a_3^2}-d_{12}+\frac{d_{23}a_1}{a_3}\right)x_2+c_1-\frac{c_3a_1}{a_3}\right)x_1+$$
$$+\left(-d_{22}-\frac{d_{33}a_2^2}{a_3^2}+\frac{d_{23}a_2}{a_3}\right)x_2^2+\left(c_2-\frac{c_3a_2}{a_3}\right)x_2=0. \quad (24)$$

Задача 3.

#Построение границы области $\mathbf{x_1(x_2)}$ по условию $\mathbf{F(x_1, x_2)=0}$ для тепловой машины

#Используем выписанную функцию (24) $\mathbf{F(x_1, x_2)=0}$

#Используем значения коэффициентов a_i, c_i, d_{ij} для тепловой машины:

$a_1:=1;\ a_2:=1;\ a_3:=1;$

$$c_1 := -\frac{1}{T_{minus}};\quad c_2 := \frac{1}{T_{minus}} - \frac{1}{T_{plus}};\quad c_3 := \frac{1}{T_{minus}};$$

$$d_{11}:=0;\ d_{12}:=0;\ d_{21}:=0;\ d_{22} := \frac{1}{alpha*T_{plus}^2};\ d_{13}:=0;$$

$d_{23}:=0;\ d_{33}:=0;$

#Решаем получившееся уравнение относительно x_1 и получаем функцию для расчета границы области $x_1(x_2)$:

$$x_1 := solve(F, x_1);$$

$$x_1 := -\frac{1}{2}\frac{x_2(x_2 T_{minus} - 2alpha T_{plus}^2 + alpha T_{plus} T_{minus})}{alpha T_{plus}^2}.$$

Задача 4.

#Нахождение точки экстремума x_1^{max}

restart;

#Используем выписанную функцию(24): $F(x_1, x_2)=0$

#Выпишем первое уравнение системы (5)

$$eq := 2*d_{12}*x_1 + 2*d_{22}*x_2 = c_2;$$

#Выразим x_2 из выписанного уравнения:

$$x_2 := solve(eq, x_2);$$

#Подставим полученное соотношение в функцию $F(x_1, x_2)=0$ (24) и сгруппируем члены при x_1:

$$F := collect(F, x_1);$$

#Найдем дискриминант полученного квадратного уравнения

$$d := discrim(F, x_1);$$

$$d := \frac{1}{4a_3^2 d_{22}^2}\Big(4d_{22}c_2^2a_3^2d_{11} - 4d_{33}c_2^2a_2^2d_{11} + 4a_1^2c_2^2d_{22}d_{33} - 8a_1d_{33}c_1c_2a_2d_{22} - 4a_1d_{33}c_2^2d_{12}a_2 - 8a_1c_1a_3c_3d_{22}^2 - 4a_1d_{22}c_2^2a_3d_{13} + 4a_1d_{22}c_3c_2d_{13}a_2 + 4a_1d_{22}c_3a_3c_2d_{12} - 8a_1d_{22}c_3^2d_{12}a_2 - 4a_1^2c_2d_{23}c_3d_{22} + 4a_1d_{22}c_2d_{23}a_3c_1 - 2a_1c_2^2d_{23}d_{13}a_2 + 4a_1c_2d_{23}d_{12}c_3a_2 + 2a_1c_2^2d_{23}a_3d_{12} - 8d_{23}c_3a_3d_{11}c_2a_2 + 4d_{22}c_1a_3c_2d_{13}a_2 - 4d_{22}c_1a_3^2c_2d_{12} + 8d_{22}c_1a_3d_{12}c_3a_2 + 8d_{33}c_1c_2d_{12}a_2^2 + 2c_2^2d_{12}a_3d_{13}a_2 - 4c_3d_{13}c_2d_{12}a_2^2 - 4c_3a_3d_{12}^2a_2c_2 - 8c_2d_{23}a_3d_{12}a_2c_1 + 4c_2^2d_{23}a_3d_{11}a_2 + 4c_1^2a_3^2d_{22}^2 + 4a_1^2c_3^2d_{22}^2 + 4c_3^2d_{12}^2a_2^2 + c_2^2d_{12}^2a_3^2 + c_2^2d_{13}^2a_2^2 + a_1^2c_2^2d_{23}^2\Big)$$

#Если найденный дискриминант **d** больше или равен нулю, поток x_1 ограничен сверху и точка экстремума x_1^{max} может быть найдена.

В статье проанализированы общие свойства множества реализуемых режимов необратимых термодинамических систем. Показано, что задача построения этого множества может быть решена с использованием программных средств пакета *Maple*. В качестве примера построена область реализуемости тепловой машины.

Литература

1. Хейвуд Р. Термодинамика равновесных процессов. М.: Мир, 1983.

2. Орлов В.Н., Розоноэр Л.И. Оценки эффективности управляемых термодинамических процессов на основе уравнений баланса энергии, вещества и энтропии: матер. X Всесоюз. совещ. по проблемам управления. М.: Наука, 1986.

3. Цирлин А.М. Математические модели и оптимальные процессы в макросистемах. М.: Наука, 2006.

4. Розоноэр Л.И., Цирлин А.М. Оптимальное управление термодинамическими системами // Автоматика и телемеханика. 1983. № 1. С. 70–79; № 2. С. 88–101; № 3. С. 50–64.

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ АЛГОРИТМА ПРИБЛИЖЕННОГО РЕШЕНИЯ ЗАДАЧИ УПРАВЛЕНИЯ

(Работа поддержана РФФИ, проект № 09-01-00246-а)

Е.Ф. Сачкова (ИПС РАН, г. Переславль-Залесский, elenas@u.pereslavl.ru)

PROGRAM REALIZATION OF ALGORITHM FOR APPROXIMATE SOLVING OF CONTROL PROBLEM

Sachkova Elena F. *(Program Systems Institute of RAS, Pereslavl-Zalessky, elenas@u.pereslavl.ru)*

Abstract. A computer realization of algorithm for approximate solving the control problem for three-dimensional nonlinear systems governed by ordinary differential equations with two linear unbounded controls is considered. An iteration algorithm of solving local control problem based on method of nilpotent approximation is implemented in computer system Maple and tested for the system describing orientation control of a sphere rolling on a plane. For solving global control problem, a program complex was constructed and tested for impulsive control of a manipulator.

Keywords: *control problem, nonlinear system, control system, nilpotent approximation, algorithm, program.*

Рассматривается компьютерная реализация алгоритма приближенного решения задачи управления для трехмерных нелинейных систем, описываемых обыкновенными дифференциальными уравнениями, с двумя линейными неограниченными управлениями. Итерационный алгоритм для решения локальной задачи управления построен на основе метода нильпотентной аппроксимации, реализован в системе компьютерной математики Maple, апробирован на примере управления ориентацией катящейся по плоскости сферы. Для решения глобальной задачи управления раз-

работан комплекс программ, приведена его апробация на примере импульсного управления двухзвенным манипулятором.

Ключевые слова: *задача управления, нелинейная система, управляемая система, нильпотентная аппроксимация, алгоритм, программа.*

Настоящая работа посвящена описанию программной реализации алгоритма перемещения трехмерной нелинейной системы с двумя линейными управлениями из заданного начального состояния в малую окрестность заданного финального состояния. Написанная в системе компьютерной математики *Maple* [1] программа осуществляет последовательное приближение системы к цели с помощью управлений другой, более простой управляемой системы, являющейся аппроксимацией исходной.

Широкие возможности комплекса программ для приближенного решения локальной двухточечной граничной задачи управления обусловлены как эффективностью вычислительного алгоритма, так и эффективностью численных и символьных расчетов, осуществляемых системой *Maple*.

На базе полученного комплекса программ для приближенного решения локальной задачи управления разработан комплекс программ для приближенного решения глобальной задачи управления.

Полученный программный комплекс может быть полезен при решении прикладных задач, возникающих при управлении мобильными роботами, спутниками (при неограниченных управлениях) и иными импульсными системами [2], качением твердых тел [3].

Постановка задачи управления

Рассматриваются системы вида

$$\mathbf{dx/dt=u_1X_1(x)+u_2X_2(x), x\in R^3,}$$
$$\mathbf{u=(u_1,u_2)\in R^2,} \quad (1)$$

где $\mathbf{X_1, X_2, X_3=[X_1,X_2]=\frac{\partial X_2}{\partial x}X_1-\frac{\partial X_1}{\partial x}X_2}$ (2)

– линейно независимые гладкие векторные поля в $\mathbf{R^3}$, с заданными граничными условиями и точностью: $\mathbf{x(0)=x^0, x(T)=x^1, \varepsilon>0,}$ (3)

где $\mathbf{x^0, x^1\in R^3, T>0}$.

Ставится *задача управления*: требуется переместить систему (1), удовлетворяющую условию (2) из начального состояния $\mathbf{x^0}$ в $\boldsymbol{\varepsilon}$ – окрестность конечного состояния $\mathbf{x^1}$ за время $\mathbf{T>0}$.

Задача (1)–(3) разрешима в $\mathbf{R^3}$, так как из условия (2) следует полная управляемость системы (1) в $\mathbf{R^3}$ [3].

Алгоритм приближенного решения задачи управления (1)–(3) основан на методе нильпотентной аппроксимации системы (1), (2) в окрестности точки $\mathbf{x^1}$. В алгоритме используются две управляемые системы: исходная (1), (2) и ее нильпотентная аппроксимация в окрестности точки $\mathbf{x^1}$, которая сводится заменой переменных к симметричной нильпотентной системе

$$\mathbf{dy_1/dt=u_1, dy_2/dt=u_2,}$$
$$\mathbf{dy_3/dt=(u_2y_1-u_1y_2)/2.} \quad (4)$$

Метод приближенного решения задачи управления (1)–(3) заключается в последовательном приближении системы (1) к целевой точке $\mathbf{x^1}$ с помощью управлений системы (4), вычисляемых на каждой итерации и точно переводящих систему (4) в целевую точку. Из общей теории следует, что для любой точки $\mathbf{x^1}$ существует радиус сходимости $\boldsymbol{\delta}(\mathbf{x^1})>0$ этого алгоритма, такой, что для всех точек $\mathbf{x^0}, |\mathbf{x^0}-\mathbf{x^1}|<\boldsymbol{\delta}$, строящаяся в алгоритме последовательность приближений $\mathbf{q^n}$ сходится к $\mathbf{x^1}$ [4]. Далее будем решать задачу перемещения системы (1) из точки $\mathbf{x^0}$ в точку $\mathbf{x^1}$ при условии $|\mathbf{x^0}-\mathbf{x^1}|<\boldsymbol{\delta}$; такую задачу управления назовем *локальной*. Заметим, что граничным состояниям $\mathbf{x^0, x^1}$ исходной системы соответствуют граничные состояния $\mathbf{y^0, 0}$ соответственно системы (4).

Решения задачи управления для системы (4) с граничными условиями

$$\mathbf{y(0)=y^0, y(T)=T>0,} \quad (5)$$

найдены в пяти классах управлений: в тригонометрическом, кусочно-постоянном с одним переключением, оптимальном в смысле минимума функционала субримановой длины [5]; центральном и фокусном, построенных с помощью линейных векторных полей на плоскости, имеющих особенность, соответственно, типа центр и типа фокус. Все эти управления реализованы в виде пяти процедур программы *FindControlLoc* и составляют библиотеку управлений *NilpControls*.

С помощью локального алгоритма можно построить и *глобальный*, введя по некоторому правилу промежуточные узлы $\mathbf{x_1^1, \cdots, x_k^1}$ так, что $\mathbf{x_1^1=x^0, x_k^1=x^1}$ и $\left|\mathbf{x_{i+1}^1}-\mathbf{x_i^1}\right|<\boldsymbol{\delta}\left(\mathbf{x_i^1}\right)$.

Описание программы FindControlLoc

Локальный алгоритм приближенного решения задачи управления (1)–(3) реализован в виде компьютерной программы *FindControlLoc*, написанной на входном языке системы *Maple*. Отметим существенные характеристики программы, а именно, возможности рассматривать произвольные системы вида (1), (2), выбирать произвольные граничные условия (3) (близкие в смысле $\boldsymbol{\delta}$) и выбирать класс управлений.

Гибкость программы *FindControlLoc* обусловлена, в частности, тем, что она использует средства *Maple*-языка – языка процедурного программирования. Фрагменты алгоритма реализованы в нескольких процедурах, которые вызываются из тела программы. Это позволяет использовать один и тот же код для различных классов управлений,

экономно проводить вычисления, например, для фиксированного финального состояния вычислять нильпотентную аппроксимацию только один раз.

Процедуры программы FindControlLoc:

NilpApprox() – вычисляет нильпотентную аппроксимацию системы (1), (2) в точке $\mathbf{x} \in \mathbf{R^3}$;

ChangeCoords() – выполняет замену переменных в окрестности точки $\mathbf{x^1} \in \mathbf{R^3}$;

dsolve(), *odeplot*() – встроенные процедуры пакета расширения *Maple DEtools*.

Библиотека NilpControls.

Все процедуры библиотеки *NilpControls* вычисляют управления, являющиеся решениями задачи (4), (5) [5]:

Optimal() – вычисляет программные управления, оптимальные в смысле минимума функционала субримановой длины

$$\mathbf{L} = \int_0^{\mathbf{T}} \sqrt{\mathbf{u_1^2} + \mathbf{u_2^2}}\ \mathbf{dt} \to \mathbf{min};$$

Trig() – вычисляет программные управления в классе тригонометрических функций;

PieceConst() – вычисляет кусочно-постоянные с одним переключением программные управления, порождающие синтез;

Centre() – вычисляет программные управления, полученные с помощью линейного поля на плоскости, имеющего в нулевой точке особенность типа центр. Управления этого класса имеют одно переключение, постоянны на второй половине временного отрезка;

Focus() – вычисляет программные управления, полученные с помощью линейного поля на плоскости, имеющего в нулевой точке особенность типа фокус (рассматривается неустойчивый фокус). Управления этого класса имеют одно переключение, постоянны на второй половине временного отрезка.

Проанализируем описанную библиотеку. Управления *Optimal*, *Trig* решают задачу управления (4), (5) во всем пространстве состояний $\mathbf{R^3}$, а управления *PieceConst*, *Centre*, *Focus*, в отличие от первых двух, только в $\mathbf{R^3} \setminus \{\mathbf{y_1^2 + y_2^2 = 0}\}$, это означает, что они неприменимы в случае, если $\mathbf{y^0} \in \{\mathbf{y_1^2 + y_2^2 = 0}\}$, и нужно искать другую стратегию перемещения. Все управления библиотеки, за исключением *Trig*, порождают управления с обратной связью, что обусловливает их устойчивость к небольшим погрешностям начального положения.

Программа FindControlLoc.

Входные данные: $\mathbf{X_1},\ \mathbf{X_2} \in \mathbf{Vec(R^3)},\ \mathbf{x^0},\ \mathbf{x^1} \in \mathbf{R^3},\ \mathbf{T} > 0,\ \boldsymbol{\varepsilon} > 0,\ \mathbf{nc}$ – имя процедуры из библиотеки *NilpControls*, $par(nc)$ – числовые параметры для процедуры nc;

Выполняемые действия.

1. Инициализируются пакеты расширения системы *Maple*: *linalg, DEtools, plottools, plots*.

2. Инициализируются процедуры *NilpApprox*, *ChangeCoords*, библиотека *NilpControls*.

3. Считываются данные: $\mathbf{X_1, X_2, x^1}$.

4. Вычисляется матрица $\mathbf{C}$=***NilpApprox***$(\mathbf{X_1, X_2, x^1})$.

5. Считываются данные $\mathbf{x^0}$, $\mathbf{T}$, $\boldsymbol{\varepsilon}$; из библиотеки *NilpControls* выбирается процедура-управление nc, считываются числовые параметры $par(nc)$.

6. Входное начальное состояние $\mathbf{x^0}$ запоминается в переменной $\mathbf{q^0}$, счетчику итераций присваивается нулевое значение: $\mathbf{i} := \mathbf{0}$.

7. Итерационный процесс реализуется с помощью условного цикла $\mathbf{While(Dist(q^0, x^1)} \geq \boldsymbol{\varepsilon}\mathbf{)\ do\ \ldots}$, где $\mathbf{Dist(q^0, x^1)}$ – евклидово расстояние между точками $\mathbf{q^0, x^1}$.

i-я *итерация*: пусть $\mathbf{q^0}$ – приближение к $\mathbf{x^1}$ на $(\mathbf{i-1})$-й итерации (считаем $\mathbf{q^0 = x^0}$ приближением к $\mathbf{x^1}$ на нулевой итерации);

a) счетчик итераций увеличивается на единицу **(i:=i+1)**;

b) вычисляются координаты начального состояния системы (4): $\mathbf{y^0 = ChangeCoords(q^0, x^1)}$;

c) с помощью процедуры $\mathbf{nc} \in$**NilpControls** вычисляются управления и запоминаются в массиве переменной длины $\mathbf{u[i] := NilpControls(nc)(y^0, T, par(nc))}$ (номер ячейки массива равен значению счетчика итераций);

d) численно решается задача Коши: $\mathbf{traj = dsolve(X_1, X_2, u[i](t), q^0)}$;

e) вычисляется следующее приближение к финальной точке $\mathbf{x^1}$: $\mathbf{q^0 = traj(T)}$.

8. Если $\mathbf{Dist(q^0, x^1)} < \varepsilon$, то переменной **N** присваивается количество итераций **i** если **N=0**, то программа останавливается, если **N>0**, то переходит к следующему пункту.

9. Делается линейное преобразование переменной $\mathbf{t}$ в управлениях $\mathbf{u[i](t)}$, **i=1,…,N**.

10. Управления $\mathbf{u[i](t)}$, **i=1,…,N**, умножаются на функции $\boldsymbol{\delta}\mathbf{[i]}$ с соответствующими номерами **i**, где

$$\boldsymbol{\delta}\mathbf{[i]} = \begin{cases} \mathbf{0}, \text{ если } \mathbf{t} \in [\mathbf{0}, (\mathbf{i-1})\mathbf{T/N}], \\ \mathbf{1}, \text{ если } \mathbf{t} \in [(\mathbf{i-1})\mathbf{T/N}, \mathbf{iT/N}], \\ \mathbf{0}, \text{ если } \mathbf{t} \in [\mathbf{iT/N}, \mathbf{T}]), \end{cases}$$

полученные функции, определенные на временном отрезке $[\mathbf{0}, \mathbf{T}]$, суммируются. Получаются искомые управления $\mathbf{u(t)}$.

11. Делается проверка: управления $\mathbf{u(t)}$ подставляются в исходную систему (1), численно решается задача Коши: $\mathbf{traj = dsolve(X_1, X_2, u(t), q^0)}$, вычисляется состояние, в которое приходит система: $\mathbf{q^0 = traj(T)}$; вычисляется $\mathbf{Dist(q^0, x^1)}$.

