Федеральное государственное автономное образовательное
учреждение высшего образования
«Национальный исследовательский университет
Высшая школа экономики»

На правах рукописи

Турунтаев Илья Сергеевич

Математическое и программное обеспечение интерактивной системы обучения с автоматической генерацией задач и специализированными алгоритмами конструирования дерева решения

Специальность 05.13.11 —
«Математическое и программное обеспечение вычислительных
машин, комплексов и компьютерных сетей»

Диссертация на соискание учёной степени
кандидата технических наук

Научный руководитель:
доктор физ.-мат. наук, профессор
Данилов Владимир Григорьевич

Москва — 2018

Оглавление

Введение

Образовательное программное обеспечение

Последние несколько десятилетий ознаменованы поразительным по своей интенсивности ростом в области информационно-коммуникационных технологий (ИКТ). Современный уровень развития этой сферы и широкое распространение сети интернет не только сделали возможным решение задач, считавшихся неразрешимыми или излишне сложными еще недавно, но и позволили предлагать качественно новые подходы к решению различных проблем, так или иначе решенных прежде. Особый интерес представляет вопрос об организации и поддержке учебного процесса. В этой области, как и во многих других, с использованием современных информационных технологий возможно провести серьезные улучшения.

Интерес к этому вопросу возник практически сразу с появлением пользовательских песональных компьютеров (ПК). Первые попытки приложения достижений в области ИКТ к организации процесса обучения заключались в составлении специальных обучающих компьютерных программ. Такие программы зачастую представляли собой базу заданий (с ответами) с некоторым интерфейсом доступа к этим заданиям и взаимодействия с ними. Способов взаимодействия пользователя с заданиями при их решении здесь может быть, по большому счету, два: выбор одного из нескольких вариантов ответов или возможность составить ответ из предлагаемого набора объектов. С точки зрения же организации интерфейса в целом возможен целый спектр вариантов. В простейшем случае такое приложение предоставляет последовательный доступ к задачам (в выбранной тематике) с текстовым условием и ответом, продвижение осуществляется по мере успешного решения. Известны, однако, и программы, организованные по принципу компьютерной игры. Наибольшее распространение в свое время получили так называемые развивающие игры. Такие продукты зачастую представляют собой полноценную компьютерную игру с собственной сюжетной линией, с одной отличительной чертой — задания подразумевают приложение умственных способностей пользователя и имеют своей целью их развитие. Хорошо себя зарекомендовали продукты данного типа, нацеленные на аудиторию дошкольного