Вывод результата.

1. Если $\mathbf{Dist(q^0, x^1)} < \varepsilon$, то управление $\mathbf{u(t)}, \mathbf{t} \in [\mathbf{0}, \mathbf{T}]$, выводится аналитически и графически. Трехмерное изображение траектории исходной системы выводится с помощью функции *odeplot*; для анализа результата выводятся ее двухмерные проекции и графики компонент.

2. Если $\mathbf{Dist(q^0, x^1)} \geq \boldsymbol{\varepsilon}$, то выводится сообщение об ошибке.

Конец программы.

Проанализируем возможные причины ошибок:

1) недостаточная точность $\boldsymbol{\varepsilon}$ в итерационном процессе;

2) наличие фазовых ограничений: траектория системы может выходить за пределы области;

3) задача не локальна для выбранного класса управлений.

Для устранения ошибок дадим следующие рекомендации. В первом случае надо повысить точность в итерациях; во втором можно поварьировать параметры **par(nc)** или выбрать другую процедуру **nc**; в третьем случае следует выбрать другую процедуру **nc** или построить глобальный алгоритм.

Программный комплекс

Процедурные свойства *Maple*-языка позволили создать библиотеку процедур приближенного решения локальной задачи управления FCL. Каждая процедура – это программа *FindControlLoc*(*nc*), **nc** – процедура библиотеки *NilpControls*. Из основной программы *GlobAlg* вызываются процедуры FCL, которые, в свою очередь, вызывают процедуры *NilpControls*, *Nilpapprox*, встроенные процедуры *Maple*. Программа *GlobAlg* может быть организована линейно, в виде цикла **For …** , в смешанном варианте. Программа *GlobAlg* универсальна. Она позволяет решать локальные и глобальные задачи управления вида (1)–(3) точно для подкласса неголономных систем – нильпотентных управляемых систем, и приближенно – для произвольных систем вида (1), (2). Решая глобальную задачу управления, можно для каждой локальной подзадачи использовать свою процедуру из библиотеки FCL, что означает возможность конструирования комбинированных алгоритмов. Эти возможности программного комплекса позволяют рассматривать глобальные задачи управления с фазовыми ограничениями. Такие задачи возникают, например, при решении задач управления для систем с трехмерными орбитами и двухмерным линейным управлением без ограничений. Комбинированные алгоритмы апробированы на примере такого рода задач – на задаче импульсного управления двухзвенным манипулятором с естественными фазовыми ограничениями.

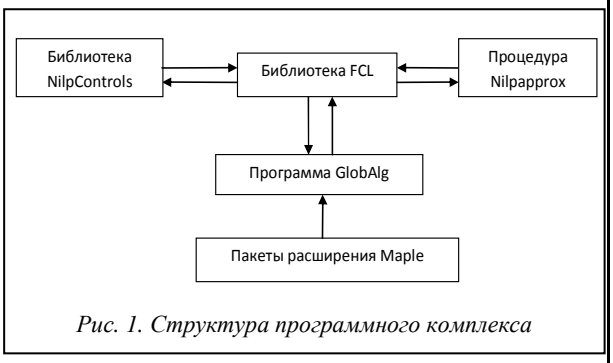


Рис. 1. Структура программного комплекса

На рисунке 1 показана структура программного комплекса для решения задачи управления (1)–(3). Основная программа *GlobAlg* преобразует входные данные $\mathbf{X_1, X_2} \in \mathbf{Vec(R^3)}$, $\mathbf{x^0, x^1} \in \mathbf{R^3}$, $\mathbf{T} > 0$, $\boldsymbol{\varepsilon} > 0$, в функции $\mathbf{u_1(t), u_2(t)} \in \mathbf{KC[0, T]}$ такие, что соответствующая траектория исходной системы выходит из начальной точки и входит в $\boldsymbol{\varepsilon}$-окрестность финальной точки. Если в задаче имеются фазовые ограничения, то программа не учитывает их автоматически и необходимо конструировать алгоритм, опираясь на свойства управлений библиотеки *NilpControls*.

Пример управления ориентацией катящейся по плоскости сферы

Управляемая система, описывающая качение сферы по плоскости без прокручивания и проскальзывания, описана в [3]. Рассмотрим подсистему этой системы, описывающую изменение ориентации сферы. Переходя в этой подсистеме от ортогональных 3×3 матриц к кватернионам и применяя проекцию на трехмерное пространство, получим следующую систему:

$$\mathbf{dx_1/dt} = -\mathbf{x_3\, u_1} + \mathbf{x_2 u_2},$$

$$\mathbf{dx_2/dt} = -\sqrt{\mathbf{1} - \mathbf{x_1^2} - \mathbf{x_2^2} - \mathbf{x_3^2}}\,\mathbf{u_1} - \mathbf{x_1 u_2},$$

$$\mathbf{dx_3/dt} = \mathbf{x_1 u_1} - \sqrt{\mathbf{1} - \mathbf{x_1^2} - \mathbf{x_2^2} - \mathbf{x_3^2}}\,\mathbf{u_2}, \quad (6)$$

$$\mathbf{q} = (\mathbf{x_1, x_2, x_3}) \in \mathbf{B^3} = \{\mathbf{x_1^2 + x_2^2 + x_3^2} < 1\},$$

$$\mathbf{u} = (\mathbf{u_1, u_2}) \in \mathbf{R^2}.$$

С помощью компьютерной программы *FindControlLoc* задача управления (1)–(3) для системы (6) решена в пяти классах управлений с использованием пяти процедур библиотеки *NilpControls*.

Пример работы программы FindControlLoc.

Входные данные: векторные поля системы (6)

$$\mathbf{X_1} = (-\mathbf{x_3}, -\sqrt{\mathbf{1} - \mathbf{x_1^2} - \mathbf{x_2^2} - \mathbf{x_3^2}}, \mathbf{x_1})^{\mathbf{T}},$$

$$\mathbf{X_2} = (\mathbf{x_2}, -\mathbf{x_1}, -\sqrt{\mathbf{1} - \mathbf{x_1^2} - \mathbf{x_2^2} - \mathbf{x_3^2}})^{\mathbf{T}},$$

граничные условия:

$\mathbf{x^0}$ =[.3555263893, .2583050417, .1359392951],

$\mathbf{x^1}$ =[.1381591491, 0., .05841275134];

время $\mathbf{T} = \mathbf{1}$; точность $\boldsymbol{\varepsilon} = \mathbf{10^{-6}}$, $\mathbf{nc} \in$ *NilpControls*, $par(nc)$.

Выполняемые действия:

$\mathbf{Dist(x^0, x^1)}$ =**.3463818366**;

Коэффициенты нильпотентной аппроксимации в точке $\mathbf{x^1}$:

$$\mathbf{C} = \begin{pmatrix} -.\mathbf{06987008492} & -\mathbf{0.5} \\ \mathbf{0.5} & -.\mathbf{0698700849} \end{pmatrix};$$

Таблица зависимости $\mathbf{N} = \mathbf{N(nc, par(nc))}$:

Процедура **nc**	Параметры **par(nc)**	Число итераций **N**
Optimal		6
Trig	1.	5
PieceConst	1.8849555922	5
Centre	(1, 4, -1)	8
Centre	(0, 5, -3)	4
Focus	(0, 5, -2, 0.3)	3

Выходные данные: управления $\mathbf{u(t)}$, $\mathbf{t \in [0, 1]}$.

Пример импульсного управления манипулятором

Рассматривается управляемая система, моделирующая импульсное управление двухзвенным манипулятором:

$$\mathbf{dx_1/dt} = -\mathbf{u_1}\,\mathbf{x_3}(\mathbf{x_3} + \mathbf{cosx_2})/\left(\mathbf{2} + \mathbf{2x_3 cosx_2} + \mathbf{x_3^2}\right) - \mathbf{u_2}\ \mathbf{sinx_2}/\left(\mathbf{2} + \mathbf{2x_3 cosx_2} + \mathbf{x_3^2}\right),$$

$$\mathbf{dx_2/dt} = \mathbf{u_1},$$

$$\mathbf{dx_3/dt} = \mathbf{u_2}, \qquad (7)$$

где $\mathbf{x} = (\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}) \in \mathbf{R^3}$, $\mathbf{u} = (\mathbf{u_1}, \mathbf{u_2}) \in \mathbf{R^2}$,

с граничными условиями

$$\mathbf{x(0)} = (\boldsymbol{\pi}/\mathbf{4}, \boldsymbol{\pi}/\mathbf{2}, \mathbf{1}),\ \mathbf{x(T)} = (\boldsymbol{\pi}/\mathbf{2}, \mathbf{0}, \mathbf{2}) \qquad (8)$$

и фазовыми ограничениями

$$-\boldsymbol{\pi}/\mathbf{2} \le \mathbf{x_1} \le \boldsymbol{\pi}/\mathbf{2},\ -\boldsymbol{\pi}/\mathbf{2} \le \mathbf{x_2} \le \boldsymbol{\pi}/\mathbf{2},\ \mathbf{0} \le \mathbf{x_3} \le \mathbf{2}. \qquad (9)$$

Для решения задачи (7)–(9) применяется следующая стратегия, реализуемая программой *GlobAlg*. Сначала выбирается промежуточная цель, и с помощью постоянных управлений система (7) с границы перемещается внутрь области (9), затем выбирается вторая промежуточная цель и решается двухточечная граничная задача управления двумя способами: с помощью центральных управлений $\mathbf{u_{centre}(t)}$ и с помощью оптимальных управлений $\mathbf{u_{opt}(t)}$. Затем с помощью постоянных управлений система (7) перемещается в $\boldsymbol{\varepsilon}$-окрестность финальной точки.

Подытоживая, следует отметить, что в статье приводится компьютерная реализация *FindControlLoc* вычислительного алгоритма приближенного решения локальной задачи управления (1)–(3). Описывается библиотека управлений *NilpControls*, благодаря которой программа *FindControlLoc* осуществляет пять стратегий управления системой в малой окрестности целевой точки. Описывается программный комплекс, основанный на базовой программе *FindControlLoc*, который позволяет решать глобальные задачи управления с фазовыми ограничениями. Приводятся примеры работы алгоритмов локального и глобального, учитывающего фазовые ограничения системы. Дальнейшее развитие программного комплекса для повышения эффективности вычислений планируется с использованием методов параллельного программирования.

Литература

1. Дьяконов В. Maple 6: учебный курс. СПб: Питер, 2001.

2. Гурман В.И. Принцип расширения в задачах оптимального управления. М.: Наука, 1997. 288 с.

3. Аграчев А.А., Сачков Ю.Л. Геометрическая теория управления. М.: Физматлит, 2005.

4. Jean. F. // Lectures on Dynamical and Contol Systems, Trieste, 2003.

5. Сачкова Е.Ф. Решение задачи управления для нильпотентной системы // Дифференциальные уравнения. 2008. № 12. С. 1704–1707.

АЛГОРИТМ РАСЧЕТА ТЕХНОЛОГИЧЕСКОЙ И ЭКОНОМИЧЕСКОЙ ЭФФЕКТИВНОСТИ ПРЕДПРИЯТИЯ

(Работа выполнена при финансовой поддержке РФФИ, грант № 08-06-00141)

С.А. Амелькин, к.т.н.

(ИПС им. А.К. Айламазяна РАН, г. Переславль-Залесский, sam@sam.botik.ru)

AN ALGORITHM OF DETERMINATION OF TECHNOLOGICAL AND ECONOMIC EFFICIENCY OF AN ENTERPRISE

Amelkin Sergey A., Ph.D. *(Program Systems Institute of RAS, Pereslavl-Zalesskij, sam@sam.botik.ru)*

Abstract. An industrial enterprise operating in an open economic system is considered. Technological equipment of the enterprise ia a heat engine. The problem at issue is to determine both thermodynamic and economic efficiencies of the enterprise. Optimality conditions for the problem are obtained, an algorithm of its solution is constructed.

Keywords: *open economic system, heat engine, thermodynamic and economic efficiencies of the enterprise.*

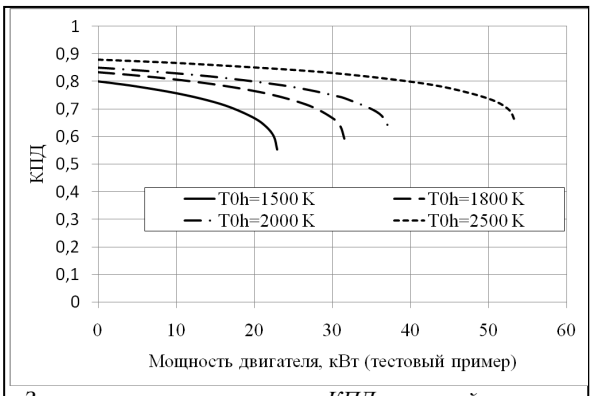
В работе рассмотривается производственное предприятие, работающее в открытой экономической системе. Технологическое оборудование предприятия представляет собой тепловую машину. Требуется определить максимальное значение термодинамической (КПД тепловой машины) и экономической (рентабельности предприятия) эффективности такого производства. Получены условия оптимальности для этой задачи.

Ключевые слова: *открытая экономическая система, тепловая машина,термодинамическая и экономическая эффективность производства.*

Максимальная прибыль производственной фирмы может быть достигнута как за счет выбора

цен (или объема выпуска товара) при наличии монополистической власти, так и за счет оптимальной организации технологического процесса. Задача оптимального выбора технологического процесса сводится к определению минимальных издержек при заданном объеме производства. Решение ее – кривая развития фирмы – дает возможность ставить задачу оптимального ценообразования, позволяющего добиться максимума прибыли или предельного значения любого другого экономического критерия.

Подробно изучен случай, когда технологическая линия представляет собой тепломеханическую систему (например, тепловую машину). Исследования в этой области получили название «термоэкономика» [1]. Кривая производственных возможностей в этом случае – это кривая зависимости максимального КПД тепловой машины от ее мощности. На этой кривой выбираются точки, соответствующие максимуму прибыли при постоянных издержках (см. рис.).



Зависимость максимального КПД тепловой машины от ее мощности

Такой подход не учитывает взаимосвязи между стоимостью ресурсов, требуемых для технологического процесса, и параметрами этого процесса: затраты на потоки теплоты связаны не с экзогенными параметрами этих потоков (температурой источника, площадью поверхности контакта рабочего тела с источником), а с интенсивностью потока, определяемого температурой рабочего тела.

Рассмотрим задачу определения кривой развития фирмы, то есть зависимость минимальных издержек фирмы от производимой мощности, и проанализируем ее свойства для случая, когда

- фирма может влиять на цены как целевого потока (мощности), так и потоков теплоты, поступающих и отводимых от рабочего тела;
- цены на ресурсы зависят не только от интенсивности потоков, но и от значений параметров уравнений теплопередачи; это означает, что цена ресурса определяется и количеством ресурса, и его качеством.

Опишем тепловую машину. Рабочее тело с распределенными параметрами контактирует с двумя источниками, имеющими постоянные температуры $\mathbf{T_{0h}}$ и $\mathbf{T_{0l}}$. Температуры рабочего тела в контакте с источниками $\mathbf{T_{0h}}$ и $\mathbf{T_{0l}}$ соответственно. Будем предполагать законы теплопередачи линейными:

$$\mathbf{q_h}\,\boldsymbol{\alpha}_h(\mathbf{T_{0h}}-\mathbf{T_h});\ \mathbf{q_l}=\boldsymbol{\alpha}_l(\mathbf{T_{0l}}-\mathbf{T_l}). \qquad (1)$$

Тепловая машина за счет теплообмена с источниками вырабатывает мощность $\mathbf{n}=\mathbf{q_h}+\mathbf{q_l}$. Предприятие, технологической линией которого является тепловая машина, покупает теплоту и продает мощность. На всех рынках оно обладает монополистической властью: цены на теплоту $\mathbf{p_h}$, $\mathbf{p_l}$ и мощность $\mathbf{p_n}$ зависят от интенсивности потоков. Цены $\mathbf{p_h}$, $\mathbf{p_l}$ зависят и от параметров источников тепла, так что $\mathbf{p_h}=\mathbf{p_h}(\mathbf{q_h},\ \mathbf{T_{0h}})$, $\mathbf{p_l}=\mathbf{p_l}(\mathbf{q_l},\ \mathbf{T_{0l}})$. Цель предприятия – получение максимальной прибыли $\boldsymbol{\pi}$ от продажи мощности. Рассмотрим стационарный режим, когда интенсивность потоков не изменяется во времени.

Алгоритм решения задачи

Задача определения оптимальных технологических параметров решается в три этапа.

На первом этапе для каждого значения $\mathbf{n}$ определяются такие величины температуры рабочего тела $\mathbf{T_{0h}}$, $\mathbf{T_{0l}}$ и потоков теплоты $\mathbf{q_h}$, $\mathbf{q_l}$, которые обеспечивают наибольший КПД тепловой машины. Для этого следует решить задачу определения минимальной диссипации:

$$\boldsymbol{\sigma}=\frac{\mathbf{q_h}(\mathbf{T_{0h}},\mathbf{T_h})}{\mathbf{T_{0h}}}+\frac{\mathbf{q_l}(\mathbf{T_{0l}},\mathbf{T_l})}{\mathbf{T_{0l}}}\rightarrow\min_{\mathbf{T_h},\mathbf{T_l}} \qquad (2)$$

при условии, вытекающем из энергетического баланса рабочего тела

$$\mathbf{q_h}(\mathbf{T_{0h}},\mathbf{T_h})+\mathbf{q_l}(\mathbf{T_{0l}},\mathbf{T_l})=\mathbf{n}, \qquad (3)$$

и условии энтропийного баланса рабочего тела

$$\frac{\mathbf{q_h}(\mathbf{T_{0h}},\mathbf{T_h})}{\mathbf{T_{0h}}}+\frac{\mathbf{q_l}(\mathbf{T_{0l}},\mathbf{T_l})}{\mathbf{T_{0l}}}=\mathbf{0}. \qquad (4)$$

На втором этапе требуется найти минимальные издержки предприятия. При заданном значении мощности $\mathbf{n}$ и известной зависимости цены готовой продукции $\mathbf{p(n)}$ доход предприятия $\mathbf{np(n)}$ также задан. Поэтому задача о минимуме издержек идентична задаче о максимальной экономической эффективности – рентабельности предприятия:

$$\mathbf{c}=\mathbf{p_h}(\mathbf{q_h},\mathbf{T_{0h}})\mathbf{q_h}(\mathbf{T_{0h}},\mathbf{T_h^*})+ \\ +\mathbf{p_l}(\mathbf{q_l},\mathbf{T_{0l}})\mathbf{q_l}(\mathbf{T_{0l}},\mathbf{T_l^*})\rightarrow\min_{\mathbf{T_{0h}},\mathbf{T_{0l}}}, \qquad (5)$$

где $\mathbf{T_h}$, $\mathbf{T_l}$ – решение задачи (2)–(4). На этом этапе не требуется учитывать условие (3), так как значения $\mathbf{T_h^*}$, $\mathbf{T_l^*}$ определены с учетом этого условия и выражения для оптимальных значений этих температур зависят от $\mathbf{n}$.

В задаче (5) учитываются только текущие издержки на приобретение теплоты. Когда требуется учесть издержки на другие факторы производства

(например, на труд, капитал в виде амортизации оборудования и пр.), выражения для этих факторов производства должны быть включены в критерий (5). Впрочем, если эти факторы производства не зависят от температур $\mathbf{T_{0h}}$, $\mathbf{T_{0l}}$, задача (5) будет сепарабельной. Например, если амортизация оборудования (определяемая капитальными вложениями) $\mathbf{c_k}$ зависит от площадей контакта рабочего тела с источниками $\mathbf{c_k=f(\alpha_h, \alpha_l)}$, то требуется наряду с задачей (5) решить задачу

$$\mathbf{f(\alpha_h,\alpha_l)\to\min_{\alpha_h,\alpha_l}} \qquad (6)$$

при условии либо сохранения общей площади теплообмена,

$$\mathbf{\alpha_h+\alpha_l=A}, \qquad (7)$$

либо сохранения эквивалентного значения коэффициента теплопередачи

$$\mathbf{\frac{\alpha_h\alpha_l}{\alpha_h\alpha_l}=\alpha_0}. \qquad (7')$$

(Примем, что коэффициенты теплопередачи пропорциональны площади контакта с источниками. Тогда ограничение на общую площадь теплообмена можно заменить на требование постоянства суммы коэффициентов теплопередачи.) Решение этих задач рассмотрено в [2].

Решая задачу минимума издержек при заданной мощности, получим решение задачи об эффективности производства – как технологической (КПД машины), так и экономической (рентабельность). В результате решения задач (2)–(4), (5) и (6)–(7') получаем зависимость $\mathbf{c(n)}$, которая может использоваться ***на третьем этапе*** для решения задачи

$$\mathbf{\pi=p(n)n-c(n)\to\max_n}. \qquad (8)$$

Решение задачи (8), стандартной для курса микроэкономики, приводит к требованию равенства максимальных издержек предельному доходу [3].

Можно решать и другие задачи, например, о максимальной мощности. При любом абсолютном критерии на этом этапе решение соответствует максимальным значениям КПД и рентабельности, а значит, режим работы предприятия относится к классу процессов минимальной диссипации и в термодинамическом, и в экономическом смыслах.

Рассмотрим задачи каждого этапа подробнее.

Задача о минимуме диссипации при заданном целевом потоке (мощности) (2)–(4) решена в [4]. Получена зависимость КПД тепловой машины от мощности и параметров линейных законов теплопередачи:

$$\mathbf{\eta=1-\frac{\alpha(T_{0h}+T_{0l})-n-\sqrt{\left[n+\alpha(T_{0h}-T_{0l})\right]^2-4\alpha T_{0h}n}}{2\alpha T_{0h}}}, \qquad (9)$$

где $\mathbf{\alpha=\alpha_h\alpha_l/(\alpha_h+\alpha_l)}$ – эквивалентный коэффициент теплопередачи.

На данном этапе решения задачи нас интересуют оптимальные зависимости потоков теплоты от источников к рабочему телу.

Получим эти зависимости. Функция Лагранжа для задачи (2)–(4) имеет вид

$$\mathbf{L=L_h+L_l+\lambda n}, \qquad (10)$$

где $\mathbf{L_i=q_i(T_{0i},T_i)\left(\frac{1}{T_{0i}}-\lambda+\frac{\mu}{T_i}\right)}$, $\mathbf{i\in\{h,l\}}$. (11)

Здесь $\mathbf{\lambda}$, $\mathbf{\mu}$ – неопределенные множители Лагранжа, соответствующие ограничениям (3)–(4). Необходимые условия оптимальности $\mathbf{\partial L/\partial T_i=0}$ $\mathbf{(i\in\{h,l\})}$ приводят к равенствам

$$\mathbf{T_i=T_{0i}\sqrt{\frac{\mu}{T_{0i}\lambda-1}}=T_{0i}(1-x_i(T_{0i}))}. \qquad (12)$$

Обозначив $\mathbf{x_i(T_{0i})=1-\sqrt{\frac{\mu}{T_{0i}\lambda-1}}}$ и подставив найденные значения $\mathbf{T_i}$ в условия (3), (4), получим систему уравнений

$$\begin{cases}\mathbf{\alpha_h T_{0h}x_h(T_{0h})+\alpha_l T_{0l}x_l(T_{0l})=n};\\ \mathbf{\alpha_h\frac{x_h(T_{0h})}{1-x_h(T_{0h})}+\alpha_l\frac{x_l(T_{0l})}{1-x_l(T_{0l})}=0}.\end{cases} \qquad (13)$$

Выражая из первого уравнения (13) $\mathbf{x_l(x_h)}$ и подставляя полученное выражение во второе уравнение (13), получаем квадратное уравнение

$$\mathbf{\alpha_h T_{0h}x_h^2-\left[n+\frac{\alpha_h\alpha_l}{\alpha_h+\alpha_l}(T_{0h}-T_{0l})\right]x_h+\frac{\alpha_l}{\alpha_h+\alpha_l}n=0}.$$

Используя выражение для эквивалентного коэффициента теплопередачи $\mathbf{\alpha=\alpha_h\alpha_l/(\alpha_h+\alpha_l)}$, находим значения $\mathbf{x_h}$ и $\mathbf{x_l}$, подставляем их в (12), получаем выражения для $\mathbf{T_h^*}$ и $\mathbf{T_l^*}$. Найденные значения подставляем в уравнения кинетики (1) и находим искомые выражения для теплоты:

$$\begin{aligned}\mathbf{q_h}&=\mathbf{\frac{1}{2}\left[n+\alpha(T_{0h}-T_{0l})-\sqrt{D}\right]},\\ \mathbf{q_l}&=\mathbf{\frac{1}{2}\left[n-\alpha(T_{0h}-T_{0l})+\sqrt{D}\right]},\end{aligned} \qquad (14)$$

где $\mathbf{D=\left[n+\alpha(T_{0h}-T_{0l})\right]^2-4\alpha T_{0h}n}$. (15)

Задача о минимуме издержек. Зная зависимость потоков теплоты (14), (15) от значений $\mathbf{T_{0h}}$, $\mathbf{T_{0l}}$, можно выписать условия оптимальности для задачи (5):

$$\begin{cases}\mathbf{\frac{\partial c}{\partial T_{0h}}=0\Rightarrow\left(\frac{\partial p_h}{\partial q_h}q_h+p_h\right)\frac{\partial q_h}{\partial T_{0h}}+}\\ \qquad\mathbf{+\left(\frac{\partial p_l}{\partial q_l}q_l+p_l\right)\frac{\partial q_l}{\partial T_{0h}}+\frac{\partial p_h}{\partial T_{0h}}q_h=0},\\ \mathbf{\frac{\partial c}{\partial T_{0l}}=0\Rightarrow\left(\frac{\partial p_h}{\partial q_h}q_h+p_h\right)\frac{\partial q_h}{\partial T_{0l}}+}\\ \qquad\mathbf{+\left(\frac{\partial p_l}{\partial q_l}q_l+p_l\right)\frac{\partial q_l}{\partial T_{0l}}+\frac{\partial p_l}{\partial T_{0l}}q_l=0}.\end{cases} \qquad (16)$$

Рассмотрим значения производных $\mathbf{\partial q_i/\partial T_{0j}=0}$

(i, j∈{h, *l*}). Для этого представим выражения для потоков теплоты в виде

$$\mathbf{q_h=\frac{1}{2}(n+\kappa)};\ \mathbf{q_l=\frac{1}{2}(n-\kappa)}\ , \tag{17}$$

где $\boldsymbol{\kappa=\alpha(T_{0h}-T_{0l})-\sqrt{D}}$. Только $\boldsymbol{\kappa}$ зависит от значений $\mathbf{T_{0h}}$, $\mathbf{T_{0l}}$. Поэтому

$$\mathbf{\frac{\partial q_h}{\partial T_{0j}}=-\frac{\partial q_l}{\partial T_{0j}},\ j\in\{h,l\}}\ , \tag{18}$$

а значит, уравнения (16) могут быть сведены к равенству

$$\mathbf{\frac{\frac{\partial p_h}{\partial T_{0h}}q_h}{\frac{\partial p_h}{\partial T_{0h}}}=-\frac{\frac{\partial p_l}{\partial T_{0l}}q_l}{\frac{\partial p_l}{\partial T_{0l}}}}\ . \tag{19}$$

Надо учесть, что $\mathbf{q_h>0}$, $\mathbf{q_l<0}$, $\mathbf{\partial q_h/\partial T_{0h}<0}$, $\mathbf{\partial q_l/\partial T_{0l}<0}$. Последние два неравенства требуют пояснений, так как согласно (1) обе эти производные должны быть положительными. Однако мы рассчитываем эти производные с учетом решения задачи (2)–(4). В этом случае $\mathbf{q_h=q_h(T_{0h},\ T_{0l})}$, $\mathbf{q_h=q_h(T_{0h},\ T_{0l})}$, которые определяются формулами (14), (15). При увеличении температуры $\mathbf{T_{0h}}$ требуется меньший поток теплоты, чтобы обеспечить заданную мощность **n**; при увеличении температуры $\mathbf{T_{0l}}$ требуется большее значение интенсивности отвода теплоты от рабочего тела, а значит, меньшее значение $\mathbf{q_{0l}}$.

Из равенства (14) примем, что

$$\mathbf{sign\left(\frac{\partial p_h}{\partial T_{0h}}\right)=sign\left(\frac{\partial p_l}{\partial T_{0l}}\right)}\ .$$

Это также становится понятным, если учесть, что при $\mathbf{q_l<0}$ требуется, чтобы $\mathbf{p_l}$ было также отрицательным. Производные $\mathbf{\partial q_h/\partial T_{0h}}$, $\mathbf{\partial q_l/\partial T_{0l}}$ связаны с интенсивностью потоков $\mathbf{q_h}$ и $\mathbf{q_l}$ следующим образом:

$$\begin{aligned}\mathbf{\frac{\partial q_h}{\partial T_{0h}}}&=\mathbf{\frac{1}{2}\alpha-\frac{1}{4\sqrt{D}}\left[2\alpha(n+\alpha(T_{0h}-T_{0l}))-4\alpha n\right]=}\\&=\mathbf{\frac{\alpha}{2}\left[1+\frac{n-\alpha(T_{0h}-T_{0l})}{\sqrt{D}}\right]=\frac{\alpha q_l}{\sqrt{D}}},\\ \mathbf{\frac{\partial q_l}{\partial T_{0l}}}&=\mathbf{\frac{1}{2}\alpha-\frac{1}{4\sqrt{D}}\left[-2\alpha(n+\alpha(T_{0h}-T_{0l}))\right]=}\\&=\mathbf{\frac{\alpha}{2}\left[1-\frac{n+\alpha(T_{0h}-T_{0l})}{\sqrt{D}}\right]=-\frac{\alpha q_h}{\sqrt{D}}}.\end{aligned} \tag{20}$$

С учетом (20) равенство (19) примет вид

$$\mathbf{\left(\frac{q_l}{q_h}\right)^2=\frac{\partial p_l/\partial T_{0l}}{\partial p_h/\partial T_{0h}}}\ , \tag{21}$$

откуда с учетом отрицательности величины $\mathbf{q_l}$:

$$\begin{aligned}&\mathbf{\frac{q_l}{q_h}=-\sqrt{\frac{\partial p_l/\partial T_{0l}}{\partial p_h/\partial T_{0h}}}}\ \text{или}\\&\mathbf{\eta=1+\frac{q_l}{q_h}=1-\sqrt{\frac{\partial p_l/\partial T_{0l}}{\partial p_h/\partial T_{0h}}}}.\end{aligned} \tag{22}$$

Уравнение (22) является условием оптимальности для выбора таких значений $\mathbf{T_{0h}}$, $\mathbf{T_{0l}}$, чтобы экономическая эффективность предприятия была бы наибольшей, а значит, диссипация капитала – наименьшей [2].

Интересно, что для выбора функций спроса и предложения (кинетики ресурсообмена в экономической системе), определяющих зависимость цен от интенсивностей потоков и величин температур источников $\mathbf{T_{0j}}$ **(j∈{h, *l*})**, удобно воспользоваться следующей зависимостью:

$$\mathbf{p_j=v_j-\frac{p_j q_j(T_{0h},T_{0l})}{T_{0j}},\ j\in\{h,l\}} \tag{23}$$

Это связывает цену с потоком энтропии, отбираемым от **j**-го источника.

К сожалению, расчет значений $\mathbf{T_{0h}}$, $\mathbf{T_{0l}}$ для зависимостей (23), удовлетворяющих условиям (22), а значит, и решение задач третьего этапа, аналитически невозможны – требуется численный расчет для конкретных значений параметров кинетики ресурсообмена $\boldsymbol{\alpha}$, $\boldsymbol{\beta_h}$, $\boldsymbol{\beta_l}$, $\mathbf{v_h}$, $\mathbf{v_l}$, а также заданной функции спроса $\mathbf{p(n)}$.

Литература

1. Sieniutycz S., Salamon P. (eds.). Finite-Time Thermodynamics and Thermoeconomics. Taylor & Francis, 1990.

2. Миронова В.А., Амелькин С.А., Цирлин А.М. Математические методы термодинамики при конечном времени. М.: Химия, 2000.

3. Пиндайк Р., Рубинфельд Д. Микроэкономика. М.: Экономика. Дело, 1992.

4. Цирлин А.М. Оптимальные циклы и циклические режимы. М.: Энергоатомиздат, 1985.

АЛГОРИТМ РАСЧЕТА ПРЕДЕЛЬНЫХ ВОЗМОЖНОСТЕЙ СТАЦИОНАРНЫХ ТЕПЛОМЕХАНИЧЕСКИХ СИСТЕМ

А.А. Ахременков, к.т.н. (ИПС РАН, г. Переславль-Залесский, andrei@eco.botik.ru)

COMPUTATION ALGORITHM FOR LIMIT POSSIBILITIES OF THERMODYNAMIC

STATIONARY SYSTEMS

Akhremenkov Andreiy A., Ph.D.

(Program Systems Institute of RAS, SARC, Pereslavl-Zalessky, andrei@eco.botik.ru)

Abstract. In this paper consider computation problem of maximum power production in thermodynamic system with heat pump.

Keywords: *finite time thermodynamic, heat pump, thermodynamic system.*

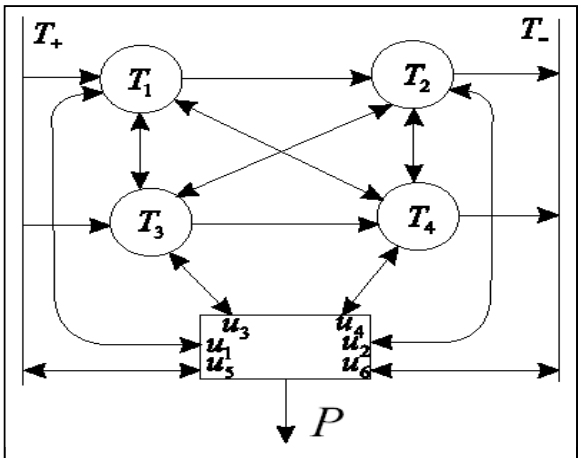
В работе рассмотрена задача расчета максимальной извлекаемой мощности для термической системы произвольной конфигурации, содержащей тепловую машину.

Ключевые слова*: оптимизационная термодинамика, тепломеханический преобразователь, тепломеханические системы.*

Термодинамика при конечном времени получила первый толчок для своего развития в задаче о цикле тепловой машины с максимальной мощностью [1, 2]. При этом в большинстве работ рассматривалась система, состоящая из тепловой машины и нескольких резервуаров с заданными и постоянными температурами [3].

Между тем эта задача особенно актуальна для систем, состоящих из термических резервуаров и подсистем конечной емкости с различающимися температурами, находящихся в контакте с резервуарами и друг с другом (см. рис.).

В такой системе с отличающимися друг от друга температурами резервуаров при заданных законах и коэффициентах теплопереноса устанавливается стационарный режим, характеризующийся распределением температур между подсистемами и дискретным температурным полем. При этом каждая подсистема (резервуар, подсистемы конечной емкости и тепловая машина) считается внутренне равновесной, так что необратимые эффекты возникают на границах подсистем. Только при этих допущениях для каждой подсистемы справедливо термодинамическое описание. Такие системы в литературе называют *endoreversible*-системами.



Открытая термодинамическая система

Естественно возникает вопрос, какую максимальную мощность можно извлечь в стационарной термодинамической системе с использованием тепловой машины, имеющей заданные коэффициенты теплообмена при контакте с каждым элементом системы. Назовем это задачей о максимальной мощности. В такой постановке она обобщает задачу И.И. Новикова [1].

Математическое описание

Рассмотрим термодинамическую систему (структура ее представлена на рисунке), состоящую из **(n-m)** резервуаров с постоянными температурами, **m** подсистем конечной емкости, температуры которых определяются запасом их внутренней энергии. Каждая подсистема может контактировать с любой подсистемой и резервуаром, к ней могут подводиться конвективные потоки тепла извне $\mathbf{q_i^K}$. Потоки обмена теплом подсистем друг с другом обозначим как $\mathbf{q_{ij}(T_i, T_j)}$. Эти потоки связаны с различием температур подсистем. Будем считать, что поток положителен $\mathbf{q_{ij}(T_i, T_j)>0}$, если тепло поступает к **i**-й подсистеме, то есть $\mathbf{T_i<T_j}$. Функции $\mathbf{q_{ij}(T_i, T_j)}$ обладают следующими свойствами:

$\mathbf{q_{ij}(T_i, T_j)}$ – непрерывная и непрерывно дифференцируемая по совокупности аргументов;

$$\mathbf{q_{ij}(T_i, T_j)=-\,q_{ji}(T_j, T_i);\ q_{ij}(T_i, T_j)\equiv 0 \Leftarrow T_i = T_j;}$$

$$\frac{\partial q_{ij}}{\partial T_i}<0, \frac{\partial q_{ij}}{\partial T_j}>0;\ \beta_{ij}=\frac{\partial q_{ij}}{\partial T_i}=\frac{\partial q_{ij}}{\partial T_j}=\beta_{ji}.$$

Последнее свойство характерно для систем теплообмена.

Температура подсистем зависит от запаса внутренней энергии и изменяется в результате подвода (отвода) тепла к ней:

$$\dot{T}_i=\frac{q_i}{C_i}, \qquad (1)$$

где $\mathbf{q_i}$ – суммарный поток тепла, подводимый к подсистеме; $\mathbf{C_i}$ – ее теплоемкость.

Рассмотрим стационарное состояние тепломеханической системы, содержащей тепловую машину, которая может контактировать с подсистемами, получая или отдавая им потоки тепла, потребляя или вырабатывая мощность. Требуется найти такие температуры контакта $\mathbf{u_i}$ рабочего тела тепловой машины с каждой из подсистем, при которых получаемая в единицу времени работа (мощность **P**) максимальна. Если максимальная мощность отрицательна, то она соответствует минимуму затрат подводимой извне энергии (в этом случае тепломеханическим преобразователем является тепловой насос).

Постановка задачи и условия оптимальности

Обозначим через $\mathbf{u_i}$ температуру рабочего тела при контакте с **i**-й подсистемой; $\mathbf{q_{ij}(T_i, T_j)}$ – поток тепла между **i**-й подсистемой и преобразователем; **P** – мощность преобразователя. Поток, поступающий в каждую из подсистем, будем считать положительным.

Формализуем задачу о максимальной мощности:

$$\mathbf{P=\sum_{i=1}^{n} q_i(T_i,u_i)\to\max_{u_i>0}} \qquad (2)$$

при условиях

$$\mathbf{P=\sum_{i=1}^{n}\frac{q_i(T_i,u_i)}{u_i}=0}\ ; \qquad (3)$$

$$\mathbf{\sum_{j=1}^{n} q_{ij}(T_i,T_j)=q_i(T_i,u_i),\ i=1,...,m.} \qquad (4)$$

Критерий (2) следует из энергетического баланса рабочего тела преобразователя. Условие (3) вытекает из энтропийного баланса рабочего тела, а (4) – энергетический баланс для **i**-й подсистемы конечной емкости.

Предположим, что потоки $\mathbf{q_i}$, $\mathbf{q_{ij}}$ линейно зависят от разности температур

$$\mathbf{q_i=\alpha_i(T_i-u_i),\ q_{ij}=\alpha_{ij}(T_j-T_i),} \qquad (5)$$

где $\boldsymbol{\alpha_i}, \boldsymbol{\alpha_{ij}}$ – коэффициенты теплопереноса. Такой закон теплообмена называют ньютоновским.

Очевидно, что при числе резервуаров, большем или равном двум, и различающихся температурах резервуаров максимальная мощность положительна.

Разобьем задачу на три подзадачи.

1. Максимизировать извлекаемую мощность $\mathbf{P_r^*(\sigma_r)}$ при контакте преобразователя с резервуарами для заданного значения $\boldsymbol{\sigma_r}$ – потока энтропии от резервуаров к рабочему телу

$$\mathbf{P_r(\sigma_r)=\sum_{i=m+1}^{n} q_i\to\max_{q_i}} \qquad (6)$$

при условии

$$\mathbf{\sum_{i=m+1}^{n}\frac{q_i}{u_i}=\sigma_r}\ . \qquad (7)$$

2. Максимизировать извлекаемую мощность $\mathbf{P_s^*(\sigma_s)}$ при контакте преобразователя с подсистемами конечной емкости для заданного значения $\boldsymbol{\sigma_s}$ – потока энтропии от подсистем к рабочему телу

$$\mathbf{P_s(\sigma_s)=\sum_{i=1}^{m} q_i\to\max_{q_i}} \qquad (8)$$

при условии

$$\mathbf{\sum_{i=1}^{m}\frac{q_i}{u_i}=\sigma_s} \qquad (9)$$

и балансовых соотношениях (4).

3. Найти максимальную суммарную мощность при условии баланса по энтропии для рабочего тела преобразователя, производство энтропии в котором равно нулю:

$$\mathbf{P(q_i)=P_s^*(\sigma_s)+P_r^*(\sigma_r)\to\max/\sigma_r+\sigma_r=0}\ . \qquad (10)$$

Первая из этих задач рассмотрена в работе [4] для случая $\boldsymbol{\sigma_r=0}$, где показано, что для ньютоновского теплообмена значение приведенной температуры контакта $\boldsymbol{\Lambda}$ определяется как

$$\mathbf{\sqrt{\Lambda}=\frac{\sum_{i=m+1}^{n}\alpha_i\sqrt{T_i}}{\sigma_r+\alpha_\Sigma^r},\ \alpha_\Sigma^r=\sum_{i=m+1}^{n}\alpha_i}\ . \qquad (11)$$

Максимальная мощность, извлекаемая при контакте с резервуарами:

$$\mathbf{P_r^*(\sigma_r)=\sum_{i=m+1}^{n}\alpha_i\sqrt{T_i}\left(\sqrt{T_i}-\frac{\sum_{j=m+1}^{n}\alpha_i\sqrt{T_i}}{\sigma_r+\alpha_\Sigma^r}\right)}\ ; \qquad (12)$$

с ростом $\boldsymbol{\sigma_r}$ эта мощность монотонно возрастает.

Во второй задаче требуется найти не только температуры контактов $\mathbf{u_i}$, но и температуры подсистем $\mathbf{T_i}$ для **i=1, …, m**. Выразим $\mathbf{u_i}$ через $\mathbf{q_i}$ и $\mathbf{T_i}$ при условии, что потоки тепла заданы в ньютоновской форме (5):

$$\mathbf{u_i=T_i-\frac{q_i}{\alpha_i}}\ , \qquad (13)$$

и перепишем балансовые соотношения (4) как систему линейных уравнений относительно температур подсистем $\mathbf{T_i}$, **i=1, …, m**:

$$\mathbf{\sum_{j=1}^{m}\alpha_{ij}T_j-T_i\sum_{j=1}^{n}\alpha_{ij}=q_i-\sum_{j=m+1}^{n}\alpha_{ij}T_j,\ i=1,...,m}\ . \qquad (14)$$

Или в матричной форме:

$$\mathbf{A(\alpha)T=C(q)}\ ,$$

$$\text{где}\ \mathbf{A(\alpha)}=\begin{Vmatrix}\alpha_{11}-\tilde{\alpha}_1 & \alpha_{12} & \cdots & \alpha_{1m}\\ \alpha_{21} & \alpha_{22}-\tilde{\alpha}_2 & \cdots & \alpha_{2m}\\ \vdots & \vdots & \ddots & \vdots\\ \alpha_{m1} & \alpha_{m2} & \cdots & \alpha_{mm}-\tilde{\alpha}_m\end{Vmatrix}, \qquad (15)$$

$$\mathbf{\tilde{\alpha}_i=\sum_{j=1}^{n}\alpha_{ij}\ ,\ C(q_i)=q_i-\sum_{j=m+1}^{n}\alpha_{ij}T_j=q_i-R_i}\ .$$

Обозначим элементы матрицы $\mathbf{A(\alpha)}$ через $\mathbf{a_{ij}(\alpha)}$, а элементы обратной ей матрицы $\mathbf{A^{-1}}$ через $\mathbf{b_{ij}(\alpha)}$ и выразим температуры подсистем через потоки $\mathbf{q_i}$ и заданные температуры резервуаров:

$$\mathbf{T=A^{-1}C(q),}$$

$$\mathbf{T_i(q)=b_{i1}(q_1-R_1)+b_{i2}(q_2-R_2)+...} \qquad (16)$$

$$\mathbf{+b_{im}(q_m-R_m).}$$

При этом $\mathbf{\frac{\partial T_i}{\partial q_\nu}=b_{i\nu},\ i=1,...,m,\ \nu=1,...,m}$.

Перейдем от задачи (8), (9), (4) относительно $\mathbf{u_i}$, $\mathbf{T_i}$ к оптимизационной задаче относительно потоков тепла $\mathbf{q_i}$: $\mathbf{P=\sum_{i=1}^{m} q_i \to \max\limits_{q_i}}$ при условии

$$\mathbf{P=\sum_{i=1}^{m}\frac{\alpha_i q_i}{\alpha_i T_i(q)-q_i}=\sigma_s}. \qquad (17)$$

Выпишем функцию Лагранжа для задачи (17):

$$\mathbf{L=\sum_{i=1}^{m} q_i\left(1-\frac{\Lambda_s\alpha_i}{\alpha_i T_i(q)-q_i}\right)}. \qquad (18)$$

Условие ее стационарности по $\mathbf{q_j}$ примет форму

$$\mathbf{\frac{\partial L}{\partial q_j}=0\Rightarrow\sum_{i=1}^{m}\frac{\alpha_i^2 q_i\Lambda_s b_{ij}}{(\alpha_i T_i(q)-q_i)^2}+1-}$$

$$\mathbf{-\frac{\alpha_j^2 T(q)_i\Lambda_s}{(\alpha_j T_j(q)-q_j)^2}=0,\ j=1,...,m}. \qquad (19)$$

Решив систему из **m+1** уравнений (19), (9) относительно $\mathbf{q_i}$, находим оптимальные значения $\mathbf{q_i^*(\sigma_s)}$ и $\mathbf{\Lambda_s^*(\sigma_s)}$, а затем значения $\mathbf{T_i^*(\sigma_s)}$, $\mathbf{u_i^*(\sigma_s)}$ и $\mathbf{P_i^*(\sigma_s)}$. При таком решении удобнее задавать не $\boldsymbol{\sigma_s}$, а $\mathbf{\Lambda_i^*}$, затем по формуле (9) рассчитывать соответствующее решению значение потока энтропии $\boldsymbol{\sigma_s}$.

В третьей задаче требуется найти максимальную суммарную мощность, извлекаемую из резервуаров и подсистем.

Выпишем функцию Лагранжа задачи (10):

$$\mathbf{L=P_s^*(\sigma_s)+P_r^*(\sigma_r)+\lambda(\sigma_s+\sigma_r)}. \qquad (20)$$

Из условия стационарности функции Лагранжа по $\boldsymbol{\sigma_s}$, $\boldsymbol{\sigma_r}$ получаем соотношение

$$\mathbf{\frac{\partial P_s^*}{\partial\sigma_s}=\frac{\partial P_r^*}{\partial\sigma_r}}. \qquad (21)$$

Так как производные $\mathbf{\frac{\partial P_s^*}{\partial\sigma_s}}$ и $\mathbf{\frac{\partial P_r^*}{\partial\sigma_r}}$ равны $\mathbf{\Lambda_s^*(\sigma_s)}$ и $\mathbf{\Lambda_r^*(\sigma_r)}$ соответственно, извлекаемая мощность $\mathbf{P(q_i)}$ максимальна, когда

$$\mathbf{\Lambda_r^*(\sigma)=\Lambda_s^*(-\sigma),\sigma=\sigma_r=-\sigma_s}. \qquad (22)$$

Приведем **алгоритм расчета максимальной мощности**, извлекаемой тепломеханическим преобразователем. Поскольку решение задачи разделено на три этапа, рассмотрим поэтапный алгоритм.

1. Заданные переменные: температуры резервуаров $\mathbf{T_i}$, **i=m+1, n [K]**; матрицу коэффициентов теплопереноса $\boldsymbol{\alpha_{ij}}$, **i, j=1, …, n [Вт/К]** коэффициенты теплопереноса для контактов преобразователя с подсистемами и резервуарами $\boldsymbol{\alpha_i}$, **i=1, ..., n [Вт/К]**.

2. Выпишем алгоритм нахождения максимальной извлекаемой мощности при контакте с резервуаром при заданном значении $\boldsymbol{\sigma_r}$:

a) найдем значение приведенной температуры контакта $\mathbf{\Lambda(\sigma_r)}$ по формуле (11) и значение температур контакта преобразователя с резервуарами по формуле (13);

b) найдем значения максимальной извлекаемой мощности как функцию потока энтропии $\mathbf{P_r(\sigma_r)}$ по формуле (12).

3. Выпишем алгоритм нахождения максимальной извлекаемой мощности при контакте с подсистемами. Поскольку для решения этой задачи необходимо решить систему из **m+1** уравнений (19), (9) относительно $\mathbf{q_i}$ и $\mathbf{\Lambda_s}$, будем использовать для этого метод Ньютона:

a) обратим матрицу $\mathbf{A(\alpha)}$ (15) любым доступным методом и выразим значения температур подсистем $\mathbf{T_i}$ относительно $\mathbf{q_i}$ по формуле (16);

b) составим матрицу Якоби $\mathbf{W(q,\Lambda_s)}$ для системы (19), (9) относительно $\mathbf{q_i}$, **i=1, …, m** и $\mathbf{\Lambda_s}$ с учетом того, что

$$\mathbf{F_j=\sum_{i=1}^{m}\frac{\alpha_i^2 q_i\Lambda_s b_{ij}}{(\alpha_i T_i(q)-q_i)^2}+1-\frac{\alpha_j^2 T(q)_i\Lambda_s}{(\alpha_j T_j(q)-q_j)^2}=0,\ j=1,..,m}$$

$$\mathbf{F_{m+1}=\sum_{i=1}^{m}\frac{\alpha_i q_i}{\alpha_i T_i(q)-q_i}-\sigma_s}.$$

c) в качестве начального приближения возьмем $\mathbf{q_i^0=0}$, $\mathbf{\Lambda_s^0=\frac{1}{n-m}\sum_{i=m+1}^{n} T_i}$;

d) найдем следующие приближения:

$$\mathbf{<q^{k+1},\Lambda_s^{k+1}>=<q^k,\Lambda_s^k>-W^{-1}(q^k,\Lambda_s^k)F(q^k,\Lambda_s^k)};$$

e) вычислим значение функции **F** в точке $\mathbf{q^{k+1}}$ $\mathbf{\Lambda_s^{k+1}}$; повторим итерацию до необходимой точности.

f) найдя оптимальные значения $\mathbf{q_i^*(\sigma_s)}$ и $\mathbf{\Lambda_s^*(\sigma_s)}$, можно найти температуры подсистем $\mathbf{T_i^*}$ по формуле (16) и значение температур контакта $\mathbf{u_i^*}$ и извлекаемой мощности $\mathbf{P_s^*}$.

4. Вычислим значения $\mathbf{\Lambda_r(\sigma_r)}$ и $\mathbf{P_r^*(\sigma_r)}$ для значений $\mathbf{\sigma_r=0}$.

5. Решим вторую задачу для $\mathbf{\sigma_r=-\sigma_s}$. Вычислим $\mathbf{\Lambda_s^*(\sigma_s)}$.

6. Если $\mathbf{\left|\Lambda_s^*(\sigma_s)-\Lambda_r^*(\sigma_r)\right|>\delta}$, перейдем к п. 4 с учетом $\mathbf{\sigma_r=\sigma_r+\varepsilon}$.

7. Вычислим значение извлекаемой мощности для полученных оптимальных $\mathbf{\Lambda_s}$ и $\boldsymbol{\sigma_s}$.

Пример. Рассмотрим систему, состоящую из двух резервуаров, четырех подсистем и преобразователя. Структура системы показана на рисунке.

Матрица коэффициентов теплопереноса имеет вид

$$\{\alpha_{ij}\}=\begin{Vmatrix} 0 & 800 & 900 & 700 & 4000 \\ 8000 & 500 & 9000 & 100 & \\ 900 & 5000 & 300 & 2000 & \\ 700 & 9000 & 3000 & 0 & 250 \\ 4000 & 2000 & 0 & 0 & \\ 0 & 1000 & 250 & 0 & 0 \end{Vmatrix}.$$

Значения α_{ij} при **i, j=1, ..., 4** соответствуют коэффициентам взаимодействия подсистем между собой, значения при **i>4** или **j>4** – коэффициентам взаимодействия подсистем с резервуарами. Температуры резервуаров: T_+**=700** K, T_-**=300** K. Коэффициенты теплопереноса при взаимодействии преобразователя с подсистемами и резервуарами:
$\alpha_1=1000, \alpha_2=1300, \alpha_3=900, \alpha_4=800, \alpha_5=100, \alpha_6=50$.

Значения α_i при **i=1, ..., 4** соответствуют коэффициентам взаимодействия с подсистемами, а при **i>4** – коэффициентам взаимодействия с резервуарами. Коэффициенты имеют размерность Вт/К.

Сначала были найдены значения $\Lambda_r(\sigma_r)$ по формуле (11) и $P_r^*(\sigma_r)$ по формуле (12) для значений σ_r, лежащих в интервале **[0, 0,5]**. Затем решалась система из **m** уравнений (19) для фиксированного значения Λ_s на отрезке **[400, 600]** K и по формуле (9) рассчитывалось соответствующее значение σ_s.

После этого находились оптимальные $\boldsymbol{\sigma}$ и $\boldsymbol{\Lambda}$ по формуле (22), значения температур подсистем T_i по формуле (16) и температур контакта преобразователя u_i по формуле (13).

В результате были получены следующие значения оптимальных температур: T_1**=564,8** K; T_2**=540,2** K; T_3**=563,7** K; T_4**=527** K.

Оптимальные температуры контакта преобразователя с подсистемами: u_1**=557,8** K; u_2**=543,6** K; u_3**=557,7** K; u_4**=536,3** K; u_5**=620,2** K; u_6**=406** K.

Значение предельной извлекаемой мощности P^***=3,22** кВт.

В работе получены оценки максимальной извлекаемой мощности для произвольной стационарной термодинамической системы и соответствующие ей распределения потоков тепла и температур контакта рабочего тела с подсистемами. Предложен алгоритм расчета максимальной извлекаемой мощности и оптимальных характеристик тепловой машины.

Литература

1. Novikov I.I. The efficiency of atomic power stations // J. Nuclear Energy II. 1958. № 7, pp. 25–128.

2. Curzon F.L., Ahlburn B. Efficiency of a Carnot engine at maximum power output. Amer.J. Physics. 1975. № 43, pp. 22–24.

3. Amelkin S.A., Andresen B., Burzler J.M., Hoffmann K.H., Tsirlin A.M. Maximum power processes for multi-source endoreversible heat engines J. Phys. D: Appl. Phys. 2004. № 37, pp. 1400–1404.

4. Tsirlin A.M., Kazakov V., Ahremenkov A.A., , Alimova N.A. Thermodynamic constraints on temperature distribution in a stationary system with heat engine or refrigerator J. Phys. D: Appl. Phys. 2006. № 39, pp. 4269–4277.

ПРОГРАММНЫЙ КОМПЛЕКС ОПТИМИЗАЦИИ ЗАКОНОВ УПРАВЛЕНИЯ

(Работа выполнена при финансовой поддержке РФФИ, проект № 08-01-00274-а).

А.О. Блинов; В.И. Гурман, д.т.н.; Е.А. Трушкова, к.ф.-м.н.; В.П. Фраленко
(ИПС им. А.К. Айламазяна РАН, г. Переславль-Залесский, sarmat@pereslavl.ru)

SOFTWARE PACKAGE OF CONTROL OPTIMIZATION

Blinov Alexander O.; Gurman Vladimir I., Ph.D.; Trushkova Ekaterina A., Ph.D.; Fralenko Vitaly P.
(PSI RAS, Pereslavl-Zalessky, sarmat@pereslavl.ru)

Abstract. The software package «Improvement and Synthesis of Control» (SP ISCON) is constructed in Control Processes Research Center of Program Systems Institute of RAS. It contains approximation, optimization and improvement algorithms of approximately optimal control for different dynamic control processes which are successfully implemented on computer cluster. The efficiency of PC ISCON are analyzed on a series of model and applied problems.

Keywords: *optimal control problem, approximation, least-squares method, improvement method, parallel algorithm, programming language T++.*

В статье описывается программный комплекс ISCON (Improvement and Synthesis of Control), в котором успешно реализованы на кластерном вычислительном устройстве алгоритмы аппроксимации, оптимизации и улучшения приближенно-оптимального управления для различных динамических процессов с управлением. Проведено исследование возможностей и эффективности ПК ISCON на ряде модельных и прикладных задач.

Ключевые слова: *задача оптимального управления, аппроксимация, метод наименьших квадратов, метод улучшения, параллельный алгоритм, язык программирования T++.*

Статья посвящена разработке и реализации методов синтеза оптимальных целевых законов управления, что является кардинальной проблемой теории и практики управления. Ее решение

связано с бесконечномерными обратными задачами, аппроксимируемыми при численной реализации многомерными задачами, требующими неограниченно растущих вычислительных ресурсов для приближения к точным решениям. Это приводит к неизбежному выводу о необходимости реализовать решение подобных задач на современных высокопроизводительных вычислительных системах параллельной архитектуры. Парадигма параллельных вычислений в высшей степени соответствует самой природе рассматриваемых задач, связанных с множественностью однотипных операций на верхнем уровне, таких как решение обратной задачи через множество решений прямой (прямо или косвенно), а также формирование и численная реализация полей управлений (ситуационных управлений).

В статье приводятся результаты разработки, реализации и исследования экспериментального варианта *программного комплекса* (ПК) улучшения и оптимизации законов управления для приложений в различных областях (ПК ISCON) с распараллеливанием и переносом на кластер.

Комплекс предназначен для моделирования сложных динамических процессов, решения оптимизационных задач и задач улучшения управления для различных прикладных областей на кластерном вычислительном устройстве. С этой целью в нем реализованы алгоритмы аппроксимации, оптимизации и улучшения приближенно-оптимального управления. Главными компонентами комплекса являются графический интерфейс, сервер управления, управляющие модули и набор исполняемых модулей.

В графическом интерфейсе происходят ввод начальных данных, постановка задачи, выбор метода решения задачи, управление потоками данных, визуализация и сохранение результатов. Сервер управления участвует в обеспечении пользователей доступом к возможностям комплекса, принимает запросы на решение выбранных задач с выбранными пользователем настройками. Управляющие модули принимают полученную от сервера управления информацию и выполняют развертывание полигона для вычислений, запуская в дальнейшем либо локально, либо удаленно исполняемый модуль решаемой задачи. Кроме того, модули обеспечивают сбор выходных данных и их передачу обратно серверу управления.

Основной идеей при разработке архитектуры системы было обеспечение гибкости и расширяемости. Выбранная модульная схема ПК обеспечивает расширяемость и масштабируемость. Гетерогенность вычислительной среды поддержана включением в архитектуру ПК управляющих модулей, связывающих физически разделенные компоненты ПК. В зависимости от набора исполняемых модулей ПК может использоваться при создании систем, относящихся к различным задачам оптимизации и управления. Гибкость системы обеспечивается возможностью подключения доступных вычислительных устройств за счет конфигурирования управляющих модулей.

Особенности ПК:

- использование параллелизма на различных уровнях: параллельное выполнение решаемых задач, внутренний параллелизм модулей;
- модули системы реализуются в виде исполняемых файлов и могут содержать как последовательную, так и параллельную реализацию алгоритма.

Управление ПК поддерживается графическим интерфейсом, интегрированным с сервером управления.

Область применения ПК определяется реализованными методами (исполняемыми модулями), предназначенными для построения аналитического описания модели по имеющимся статистическим данным (алгоритм аппроксимации по методу наименьших квадратов), улучшения приближенно-оптимальной программы управления динамической системой, моделирования и исследования упругих систем. Указанные методы могут применяться, например, к задачам управления экономическими системами, движением летательных аппаратов (вертолеты, космические челноки), задачам оптимального конструирования упругих стержней.

Программа аппроксимации моделей динамических систем

При работе с моделями реальных динамических систем $\dot{\mathbf{x}}(t)=\mathbf{f}(t,\mathbf{x}(t),\mathbf{u}(t)),\ \mathbf{x}\in\mathbf{R}^n,\ \mathbf{u}\in\mathbf{R}^p$, типичны ситуации, когда модель имеет структуру, к которой невозможно применить тот или иной метод исследования или алгоритм синтеза управления. Чтобы упростить систему, предлагается применять алгоритм аппроксимации многомерных функций многих переменных $\mathbf{f(t, x, u)}$ (при фиксированных моментах времени) по методу наименьших квадратов многомерными полиномами:

$$\varphi(\mathbf{z})=\sum_{j=1}^{m}\psi_j\mathbf{g}_j(\mathbf{z}),\ \mathbf{z}=\left(\mathbf{z}^1,\ldots,\mathbf{z}^{n+p}\right)=(\mathbf{x},\mathbf{u})\in\mathbf{R}^{n+p},$$

где $\left\{\mathbf{g}_j(\mathbf{z})=\prod_{i=1}^{n+p}\left(\mathbf{z}^i\right)^{k_i(j)}\right\}$ – некоторый набор заданных базисных функций; $\mathbf{k}_i(\mathbf{j})$ – целые положительные числа; $\{\psi_j\}$ – соответствующий набор коэффициентов, подлежащих определению. Решается следующая задача минимизации:

$$\mathbf{S}=\sum_{i=1}^{\beta}\left(\varphi(\mathbf{z}_i)-\mathbf{f}(t,\mathbf{x}_i,\mathbf{u}_i)\right)^2\rightarrow\min_{\{\psi_j\}},$$

где β – количество узлов аппроксимации. Данная задача сводится к решению системы линейных алгебраических уравнений с постоянными коэффи-

циентами.

Для этого надо сформировать с помощью узлов аппроксимации и базисных функций приближающего полинома матрицу и столбец свободных членов для системы линейных алгебраических уравнений и решить полученную систему.

Реализована параллельность указанного метода в первой части, то есть область формирования исходных данных разбивается на части, в каждой из которых строятся матрица и столбец свободных членов для системы уравнений с помощью исходных значений в узлах текущей части. Общая матрица получается в этом случае как сумма всех построенных частичных матриц (что справедливо и для свободных членов).

Алгоритм реализован в Т-системе (язык программирования Т++) на кластере *skif.botik.ru* [1].

Входными данными задачи являются: размерность фазового пространства **n**; размерность пространства управлений **p**; правая часть управляемой системы **f(t, x, u)**; нижние ограничения на фазовую траекторию и управление; верхние ограничения на фазовую траекторию и управление; базисные функции (**j** векторов, содержащих степени вхождения переменных $\mathbf{z}^{\mathbf{i}}$ в базисную функцию $\mathbf{g}_{\mathbf{j}}$).

Выходными данными задачи являются коэффициенты аппроксимирующих полиномов (**n** действительных векторов размера **j**).

Настройка ПК по решению описанной задачи произведена для правой части динамической системы вида:

$$\mathbf{f}(\mathbf{x}^1,\mathbf{x}^2,\mathbf{x}^3,\mathbf{x}^4,\mathbf{u}^1,\mathbf{u}^2)=\begin{pmatrix}\mathbf{f}_1(\mathbf{x},\mathbf{u})\\ \mathbf{f}_2(\mathbf{x},\mathbf{u})\\ \mathbf{f}_3(\mathbf{x},\mathbf{u})\\ \mathbf{f}_4(\mathbf{x},\mathbf{u})\end{pmatrix}, \qquad (1)$$

где $\mathbf{f}_1(\mathbf{x},\mathbf{u})=\mathbf{x}^1-\mathbf{0.44}\cdot\mathbf{10}^{-5}\sqrt{(\mathbf{x}^1)^2+(\mathbf{x}^2)^2}\,\mathbf{x}^1-\mathbf{0.10u}^1$,

$$\mathbf{f}_2(\mathbf{x},\mathbf{u})=\mathbf{x}^2-\mathbf{0.44}\cdot\mathbf{10}^{-5}\sqrt{(\mathbf{x}^1)^2+(\mathbf{x}^2)^2}\,\mathbf{x}^2+\mathbf{0.12}\cdot$$
$$\cdot\mathbf{10}^{-2}(\mathbf{x}^3)^2(-\mathbf{0.05}+\mathbf{0.29}\cdot\mathbf{10}^{-2}\mathbf{x}^3-\mathbf{0.53}\cdot\mathbf{10}^{-4}(\mathbf{x}^3)^2+$$
$$+\mathbf{x}^1\left(\mathbf{0.01}-\mathbf{0.29}\cdot\mathbf{10}^{-3}\mathbf{x}^3+\mathbf{0.47}\cdot\mathbf{10}^{-5}(\mathbf{x}^3)^2\right)+$$
$$+\mathbf{x}^2\left(\mathbf{0.01}-\mathbf{0.59}\cdot\mathbf{10}^{-3}\mathbf{x}^3+\mathbf{0.12}\cdot\mathbf{10}^{-4}(\mathbf{x}^3)^2\right)+$$
$$+\mathbf{u}^1\left(\mathbf{0.08}-\mathbf{0.32}\cdot\mathbf{10}^{-2}\mathbf{x}^3+\mathbf{0.42}\cdot\mathbf{10}^{-4}(\mathbf{x}^3)^2\right)+$$
$$+\mathbf{u}^2\left(\mathbf{0.46}-\mathbf{0.01x}^3+\mathbf{0.15}\cdot\mathbf{10}^{-3}(\mathbf{x}^3)^2\right)-\mathbf{0.10},$$

$$\mathbf{f}_3(\mathbf{x},\mathbf{u})=\mathbf{x}^3+\mathbf{0.03}-\mathbf{0.52}\cdot\mathbf{10}^{-4}(\mathbf{x}^3)^2+$$
$$+\mathbf{0.01x}^1(\mathbf{0.53}-\mathbf{0.04x}^3+\mathbf{0.63}\cdot\mathbf{10}^{-3}(\mathbf{x}^3)^2)+$$
$$+\mathbf{0.01x}^2\left(-\mathbf{1.23}+\mathbf{0.08x}^3-\mathbf{0.15}\cdot\mathbf{10}^{-2}(\mathbf{x}^3)^2\right)+$$
$$+\mathbf{0.01u}^1\left(\mathbf{8.46}-\mathbf{0.52x}^3+\mathbf{0.01}(\mathbf{x}^3)^2\right)+$$
$$+\mathbf{0.01u}^2\left(\mathbf{75.61}-\mathbf{5.46x}^3+\mathbf{0.08}(\mathbf{x}^3)^2\right)-\frac{\mathbf{0.07}}{\mathbf{x}^3},$$

$$\mathbf{f}_4(\mathbf{x},\mathbf{u})=\mathbf{x}^4+\mathbf{0.01x}^2.$$

Для проведения аппроксимации указанной функции **f(x, u)**, например, линейной функцией

$$\boldsymbol{\varphi}(\mathbf{x},\mathbf{u})=\boldsymbol{\psi}_1+\boldsymbol{\psi}_2\mathbf{x}^1+\boldsymbol{\psi}_3\mathbf{x}^2+\boldsymbol{\psi}_4\mathbf{x}^3+\boldsymbol{\psi}_5\mathbf{x}^4+\boldsymbol{\psi}_6\mathbf{u}^1+\boldsymbol{\psi}_7\mathbf{u}^2,$$

в области изменения переменных

$$\mathbf{0.28}\le\mathbf{x}^1\le-\mathbf{3.2},\ -\mathbf{3.2}\le\mathbf{x}^2\le\mathbf{0},\ \mathbf{24.6}\le\mathbf{x}^3\le\mathbf{30.8},$$
$$-\mathbf{6}\le\mathbf{x}^4\le\mathbf{0},\ -\mathbf{0.35}\le\mathbf{u}^1\le\mathbf{0.35},\ -\mathbf{0.35}\le\mathbf{u}^2\le\mathbf{0.35}$$

необходимо задать следующие входные данные:

- размерность фазового пространства **n=4**;
- размерность пространства управлений **p=2**;
- правая часть управляемой системы **f(t, x, u)** (см. выше);
- нижние ограничения на траекторию и управление: (0.278, -3.2, 24.6, -6, -0.349, 0.07853);
- верхние ограничения на траекторию и управление: (7.5, 0, 30.8, 0, 0.349, 0.349);
- базисные функции: (0, 0, 0, 0, 0, 0), (1, 0, 0, 0, 0, 0), (0, 1, 0, 0, 0, 0), (0, 0, 1, 0, 0, 0), (0, 0, 0, 1, 0, 0), (0, 0, 0, 0, 1, 0), (0, 0, 0, 0, 0, 1).

Для оценки эффективности распараллеливания программы проведены запуск программы на различном числе узлов и замер времени работы в каждом случае. Полученные данные (табл. 1) позволяют сделать вывод об эффективном распараллеливании указанного класса алгоритмов для небольшого числа узлов (1–8).

Таблица 1

Анализ эффективности работы параллельной версии программы аппроксимации по МНК

Число узлов (n)	Время работы программы (t_n, с)	Ускорение (t_1/t_n)
1	3338,348	1
2	1779,791	1,876
3	1237,142	2,698
4	880,248	3,793
5	729,924	4,574
6	631,720	5,285
7	632,003	5,282
8	586,202	5,695
9	631,214	5,289
10	596,175	5,600
11	588,017	5,677
12	596,195	5,599
13	589,926	5,659
14	586,519	5,692
15	597,649	5,586
16	579,739	5,758

Программа улучшения управления

Предполагается, что модель движения в общем случае представляет собой дискретную управляемую систему, терминальный функционал качества, ограничения на управления типа неравенств, фазовые ограничения типа неравенств (различные внутри и на правом конце заданного фиксированного промежутка времени):

$$\mathbf{x}(\mathbf{t}+\mathbf{1})=\mathbf{f}\left(\mathbf{t},\mathbf{x}(\mathbf{t}),\mathbf{u}(\mathbf{t})\right),\ \mathbf{t}\in\mathbf{T}=\left\{\mathbf{t}_I,\mathbf{t}_I+\mathbf{1},...,\mathbf{t}_F\right\},$$
$$\mathbf{x}(\mathbf{t}_I)=\mathbf{x}_I,\ \mathbf{u}(\mathbf{t})\in\mathbf{D}_u,$$
$$\mathbf{D}_u=\left\{\mathbf{u}:\mathbf{T}\backslash\left\{\mathbf{t}_F\right\}\to\mathbf{R}^p\middle|\mathbf{u}_l\le\mathbf{u}(\mathbf{t})\le\mathbf{u}_u,\mathbf{t}\in\mathbf{T}\backslash\left\{\mathbf{t}_F\right\}\right\},$$
$$\mathbf{x}_l\le\mathbf{x}(\mathbf{t})\le\mathbf{x}_u,\ \mathbf{t}\in\mathbf{T}\backslash\left\{\mathbf{t}_F\right\},\ \mathbf{x}_{lF}\le\mathbf{x}(\mathbf{t}_F)\le\mathbf{x}_{uF},$$
$$\mathbf{I}=\mathbf{F}_0\left(\mathbf{x}(\mathbf{t}_F)\right)\to\mathbf{min},\ \mathbf{u}\in\mathbf{R}^p,\ \mathbf{x}\in\mathbf{R}^n.$$

Производится замена этой задачи оштрафованной, то есть задачей без фазовых ограничений,

с помощью введенных функций штрафа типа срезок:

$$\mathbf{x(t+1)=f\big(t,x(t),u(t)\big),\ t\in T,\ x(t_I)=x_I,}$$
$$\mathbf{z(t+1)=z(t)+t_F^{-1}\delta\big(x(t)\big),\ z(t_I)=0,}$$
$$\mathbf{F=\beta_0F_0\big(x(t_F)\big)+\beta_1^Tz(t_F)+\beta_2^T\delta_F\big(x(t)\big)\to min,}$$
$$\mathbf{\beta_0\in R,\ \beta_1,\beta_2\in R^n,}$$

где $\mathbf{\delta^i(x)=-min\{0,x^i-x_l^i\}+max\{0,x^i-x_u^i\},}$

$\mathbf{\delta_F^i(x)=-min\{0,x^i-x_{lF}^i\}+max\{0,x^i-x_{uF}^i\},\ i=\overline{1,n}.}$

Задача улучшения ставится следующим образом: пусть известен допустимый элемент $\mathbf{m^I=\big(u^I(t),\ x^I(t)\big)}$, требуется найти допустимый элемент $\mathbf{m^{II}=\big(u^{II}(t),\ x^{II}(t)\big)}$, такой, что $\mathbf{F\big(x^{II}(t_F),\ z^{II}(t_F)\big)<F\big(x^I(t_F),\ z^I(t_F)\big)}$.

Итерационное улучшение основано на линейно-квадратических аппроксимациях соотношений Беллмана в среднем в окрестности текущего приближения полиномами второго порядка [2, 3]. Предусмотрены регуляторы, настраиваемые так, чтобы каждая итерация была наиболее эффективной.

На основе принципа оптимальности Кротова элемент $\mathbf{m^{II}}$ ищется путем аппроксимации решения следующей задачи:

$$\mathbf{y(t+1)=g\big(t,y(t),v(t)\big),\ t\in T,\ y(t_I)=0,}$$
$$\mathbf{s(t+1)=s(t)+t_F^{-1}\delta\big(y(t)+x^I(t)\big)-t_F^{-1}\delta\big(x^I(t)\big),}$$
$$\mathbf{s(t_I)=0,}$$
$$\mathbf{y^0(t+1)=y^0(t)+\frac{1}{2}v^T(t)v(t),\ y^0(t_I)=0,}$$
$$\mathbf{G_\alpha=\alpha y^0\big(t_F\big)+}$$
$$\mathbf{+(1-\alpha)F\big(y(t_F)+x^I(t_F),s(t_F)+z^I(t_F)\big)\to min,}$$

где α – некоторое число из отрезка **[0,1]** (регулятор метода), $\mathbf{y=x-x^I}$, $\mathbf{s=z-z^I}$, $\mathbf{v=u-u^I}$, $\mathbf{g(t,y,v)=f(t,y+x^I,v+u^I)-f(t,x^I,u^I)}$.

Отыщем функцию Кротова в виде

$$\mathbf{\varphi(t,y^0,y,s)=w(t)+\psi^0(t)y^0+\psi^T(t)y+\gamma^T(t)s,}$$

где значения $\mathbf{w(t),\ \psi^0(t),\psi(t),\ \gamma(t)}$ находятся из следующих приближенных соотношений Кротова–Беллмана:

$$\mathbf{\varphi(t_F,y^0,y,s)\approx-G_\alpha(y^0,y,s),}$$
$$\mathbf{\varphi(t,y^0,y,s)\approx\max_v\varphi\Big(t+1,y^0+\frac{1}{2}v^Tv,g(t,y,v),}$$
$$\mathbf{s+\frac{1}{t_F}(\delta(y+x^I)-\delta(x^I))\Big),\ t=t_F-1,...,t_I.}$$

При этом управление (в форме синтеза), на котором достигается максимум, обозначим $\mathbf{\hat v(t,y)}$.

Опишем одну итерацию алгоритма метода улучшения. По данному начальному приближению $\mathbf{m^I=\big(u^I(t),x^I(t)\big)}$ выбираем весовые коэффициенты функционала $\mathbf{F\big(x(t_F),z(t_F)\big)}$ из следующих условий:

$$\mathbf{\beta_0=1,\beta=\big(\beta_1^1,...,\beta_1^n,\beta_2^1,...,\beta_2^n\big)^T=0,}$$
$$\text{if } \mathbf{J_0=\{1,...,2n\},}$$
$$\mathbf{\beta_0=0,\beta^j=0,\ j\in J_0,\ \beta^j=\frac{P}{(Sh^j)},\ j\in J,}$$
$$\text{if } \mathbf{J_0\neq\{1,...,2n\},}$$

где $\mathbf{h=\big(z^T(t_F),\ \delta_F^T(t_F)\big)^T}$, $\mathbf{J_0=\big\{j\in\{1,..\mu_X,2n\}\big|h^j\le0.001\big\}}$,

$\mathbf{J=\{1,...,2n\}\setminus J_0}$, $\mathbf{P=\prod_{j\in J}h^j}$, $\mathbf{S=\sum_{j\in J}\frac{P}{h^j}}$.

Фиксируем набор параметров метода. Разложив правые части соотношений Кротова–Беллмана при фиксированном $\mathbf{t\in T}$ в ряд до членов второго порядка в окрестности нуля и заменив производные их разностными аналогами (шаги разностных схем – дополнительные параметры метода), получим управление

$$\mathbf{u^{II}(t,x)=\hat v(t,x-x^I(t))+u^I(t),t\in T\setminus\{t_F\}}$$

и соответствующий элемент

$$\mathbf{m^{II}=\big(u^{II}(t)=u^{II}\big(t,x^{II}(t)\big),x^{II}(t)\big).}$$

Если улучшение произошло, проводим следующую итерацию, выбрав в качестве начального элемента $\mathbf{m^{II}}$. В противном случае берем другой набор параметров метода или останавливаем итерации.

Построенный таким образом алгоритм естественным образом ориентирован на параллельные вычисления. Алгоритм реализован в Т-системе (язык программирования Т++) на кластере *skif.botik.ru* [4].

Входными данными задачи являются:

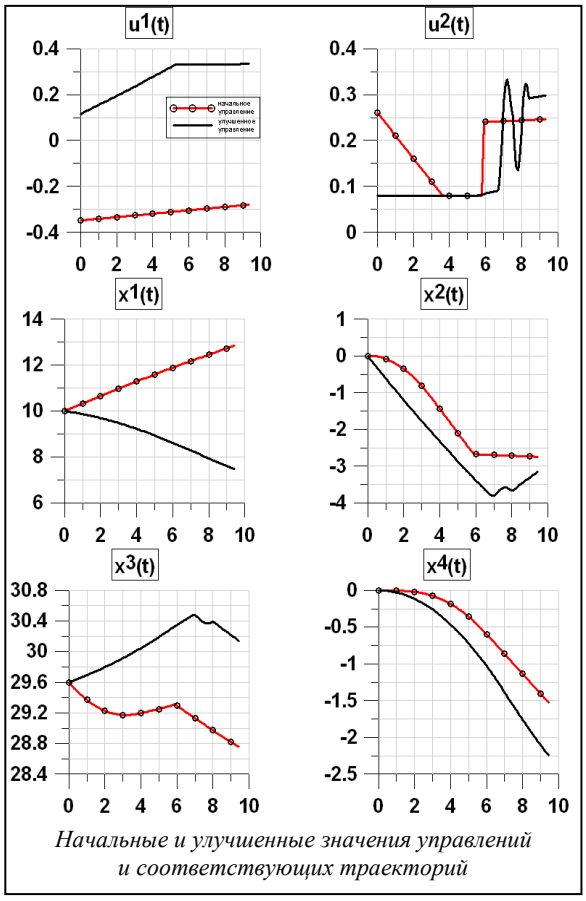
1) размерность фазового пространства **n**;
2) размерность пространства управлений **p**;
3) начальный момент $\mathbf{t_I}$;
4) конечный момент $\mathbf{t_F}$;
5) число отрезков разбиения временного интервала **m**;
6) правая часть управляемой системы **f(t, x, u)**;
7) минимизируемый функционал $\mathbf{F_0(t_F,x(t_F))}$;
8) начальное значение фазовой траектории $\mathbf{x_I}$;
9) нижние ограничения на траекторию внутри временного отрезка, вектор-индикатор наличия/отсутствия этих ограничений;
10) верхние ограничения на траекторию внутри временного отрезка, вектор-индикатор наличия/отсутствия этих ограничений;
11) нижние ограничения на траекторию в момент $\mathbf{t_F}$, вектор-индикатор наличия/отсутствия этих ограничений;
12) верхние ограничения на траекторию в момент $\mathbf{t_F}$, вектор-индикатор наличия/отсутствия этих ограничений;
13) нижние ограничения на управление;
14) верхние ограничения на управление;
15) минимальное время перекладки управле-

ния;

16) начальная программа управления.

При этом данные 1–8 являются обязательными.

Результаты работы программы выводятся в текстовый файл. В нем указываются набор параметров и номер итерации, на которой достигнуто наибольшее улучшение; далее таблицей идут



Начальные и улучшенные значения управлений и соответствующих траекторий

столбцы результатов: временной момент, значения траектории в этот момент (покоординатно), значения управлений в этот момент (покоординатно), значения отклонений от допустимого множества (покоординатно); в заключение приводится достигнутое значение целевой функции. Предусмотрена возможность вывода управления в форме синтеза ($\mathbf{u(t,x)=A(t)x+B(t)}$): вывод двух текстовых файлов, один из которых содержит матрицу коэффициентов при переменной **x** (матрицу **A(t)**), другой – матрицу коэффициентов свободных членов (матрицу **B(t)**).

Формат файла выходных результатов позволяет быстро строить графики для наглядного представления произошедшего улучшения.

Настройка ПК по решению описанной задачи произведена, в частности, для задачи улучшения начального приближенно-оптимального управления для нелинейной системы, полученной при аппроксимации модели движения вертолета в нештатной ситуации [3]:

$$\mathbf{x(t+1)=f(x(t),u(t))},\ \mathbf{t\in\{0,1,\ldots,t_F\}},\ \mathbf{x\in R^4},\ \mathbf{u\in R^2},$$

где правая часть системы **f(x, u)** определена согласно формуле (1).

Заданы начальные значения фазовых переменных, ограничения на фазовые переменные во время и в конце маневра, ограничения на управления:

$$\mathbf{x(0)=(10,0,29.6,0)^T},$$
$$\mathbf{u_-=(-0.348,0.08)^T},\ \mathbf{u_+=(0.348,0.348)^T},$$
$$\mathbf{x_-=(0,-5,24.6,-\infty)^T},\ \mathbf{x_+=(+\infty,0,30.8,+\infty)^T},$$
$$\mathbf{x_{-F}=(0,-3.2,24.6,-\infty)^T},\ \mathbf{x_{+F}=(7.5,0,30.8,+\infty)^T}.$$

Требуется минимизировать конечную высоту $\mathbf{F_0(x(t_F))=x^4(t_F)}$, что равносильно максимизации нижней границы опасной зоны аварийной посадки.

Для улучшения одного из вариантов начального управления входные данные следует задать, например, в виде:

1) размерность фазового пространства **n=4**;
2) размерность пространства управлений **p=2**;
3) начальный момент $\mathbf{t_I=0}$;
4) конечный момент $\mathbf{t_F=9.47}$;
5) число отрезков разбиения **m=947**;
6) правая часть системы **f(x, u)** (см. ранее);
7) функционал $\mathbf{F_0(x(t_F))}$ (см. ранее);
8) начальное значение $\mathbf{x_I}$=(10, 0, 29.6, 0);
9) нижние ограничения внутри отрезка (0, -5, 24.6, 0), вектор-индикатор (1, 1, 1, 0);
10) верхние ограничения внутри отрезка (7.5, 0, 30.8, 0), вектор-индикатор (0, 1, 1, 0);
11) нижние ограничения в $\mathbf{t_F}$ (0, -3.2, 24.6, 0), вектор-индикатор (1, 1, 1, 0);
12) верхние ограничения в $\mathbf{t_F}$ (7.5, 0, 30.8, 0), вектор-индикатор (1, 1, 1, 0);
13) нижние ограничения на **u** (-0.348, 0.08);
14) верхние ограничения на **u** (0.348, 0.348);
15) минимальное время перекладки (0.7, 0.35);
16) начальная программа управления берется из файла *upr_nach.txt*.

Вычисления проводились для 256 различных наборов параметров метода, при этом удалось уменьшить значение целевого функционала, удовлетворив все ограничения. Результаты работы ПК для входных числовых данных, описанных ранее, представлены на рисунке.

Для оценки эффективности распараллеливания программы проведены запуск программы на различном числе узлов и замер времени работы в каждом случае. Полученные данные представлены в таблице 2.

Таблица 2

Анализ эффективности параллельной версии программы улучшения управления

Число узлов (n)	Время работы программы (t_n, с)	Ускорение (t_1/t_n)
1	1029,85	1
3	351,99	2,93

5	218,83	4,71
7	159,60	6,45
9	130,71	7,88
11	110,29	9,34
13	93,69	10,99
15	90,10	11,43

Эти данные позволяют сделать вывод об эффективном распараллеливании указанного класса алгоритмов.

В описанном ПК успешно реализованы алгоритмы аппроксимации, оптимизации и улучшения приближенно-оптимального управления. Он содержит сервер управления (средство управления и контроля комплексом), управляющие модули (инструменты формирования среды для решения поставленных задач), исполняемые модули (выполняют счет поставленных задач) и интерфейс пользователя (средство запуска счета параметризованных задач).

Для параллельной реализации ПК была использована гетерогенная аппаратная среда. Компоненты ПК физически разделены. Графический интерфейс, сервер управления и управляющие модули работают на платформе IBM PC, а аппаратная платформа для исполняемых модулей вообще не фиксируется. В составе ПК, в частности, есть исполняемые модули, работающие на аппаратной платформе IBM PC, модули, выполняющиеся на аппаратной платформе суперкомпьютеров «СКИФ» кластерного уровня. Аппаратная платформа суперкомпьютеров «СКИФ» включает управляющую ЭВМ (фронтенд), вычислительные узлы кластерного уровня; системную сеть кластера (SCI), объединяющую вычислительные узлы; вспомогательную сеть (семейства *Ethernet*, с поддержкой TCP/IP), объединяющую управляющую ЭВМ и вычислительные узлы.

Такая гибкость при работе с исполняемыми модулями оказалась возможной из-за активного использования протокола SSH (*Secure Shell*) при построении управляющих модулей, сетевого протокола прикладного уровня, позволяющего производить удаленное управление операционной системой и передачу файлов.

Результаты проведенного исследования возможностей и эффективности ПК ISCON позволяют сделать вывод о значительном ускорении вычислительного процесса, близкого к линейному до определенного (оптимального) количества вычислительных узлов. Это подтверждает теоретический прогноз о высокой эффективности распараллеливания описанного класса задач.

Литература

1. Белышев Д.В., Блинов А.О., Фраленко В.П. Параллельный алгоритм аппроксимации моделей управляемых систем: тр. Четвертой междунар. конф. (РАСО'2008). М.: ИПУ им. В.А. Трапезникова РАН. 2008. С. 968–978.

2. Гурман В.И. Принцип расширения в задачах управления. М.: Наука. Физматлит, 1985.

3. Гурман В.И., Квоков В.Н., Ухин М.Ю. Приближенные методы оптимизации управления летательным аппаратом // АиТ. 2008. № 3. С. 191–201.

4. Коваленко М.Р., Матвеев Г.А., Осипов В.И., Трушкова Е.А. Параллельный алгоритм улучшения управления: тр. Четвертой междунар. конф. (РАСО'2008). М.: ИПУ им. В.А. Трапезникова РАН. 2008. С. 979–984.

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНОЛОГИЯ ОПТИМИЗАЦИИ ПОЗИЦИОННЫХ УПРАВЛЕНИЙ В ДИФФЕРЕНЦИАЛЬНЫХ СИСТЕМАХ

(Работа выполнена при финансовой поддержке РФФИ, проекты 08-01-00945-а, 09-01-90203-Монг-а, и программы Президента РФ, проект НШ-1676.2008.1)

О.В. Моржин (ИПС им. А.К. Айламазяна РАН, г. Переславль-Залесский, oleg-morzhin@yandex.ru);
А.И. Тятюшкин, д.т.н. (ИДСТУ Сибирского отделения РАН, г. Иркутск, tjat@icc.ru)

A COMPUTATION TECHNOLOGY FOR POSITIONAL CONTROLS OPTIMIZATION IN DIFFERENTIAL SYSTEMS

Morzhin Oleg V. *(PSI RAS, Pereslavl-Zalessky, oleg-morzhin@yandex.ru);*
Tyatyushkin Alexander I., Ph.D. *(ISDCT, Siberian Branch, RAS, Irkutsk, tjat@icc.ru)*

Abstract. The article is devoted to a computation technology for positional controls optimization in nonlinear differential systems.

Keywords: *dynamic systems, reachable sets, positional control, numerical methods.*

Статья посвящена описанию вычислительной технологии оптимизации позиционных управлений в нелинейных дифференциальных системах.

Ключевые слова: *динамические системы, множества достижимости, позиционное управление, численные методы.*

Проблема построения управления с обратной связью в нелинейных управляемых системах до сих пор остается актуальной и находится в центре внимания специалистов по управлению.

Для решения *задач оптимального позиционного управления* (ЗОПзУ) нелинейными дифференциальными системами известны различные методики. Специфической чертой подхода Н.Н. Мои-

сеева [1] является возможность декомпозиции ЗОПзУ с аддитивным целевым функционалом на элементарные задачи за счет реализации принципа оптимальности Р. Беллмана на априорных «шкалах состояний» в пространстве «время–состояния». Известный метод «блуждающих трубок» [1], призванный дать оценку области, в которой требуется введение шкал состояний, при данных начальном и/или целевом множествах обеспечивает локальное решение задачи, будучи определенным на некоторых подмножествах трубок достижимости и/или разрешимости (управляемости).

Эффективность подхода определяется суммарной трудоемкостью значительного числа элементарных операций по вычислению траектории системы для перехода из каждого узла на дискретном множестве состояний, введенном для одного узлового момента времени, в каждый узел на аналогичном множестве для последующего узла по времени. Иными словами, требуется найти (по возможности глобальное) решение *задачи оптимального программного управления* (ЗОПрУ) для рассматриваемой динамической системы при терминальных ограничениях, где программными управлениями рассматривают функции или параметры.

Конструктивными направлениями в развитии подхода Н.Н. Моисеева представляются: 1) предварительная аппроксимация трубки достижимости при заданном начальном множестве или трубки разрешимости при данном целевом множестве; 2) решение ЗОПрУ на основе современных алгоритмических и программных средств. Конструктивное развитие схемы Моисеева предложено в работах [2–4] с приложениями к различным модельным задачам.

Формулировка задач оптимального управления

Рассматривается управляемая система

$$\dot{\mathbf{x}}(\mathbf{t}) = \mathbf{f}(\mathbf{x}(\mathbf{t}),\mathbf{u},\mathbf{t}),\ \mathbf{x}(\mathbf{t}) \in \mathbf{E}^{\mathbf{n}},\ \mathbf{t} \in [\mathbf{t}_S,\mathbf{t}_F], \quad (1)$$

где управление программное **u=u(t)** или позиционное **u=u(t, x)**. Классы доступных программных и позиционных управлений:

$$\mathbf{U}_{\mathbf{program}} = \left\{\mathbf{u} \in \mathbf{PC}(\mathbf{T},\mathbf{E}^{\mathbf{m}}):\ \mathbf{u}(\mathbf{t}) \in \mathbf{U},\ \mathbf{t} \in [\mathbf{t}_S,\mathbf{t}_F]\right\},$$

$$\mathbf{U}_{\mathbf{positional}} = \left\{\mathbf{u} \in \mathbf{PC}(\mathbf{T}\times\mathbf{X},\mathbf{E}^{\mathbf{m}}):\ \mathbf{u}(\mathbf{t},\mathbf{x}) \in \mathbf{U},\ \mathbf{t} \in [\mathbf{t}_S,\mathbf{t}_F],\ \mathbf{x} \in \mathbf{X}\right\},$$

где компакт **X** определяется в контексте конкретной задачи управления.

Множеством достижимости $\mathbf{R}(\mathbf{t}^{\mathbf{I}}, \mathbf{x}^{\mathbf{I}}, \mathbf{t}^{\mathbf{I}})$ системы (1) из позиции $\{\mathbf{t}^{\mathbf{I}}, \mathbf{x}^{\mathbf{I}}\}$ $(\mathbf{t}_S \le \mathbf{t}^{\mathbf{I}} < \mathbf{t}^{\mathbf{II}} \le \mathbf{t}_F)$ называется множество, состоящее из всевозможных состояний системы в момент $\mathbf{t}^{\mathbf{II}}$ на любых доступных управлениях **u=(·)**.

Трубкой достижимости, обозначаемой $\mathbf{R}(\mathbf{t}^{\mathbf{I}}, \mathbf{x}^{\mathbf{I}}, (\mathbf{t}^{\mathbf{I}}, \mathbf{t}^{\mathbf{II}}])$, системы (1) из позиции $\{\mathbf{t}^{\mathbf{I}}, \mathbf{x}^{\mathbf{I}}\}$ на полуотрезке $(\mathbf{t}^{\mathbf{I}}, \mathbf{t}^{\mathbf{II}}]$ $(\mathbf{t}_S \le \mathbf{t}^{\mathbf{I}} < \mathbf{t}^{\mathbf{II}} \le \mathbf{t}_F)$ будем называть объединение $\bigcup_{\mathbf{t}\in(\mathbf{t}^{\mathbf{I}},\mathbf{t}^{\mathbf{II}}]} \mathbf{R}(\mathbf{t}^{\mathbf{I}},\mathbf{x}^{\mathbf{I}},\mathbf{t})$.

Аналогично определяются множества и трубки достижимости из компакта $\mathbf{X}(\mathbf{t}^{\mathbf{I}})$, лежащего на гиперплоскости, пересекающей пространство позиций при $\mathbf{t} = \mathbf{t}^{\mathbf{I}} \in [\mathbf{t}_S,\mathbf{t}_F)$.

Целевым множеством **M** назовем компакт, лежащий на гиперплоскости, пересекающей пространство позиций в момент $\mathbf{t}_F$, на который требуется привести систему (1) при всевозможных доступных управлениях.

Множеством разрешимости (**M**-управляемости), обозначаемым $\mathbf{W}(\boldsymbol{\tau}, \mathbf{t}_F, \mathbf{M})$, для системы (1) в момент $\boldsymbol{\tau} \in [\mathbf{t}_S, \mathbf{t}_F)$ при заданном целевом множестве **M** называется множество, состоящее из всевозможных состояний в момент $\mathbf{t}=\boldsymbol{\tau}$, из которых система переводима на **M** при любых управлениях $\mathbf{u}(\cdot) \in \mathbf{U}_{\mathbf{program}}$.

Трубкой разрешимости, или трубкой **M**-управляемости, обозначаемой $\mathbf{W}([\mathbf{t}^{\mathbf{I}}, \mathbf{t}^{\mathbf{II}}), \mathbf{t}_F, \mathbf{M})$, системы (1) на полуотрезке $[\mathbf{t}^{\mathbf{I}}, \mathbf{t}^{\mathbf{II}})$ $(\mathbf{t}_S \le \mathbf{t}^{\mathbf{I}} < \mathbf{t}^{\mathbf{II}} \le \mathbf{t}_F)$ при заданном множестве **M** назовем объединение $\bigcup_{\mathbf{t}\in[\mathbf{t}^{\mathbf{I}},\mathbf{t}^{\mathbf{II}})} \mathbf{W}(\mathbf{t}, \mathbf{t}_F, \mathbf{M})$.

Поточечные фазовые ограничения

$$\mathbf{g}_{\mathbf{k}}\left(\mathbf{x}(\mathbf{t}), \left\{\mathbf{u}(\mathbf{t}) \wedge \mathbf{u}(\mathbf{t},\mathbf{x})\right\}, \mathbf{t}\right) \le \mathbf{0},\ \mathbf{t} \in [\mathbf{t}_S,\mathbf{t}_F],\ \mathbf{k} = \overline{\mathbf{1,r}},\ \mathbf{r} \le \mathbf{n},$$

являются дополнительными критериями качества управления, которое, если удовлетворяет ограничениям, называется допустимым. При фазовых ограничениях речь идет об условных множествах достижимости и разрешимости, для аппроксимации которых недостаточно отсечения частей соответствующих множеств системы без фазовых ограничений. Обозначим условные и безусловные множества одинаково, так как из контекста ясен смысл обозначений.

Относительно системы (1) с заданным целевым множеством **M**, множеством начальных состояний $\mathbf{X}(\mathbf{t}_S)$, имеющим непустое пересечение с $\mathbf{W}(\mathbf{t}_S, \mathbf{t}_F, \mathbf{M})$, рассматривается ЗОПзУ с целевым критерием $\mathbf{I}(\mathbf{u},\mathbf{x}) = \int_{\mathbf{t}_S}^{\mathbf{t}_F} \mathbf{z}(\mathbf{x},\mathbf{u},\mathbf{t})\mathbf{dt} \to \mathbf{inf}$.

Решением ЗОПзУ будем называть функцию $\mathbf{u}(\cdot) \in \mathbf{U}_{\mathbf{positional}}$, определяющую управление системой для каждой позиции **{t, x}** из трубки разрешимости.

Схема численной оптимизации позиционных управлений

Для реализации подхода необходимо ввести в рассмотрение понятия аппроксимации целевого множества, трубки разрешимости и аппроксимирующего позиционного управления.

Сечение трубки разрешимости – множество

разрешимости, является также множеством достижимости системы, получаемой из исходной при ее рассмотрении в обратном времени с целевым множеством как множеством начальных состояний.

Множества достижимости нелинейных систем могут быть невыпуклыми и несвязными, и поэтому для учета этих особенностей в работе [2] предложены алгоритмы, реализующие метод сечений для аппроксимации множеств, а для ряда тестовых примеров представлены результаты численных экспериментов, иллюстрирующих эффективность метода сечений.

На отрезке $[0, t_F]$, рассматриваемом в прямом времени, вводится сетка с шагом $\Delta t = \frac{t_F}{N}$: $0=t^0<t^1<\ldots<t^j<t^{j+1}<\ldots<t^{N-1}<t^N=t_F,\ j=\overline{0,N}$.

Вводится также сетка, узлы τ^r $(r=\overline{0,N})$ которой следуют по узлам t^j: $\tau^N=t^0,\ \ldots,\ \tau^r=t^{N-r},\ \ldots,\ \tau^0=t^N$. Для краткости вместо $W(t^j, t_F, M)$ будем писать $W[t^j]$.

Аппроксимацией (ограниченного) множества разрешимости $W[t^j]$ системы (1), рассматриваемого в момент $t^j \in [0, t_F)$, будем называть такое конечное множество $\widehat{W}[t^j]=\{x^i(t^j)\}$, $i=\overline{1,q(t^j)}$, что справедлива формула

$$\left(\overset{\circ}{\forall} x^i(t^j)\in \widehat{W}[t^j]\ \left(i\in\overline{1,q(t^j)}\right)x^i(t^j)\cap W[t^j]\neq\varnothing\right)\ \&$$

$$\left(\overset{\circ}{\forall} x(t^j)\in W[t^j]\ \exists x^i(t^j)\in\widehat{W}[t^j]\ \left(i\in\overline{1,q(t^j)}\right)\&\exists\varepsilon\downarrow 0:\right.$$

$$\left.\left\|x(t^j)-x^i(t^j)\right\|\le\varepsilon\right),$$

где $q(t^j)$ означает количество элементов во множестве $\widehat{W}[t^j]$, квантор $\overset{\circ}{\forall}$ имеет смысл «почти для всех».

Аппроксимацией (ограниченной) трубки разрешимости $W([0,t_F),t_F,M)$ системы, рассматриваемой на отрезке $[0, t_F]$, называется конечное множество

$$\widehat{W}([0,t_F),t_F,M)=\bigcup_{j=0}^{N-1}\widehat{W}[t^j]=$$

$$=\left\{\left\{x^i(t^j)\right\},\ i=\overline{1,q(t^j)},\ j=\overline{0,N-1}\right\}$$

с условием $0<\Delta t\le\delta\downarrow 0$.

В основу метода сечений [2] положена следующая идея построения контура множества достижимости системы в некоторый момент t^j:

1) находятся координаты параллелепипеда, всех граней которого изнутри касается множество достижимости – для этого решается серия ЗОПрУ с целью поиска экстремальных (по возможности в глобальном смысле) значений каждой фазовой переменной;

2) в границах параллелепипеда вводится сетка с разбиением по каждой координате;

3) в результате решения серии ЗОПрУ вычисляются экстремальные (по возможности все локальные) значения некоторой фазовой координаты при фиксированных значениях для всех остальных координат.

В контексте схемы оптимизации позиционного управления узлы, представляющие содержание множества $W[t^j]$, могут быть введены условно, так как при работе оптимизационного алгоритма в случае несвязности множества достижимости будут удалены такие элементы множества $\widehat{W}[t^j]$, которые не принадлежат $W[t^j]$.

Итак, для эффективной реализации схемы необходимо учитывать возможности несвязности, вырождения в многообразие меньшей размерности для множества достижимости. Для аппроксимации, скажем, трехмерного множества достижимости его двухмерные сечения необязательно строить также методом сечений: можно применить для упрощения расчетов, к примеру, метод опорных гиперплоскостей в предположении выпуклости этих плоских сечений.

Разработанная компьютерная программа позволяет строить аппроксимации множеств достижимости. В основе лежат программы для численной оптимизации программных управлений в системах с фазовыми ограничениями.

Рассмотрим пример аппроксимации контура невыпуклого множества разрешимости.

Рассматривается система, описывающая управление с помощью $p(t)$ плоским маятником в среде с неизвестной вязкостью $q(t)$ (управление второго игрока) на отрезке времени $[0, 2]$:

$$\dot{x}_1(t)=x_2(t),$$

$$\dot{x}_2(t)=-0.15q(t)x_2(t)-10.15\sin x_1(t)+p(t).$$

На управления наложены ограничения: $|p(t)|\le 10$, $q(t)\in[0,1]$, $t\in[0,2]$. Целевое множество $M=(0, 0)$. Положим функцию $q(t)=0.5$ и для построения контура множества $W(0, 2, M)$ рассмотрим систему в обратном времени, полагая за начальный момент $t=0$:

$$\dot{x}_1(t)=-x_2(t),$$

$$\dot{x}_2(t)=0.15q(t)x_2(t)+10.15\sin x_1(t)-p(t),$$

$t\in[0,2]$. На рисунке представлен результат работы алгоритма.

Основным отличием методов сечений и опорных гиперплоскостей от методов эллипсоидов и других является вычисление аппроксимации множества достижимости, исходя непосредственно из определения этого множества.

Для нахождения семейства оптимальных программных управлений, аппроксимирующих оптимальное позиционное управление на частичном временном отрезке $[t^j, t^{j+1}]$ ($j\in\overline{0,N-1}$), проводится решение серии ЗОПрУ с целевым критерием

$$\mathbf{I_j(u)=\int_{t^j}^{t^{j+1}} z(x(t),u(t),t)dt \to inf},\ \mathbf{I(u)=\sum_{j=0}^{N-1} I_j(u)},$$

относительно системы (1)–(7) при поточечных фазовых ограничениях и краевых условиях $\mathbf{x(t^j)\in \widehat{W}[t^j]}$, $\mathbf{x(t^{j+1})\in \widehat{W}[t^{j+1}]}$.

На отрезке $\mathbf{[t_S, t_F]}$ проводится вычисление оптимального позиционного управления последовательно, переходя от отрезка $\mathbf{[t^{N-1}, t^N]}$ к отрезку $\mathbf{[t^0, t^1]}$, на основе принципа оптимальности Беллмана.

Для каждой позиции $\mathbf{\{t^j, x^j(t^j)\}}$, $\mathbf{j\in \overline{0,N-1}}$, $\mathbf{i\in \overline{1,q(t^j)}}$, определяется программное управление для движения на текущем частичном временном отрезке $\mathbf{[t^j, t^{j+1}]}$. Тем самым проводится аппроксимация позиционного управления программными управлениями – функциями или параметрами. Для простоты изложения ограничимся случаем аппроксимации параметрами.

Условие $\mathbf{x(t^{j+1})\in \widehat{W}[t^{j+1}]}$ записывается посредством терминальных ограничений следующего вида: $\mathbf{x_i(t^j)-\bar{x}_i=0}$, где $\mathbf{\bar{x}}$ – заданный числовой вектор, $\mathbf{i\in \overline{1,q(t^{j+1})}}$, $\mathbf{j\in \overline{0,N-1}}$. Для учета таких терминальных ограничений могут быть использованы различные способы, включая негладкий штрафной функционал $\mathbf{\sum_i \beta\left|x_i(t^j)-\bar{x}_i\right|}$, $\mathbf{\beta\geq 1}$.

В узлах сетки $\mathbf{\widehat{M}}$ функция цены $\boldsymbol{\varphi}$ позиционного управления $\mathbf{u(t, x)}$ принимает лишь нулевые значения. Рассмотрим функцию

$$\mathbf{y_j(t,x^i(t^j),u^m)=\int_{t^j}^{t} z(x(\varsigma),u(\varsigma),\varsigma)d\varsigma},\ \mathbf{t\in[t^j,t^{j+1}]}.$$

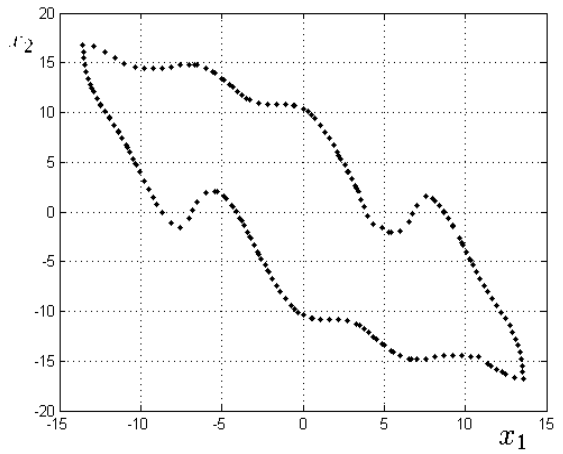
Функция цены управляющего параметра $\mathbf{u^m\in\{u^m\}_{m=0}^M}$ для позиции $\mathbf{\{t^{N-1},x^i(t^{N-1})\}}$: $\mathbf{\varphi(t^{N-1},u^m,x^i(t^{N-1},u^m))=y_{N-1}(t^N,x^i(t^{N-1}),u^m)}$, $\mathbf{i\in\overline{1,q(t^{N-1})}}$. Для позиции $\mathbf{\{t^j,x^i(t^j)\}}$ ($\mathbf{j\in\overline{0,N-1}}$, $\mathbf{i\in\overline{1,q(t^j)}}$) функция цены управления $\mathbf{u^m}$ определяется как сумма значения $\mathbf{y_j(t^{j+1},x^i(t^j),u^m)}$ и соответствующих значений функции цены на последующих частичных временных отрезках.

Функция Беллмана на множестве $\mathbf{W[t^j]}$ и ее аппроксимация как объединение всех наименьших значений функции цены по всем элементам множества $\mathbf{\widehat{W}[t^j]}$:

$$\mathbf{B(t^j,W[t^j],\{\bar{u}\})=\min_{\{u\}}\varphi(t^j,W[t^j],\{u\})\approx}$$

$$\mathbf{\approx B(t^j,\widehat{W}[t^j],\{\tilde{u}\})=\bigcup_{i=1}^{q(t^j)}\min_u\varphi(t^j,x^i(t^j),u)},$$

где $\mathbf{\{\tilde{u}\}}$ – объединение оптимальных программных управлений по всем узлам из $\mathbf{\widehat{W}[t^j]}$. Аналогично $\mathbf{\varphi^{max}(t^j,\widehat{W}[t^j],\{\tilde{\tilde{u}}\})=\bigcup_{i=1}^{q(t^j)}\max_u\varphi(t^j,x^i(t^j),u)}$.



Аппроксимация границы множества разрешимости

На отрезке $\mathbf{[t^j, t^{j+1}]}$ ($\mathbf{j\in\overline{0,N-1}}$) для каждого узла $\mathbf{x^i(t^j)\in\widehat{W}[t^j]}$ ($\mathbf{i\in\overline{1,q(t^j)}}$) проводится выбор из сгенерированного набора $\mathbf{\{u^m\}_{m=0}^M}$, во-первых, оптимального программного управления $\mathbf{\bar{u}^{i,j}}$ и, во-вторых, управления $\mathbf{\bar{\bar{u}}^{i,j}}$, на котором достигается максимальное значение функции цены, а также вычисление значений функций $\boldsymbol{\beta}$ и $\boldsymbol{\varphi}^{\mathbf{max}}$:

$$\mathbf{\bar{u}^{i,j}=\arg\min_{u^m}\left[y_j\left(t^{j+1},x^i(t^j),u^m\right)+\beta\left(t^{j+1},x^d(t^{j+1}),\bar{u}^{d,j+1}\right)\right]},$$

$$\mathbf{\beta\left(t^j,x^i(t^j),\bar{u}^{i,j}\right)=\min_{u^m}\left[y_j\left(t^{j+1},x^i(t^j),u^m\right)+\beta\left(t^{j+1},x^d(t^{j+1}),\bar{u}^{d,j+1}\right)\right]},$$

$$\mathbf{\bar{\bar{u}}^{i,j}=\arg\max_{u^m}\left[y_j\left(t^{j+1},x^i(t^j),u^m\right)+\varphi^{max}\left(t^{j+1},x^d(t^{j+1}),\bar{\bar{u}}^{d,j+1}\right)\right]},$$

$$\mathbf{\varphi^{max}\left(t^j,x^i(t^j),\bar{\bar{u}}^{i,j}\right)=\max_{u^m}\left[y_j\left(t^{j+1},x^i(t^j),u^m\right)+\varphi^{max}\left(t^{j+1},x^d(t^{j+1}),\bar{\bar{u}}^{d,j+1}\right)\right]},$$

$\mathbf{i\in\overline{1,q(t^j)}}$, $\mathbf{d\in\overline{1,q(t^{j+1})}}$, $\mathbf{j=\overline{0,N-1}}$.

Приведенная схема реализована программно, причем в случае аппроксимации позиционного управления семействами программных управляющих функций требуется привлечение программных средств для оптимизации программных управлений вместо простой схемы выбора значений параметра, изложенной ранее.

Случай аппроксимации параметрами более простой и менее трудоемкий, поэтому с точки зрения сравнительной эффективности можно считать его наиболее приемлемым.

По результатам работы алгоритмов, реализующих изложенную схему, строится композици-

онное программное управление, обеспечивающее кусочно-дифференцируемую траекторию, по которой производится перевод системы на целевое множество.

Если реализуется случай аппроксимации позиционного управления семействами параметров, то может понадобиться сглаживание получаемой траектории. С этой целью проводится приближение композиционного управления как кусочно-постоянной функции полиномиальной функции достаточно высокой степени.

Таким образом, итоговым этапом работы программной системы является применение результатов, насчитанных для всех аппроксимирующих сечений трубки разрешимости, для построения оптимального программного движения из любой позиции, взятой из аппроксимации этой трубки, на целевое множество.

Вопросы численной оптимизации программных управлений

Как уже отмечено, элементарной операцией в алгоритмах аппроксимации множеств разрешимости и оптимизации позиционного управления является ЗОПрУ, которая может оказаться достаточно трудноразрешимой. Эффективность решения серии ЗОПрУ зависит от уровня надежности (включая уровень автоматизации) программного обеспечения.

Авторами проведена реализация (на языке *Fortran*) ряда методов улучшения программных управлений. На языке *Maple* разработана программа автоматического вывода конструкций принципа максимума Понтрягина и его линеаризованной версии. Для учета концевых и поточечных фазовых ограничений реализованы методы гладких и недифференцируемых по Фреше штрафных функционалов.

Разработанная технология решения ЗОПзУ и аппроксимации множеств разрешимости не используют дифференциальное уравнение Гамильтона–Якоби–Беллмана, опираются непосредственно на определение множеств разрешимости и принцип оптимальности Беллмана, реализуемый на аппроксимации трубки разрешимости. В подходе элементарной операцией является ЗОПрУ, следовательно, эффективность его зависит от эффективности методов и многометодных схем решения ЗОПрУ.

Таким образом, аппарат аппроксимации траекторных трубок, а также решения ЗОПрУ и ЗОПзУ является единым с точки зрения достаточно полного исследования возможностей управления в нелинейных дифференциальных системах.

Литература

1. Моисеев Н.Н. Численные методы в теории оптимальных систем. М.: Наука, 1971.

2. Моржин О.В., Тятюшкин А.И. Алгоритм метода сечений и программные средства для построения множеств достижимости // Изв. РАН. Теория и системы управления. 2008. № 1. С. 5–11.

3. Тятюшкин А.И., Моржин О.В. Алгоритм численного синтеза оптимального управления // Автоматика и телемеханика. 2008. № 4.

4. Тятюшкин А.И., Моржин О.В. Конструктивные методы оптимизации управлений в нелинейных системах // Там же. 2009. № 4.

ПРЕДВАРИТЕЛЬНОЕ СООБЩЕНИЕ О ЯЗЫКЕ ИСПОЛНЯЕМЫХ ПРОГРАММНЫХ СПЕЦИФИКАЦИЙ

(Работа выполнена при поддержке Программы фундаментальных исследований Президиума РАН № 3 «Фундаментальные проблемы системного программирования»)

В.Б. Новосельцев, д.ф.-м.н. (ИПС РАН, г. Переславль-Залесский, V.Novoseltsev@acm.org);
Ф.А. Новиков, к.ф.-м.н. (Институт прикладной астрономии РАН, г. Санкт-Петербург, fedornovikov@rambler.ru)

Preliminary Report on the Language PSI

Novoseltsev Vitaliy B., Ph.D. *(PSI RAS, V.Novoseltsev@acm.org);*
Novikov Fedor A., Ph.D. *(IAA RAS, St. Petersburg, fedornovikov@rambler.ru)*

Abstract. Domain specific language for program specification is proposed. The language allows to describe models of problem areas, put computational tasks and synthesize task solution programs. Program synthesis is based upon peculiarly organized logic calculus.

Keywords: domain specific language, automatic program synthesis, formal specifications.

В статье рассматривается предметно-ориентируемый язык исполняемых программных спецификаций, позволяющий описывать модели формализуемых предметных областей, ставить на них вычислительные задачи и синтезировать программы решения задач на основе логического вывода в специальном классе исчислений.

Ключевые слова: *предметно-ориентируемый язык, автоматический синтез программ, формальные спецификации.*

Автоматический синтез программ по спецификациям – мечта методологов программирова-

ния. Несмотря на заметные успехи в области теоретической информатики, доля промышленных проектов по разработке ПО, основанных на автоматической генерации кода и доказательном программировании, остается незначительной. Наряду с яркими примерами в журнальных публикациях есть и многочисленные отрицательные отзывы практикующих программистов.

Многие полагают, и авторы разделяют это мнение, что одним из ключевых факторов, влияющих на распространение автоматического синтеза программ, является использование формальных методов построения моделей конкретных предметных областей. Введение в широкую практику программирования предварительных математических моделей предметных областей (именно моделей как определяющих конструкций) является признаком того, что программирование действительно становится инженерной областью деятельности.

В последние годы наиболее многообещающим в этой области представляется распространение идей порождающего программирования. Разумеется, порождающее программирование – это исторически далеко не первая попытка опереться в программировании на математические модели и методы, понятные всем. Необходимо указать три ранние работы, идеи которых – опора на рекурсивные декларативные определения, использование формализованной модели предметной области и дедуктивный синтез программ из крупных блоков, а не из операторов языка программирования – оказали значительное влияние на последующее развитие в данной области [1–3].

Практика показывает, что «сильный» пользователь, досконально знающий предметную область, имея в распоряжении предметно-ориентированный язык (*domain specific language*), опирающийся на развитый пакет прикладных программ, зачастую быстрее находит лучшие и более дешевые программные решения, чем команда профессиональных программистов. В настоящее время накоплен значительный запас пакетов прикладных программ для самых разных предметных областей, методы создания предметно-ориентированных языков (как текстовых, так и графических) достаточно развиты. Таким образом, ключевым вопросом является формализация знаний о предметной области.

Существует множество подходов к построению формализованных моделей предметных областей. В последнее время удалось значительно усовершенствовать формальный аппарат, на который опирается предлагаемый здесь подход [4, 5], что позволяет пойти дальше и предложить новое поколение средств спецификации моделей предметных областей.

Назначение и область применения языка

Предлагаемый язык *исполняемых программных спецификаций* (ИПС) предназначен для формализации предметных областей, определения предметно-ориентированных языков и быстрого создания прототипов программных приложений в формализуемых предметных областях (в английской нотации – *Program Specification Interpreter* – PSI).

В основу языка положены следующие принципы.

1. Ориентация на декларативную спецификацию предметной области в терминах формальной теории в специальном логическом исчислении, приспособленном для использования непрофессиональным программистом.

2. Симметричное описание структур (сущностей) и поведения (связей) конкретной задачи в данной предметной области: сущности определяются через связи, а связи через сущности.

3. Автоматический синтез схемы решения задачи с последующей компиляцией или интерпретацией схемы на основе доверительного программного фонда предметной области.

4. Открытое определение абстрактного синтаксиса языка (метамодели) и предоставление пользователю возможности расширения и изменения метамодели.

5. Определение нескольких вариантов конкретного синтаксиса, однозначно отображаемых в метамодель, в том числе графической нотации, порождающей грамматики, ограниченного естественного языка, схемы XML, системы распознающих автоматов.

Абстрактный синтаксис

Опишем метамодель языка ИПС в стиле, напоминающем Венский метод определения языков программирования. Абстрактная грамматика описывается с помощью следующих обозначений. Метаклассы обозначаются прописными буквами, а их атрибуты описываются в поясняющем тексте. Метасимвол «*****» означает итерацию. Альтернативы разделяются знаком «**|**».

Конструкция верхнего уровня называется моделью предметной области (метакласс **M –** от английского *model*). Модель агрегирует некоторое количество пакетов (метакласс **P** – *package*). Модель и пакет образуют пространство имен: различные однотипные сущности и различные однотипные именованные отношения должны иметь разные имена. Модель и пакет также имеют имена. Соответствующее грамматическое правило имеет вид: **M=P***.

Между пакетами определено отношение «непротиворечивое расширение». Если пакет **p2** непротиворечиво расширяет пакет **p1**, значит, все сущности и отношения **p2**, одноименные с сущностями и отношениями **p1**, добавляют свои свойства и составляющие к одноименным элементам **p1**,

если это можно сделать непротиворечивым образом, то есть не переопределяя элементы **p1** и не нарушая ограничения метамодели. (Используется следующее лексическое соглашение. Имя объекта, являющегося экземпляром некоторого метакласса, начинается с той же буквы, что и имя метакласса. Если объект является классом, то его имя начинается с прописной буквы, а если экземпляром, то со строчной. Таким образом, **p1** – имя конкретного пакета.) Непротиворечивое расширение играет ту же роль, что и наследование в объектно-ориентированных языках, но свободно от недостатков последнего. Кроме того, пакеты могут быть вложенными и образуют строгую композиционную иерархию.

Пакет определяет группу родственных понятий, которые образуют язык предметной области. В пакет композиционно вложены «отношения» (метакласс **R** – *relation*), которые делятся на три непересекающихся подтипа: эксплицитные, или явно заданные (метакласс **T** – *table*), имплицитные, или неявно заданные (метакласс **S** – *scheme*), и встроенные, или примитивные (метакласс **E** – *embedded*). Явно заданное отношение – это совокупность кортежей с данными (метакласс **D** – *data*). Встроенное отношение – это ссылка на внешнюю программную реализацию, которых может быть несколько (метакласс **I** – *implementation*). Самым характерным является имплицитное отношение, задаваемое набором функциональных зависимостей (метакласс **F** – *function*), постулирующих возможность вычислить значение одного атрибута – результата, если заданы значения некоторых других атрибутов – аргументов. Функциональную зависимость в моделях обозначим стрелкой «→», направленной от аргументов к результату. Соответствующее грамматическое правило имеет вид: **P = R ***

R = T | S | E.

Любое отношение, в частности схема, обязательно содержит определенное количество атрибутов (метакласс **A** – *attribute*) и может содержать некоторое количество вариантных частей (метакласс **V** – *variant*). Вариантные части являются дополнительными альтернативными частями отношения, выбираемыми с помощью условия варианта (метакласс **G** – *guard*). Вариантные части также могут содержать атрибуты и функциональные зависимости. Соответствующее грамматическое правило имеет вид:

S=A*F*V*

V=G A*F*

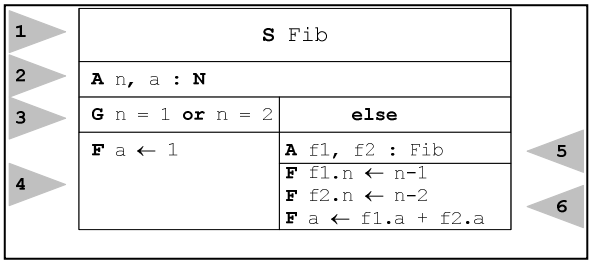
F=A*→A.

Атрибут (имплицитного) отношения может быть входным – встречается только в левых частях функциональных зависимостей, значит, его значение должно быть задано. Атрибут может быть выходным – встречается в правых частях функциональных зависимостей, значит, его значение может быть вычислено. Эта классификация лежит в основе базовой конструкции «задание» (метакласс **Q** – *question*). Задание ставится на некотором отношении, возможно, специально сконструированном на основе отношений предметной области для описания контекста конкретной задачи. В задании требуется построить программу, определяющую значения некоторых выходных атрибутов отношения по некоторым входным атрибутам. Соответствующее грамматическое правило имеет вид: **Q=A*?A*.**

Графическая нотация

Современная тенденция – использование для предметно-ориентированных языков графической нотации там, где это возможно. В большинстве случаев основу графического языка составляет метафора диаграммы графа: сущности изображаются вершинами, отношения – ребрами. У такого подхода множество неоспоримых плюсов, но есть и существенный недостаток: чтобы диаграмма читалась, в ней должно быть достаточно «воздуха».

В предлагаемом графическом языке основу составляет другая метафора – разлинованная страница, то есть разбиение отведенного поля диаграммы на части прямыми линиями. При этом общая картина не двух-, а трехмерная. Разлинованные страницы могут быть сложены в стопку и связаны друг с другом, например, прямоугольник на одной странице раскрывается в виде другой страницы. При таком подходе «воздух» совсем не используется, диаграмма – это плотно упакованная кристаллическая структура. Сущности вводятся текстовыми именами, а отношение композиции – вложенностью соответствующих блоков. В текстовой нотации применение отношения композиции, как обычно, передается точкой. Пример описания отношения изображен на рисунке (серые треугольники – не часть графической нотации, а пояснительные выноски).



В этом примере определяется отношение **Fib**(1), элементами которого являются номер (атрибут **n**) и соответствующее значение (атрибут **a**) числа Фибоначчи. Основные атрибуты **n** и **a** (2) – натуральные числа (**N** – встроенный тип). Отношение имеет две вариантные части (4), выбор которых производится по условию (3) (**else** – встроенный предикат). Причем вторая вариантная часть содержит два дополнительных атрибута **f1** и **f2** то-

го же типа **Fib** (5). Собственно отношение определяется имплицитно с помощью функциональных зависимостей (6). Подобные рекурсивные определения не только являются допустимыми, но и служат одним из основных выразительных средств языка ИПС. В данном случае для ясности указан метакласс каждой конструкции. Разумеется, на практике это делать необязательно, если метакласс восстанавливается из контекста.

Деривационная семантика

В качестве математического фундамента предлагаемого языка используется полный формализм, выразительность которого приближается к выразительности языков первого порядка – теория структурных моделей и связанное с ней исчисление **SR** [4]. Формализм структурных моделей является некоторым вариантом типизированных исчислений, при этом исчисление **SR** позволяет эффективно (не более чем с третьей степенью от длины исходного описания) устанавливать выводимость целевого утверждения.

Базовой конструкцией теории структурных моделей выступает *схема отношения*, которая содержит некоторое количество атрибутов, связанных функциональными зависимостями, а также может содержать определенное количество вариантных частей, обусловленных селекторами, зависящими от значений атрибутов, не входящих в вариантные части. Каждая вариантная часть, в свою очередь, может иметь некоторые атрибуты и функциональные зависимости. Среди атрибутов вариантных частей могут быть атрибуты, типом которых является само определяемое отношение. Таким образом, определение отношения может быть рекурсивным [5]. Это обеспечивает возможность синтеза рекурсивных программ определенного вида.

С помощью описания структурной модели для предметной области строится специальная (формальная) теория, в которой можно решать, в частности, задачу установления выводимости предложений соответствующего языка и генерации, реализующей вывод программы.

Постановка задачи в структурной модели носит непроцедурный характер – в ней указываются лишь исходные и искомые атрибуты некоторой схемы. Содержательно постановка задачи на модели есть задание на построение схемы алгоритма, реализующего требуемые вычисления.

Алгоритм синтеза схемы программы, включающий и стратегию поиска вывода в исчислении **SR**, обеспечивает синтез схемы программы за время, в худшем случае не превосходящее третьей степени длины описания модели [5]. Класс синтезируемых программ достаточно широк, он включает в себя линейные, ветвящиеся и рекурсивные конструкции, а также предусматривает распараллеливание.

Алгоритм может быть описан с использованием некоторого псевдокода, структуры управления которого традиционны для языков программирования высокого уровня. Используются следующие обозначения и соглашения:

- $\mathbf{C_k}$ **–** множество достижимых атрибутов исходной схемы;
- *по*-аксиома – *проблемно-ориентированная* аксиома из текущей схемы;
- вход в подсхему – наличие в $\mathbf{C_k}$ атрибутов, принадлежащих подсхеме;
- $\mathbf{A_0}$ и **S** берутся из формулировки задачи.

Описание алгоритма.

«Установить **i=0**, $\mathbf{C_0=A_0}$»;
«поднять флаг "*продолжать доказательство*"»;
«текущей схемой объявить **S**»;
while «поднят флаг "*продолжать доказательство*"» **do**
if «имеется *по*-аксиома текущей схемы, аргументы которой входят в C_i» **then**
«применить правило композиции и, если возможно, правило ветвления или правило распараллеливания»;
«определить $\mathbf{C_{i+1}}$ с новыми атрибутами»;
«установить **i=i+1**»; «если определился вход в подсхему, запомнить ее»
else if «имеется подсхема, в которую определен вход» **then**
«объявить ее текущей схемой»
else if «текущая – рекурсивная схема» **then**
«применить правило введения рекурсии; установить **i=i+1**»;
else «объявить текущей внешнюю схему» **fi fi fi**;
if «текущая – исходная и нет *по*-аксиом, аргументы которых входят в $\mathbf{C_i}$» **then**
«опустить флаг "*продолжать доказательство*"» **fi**
od.

Доверительный программный фонд

Модель предметной области на языке ИПС опирается на некоторый программный фонд, накопленный в данной предметной области. Составляющими этого фонда могут быть отдельные программы, наборы данных, библиотеки функций, библиотеки типов и классов. По мнению авторов, очень важно обеспечить возможность использования любого нестандартного программного фонда, поскольку, во-первых, ПО, декларирующее соответствие какому-либо стандарту, встречается сравнительно редко и, во-вторых, такое соответствие невозможно проверить, а потому на него не стоит полагаться. Вместо этого описывается форма требуемого интерфейса (контракта), который должен обеспечивать используемый программный фонд.

В модели предметной области программный фонд представляется как первичные типы, первичные функции, первичные отношения.

Первичные типы мыслятся как множество значений и множество применимых к этим значениям операций. Значения первичного типа не обладают «самостью» (*identity*), в модели предмет-

ной области работа со значениями первичных типов осуществляется либо через атрибуты, содержащие эти значения, либо через изображения констант, обозначающих эти значения.

Значение данного первичного типа может иметь различные представления в памяти компьютера, а операция реализована разными методами – эти нюансы являются внутренним делом реализующего программного фонда и в модели не отображаются. Аналогично ответственность за обработку нештатных и исключительных ситуаций, подобных переполнению, потере значности, нехватке памяти и т.д., должен брать на себя доверительный программный фонд – в модели эти детали не учитываются.

Строки (первичный тип **Str**), натуральные числа с нулем (первичный тип **Nat**) и логические значения (первичный тип **Bool**) считаются встроенными и доступными при использовании любого программного фонда или даже без него.

Первичные функции – это операции предметной области. Они могут иметь входные параметры первичных типов и должны возвращать значение первичного типа. Первичные функции могут инкапсулировать сколь угодно сложные вычисления в предметной области: с точки зрения модели предметной области первичная функция – это атомарное непрерываемое гарантированно завершающееся вычисление, доставляющее декларированный контрактом результат.

Первичные отношения – это наборы данных предметной области. Они могут быть представлены реляционными таблицами, XML-файлами или каким-то другим способом. С точки зрения модели это не имеет значения при условии, что предоставлен итератор первичного отношения, позволяющий перебирать кортежи первичного отношения, если значения атрибутов этих отношений нужны для выполнения синтезированной программы.

Предлагаемый язык ИПС является продолжением и развитием идей синтеза программ на функциональных вычислительных моделях, предложенных Э.Х. Тыугу в 70-х годах прошлого века. Позже стали использовать термины «семантические вычислительные сети» и «концептуальное программирование». На основе этой идеи в 80-е годы было несколько десятков реализаций в разных организациях как в форме систем автоматического синтеза программ общего назначения, так и в форме конкретных предметно-ориентированных пакетов прикладных программ, управляющие модули которых использовали данную идею.

В настоящее время удалось значительно усилить используемый математический аппарат, превратив эмпирическую программистскую конструкцию в развитую теорию структурных моделей и соответствующее исчисление **SR** [4, 5], за счет чего область применения структурного синтеза программ существенно расширилась.

Литература

1. Кахро М.И., Калья А.П., Тыугу Э.Х. Инструментальная система программирования ЕС ЭВМ (ПРИЗ). М.: Финансы и статистика, 1981. 157 с.

2. Непейвода Н.Н. Об одном методе построения правильной программы из правильных подпрограмм // Программирование. 1979. № 1.

3. Бабаев И.О., Новиков Ф.А., Петрушина Т.И. Язык Декарт – входной язык системы СПОРА. М.: Финансы и статистика, 1981. С. 35–73. (сб. Прикладная информатика. № 1).

4. Новосельцев В.Б. Теория структурных функциональных моделей // Сибирский матем. журн. 2006. Т. 47, № 6. С. 1342–1354.

5. Новосельцев В.Б. Синтез параллельных рекурсивных программ в функциональных моделях // Программирование. 2007. № 5. С. 1–6.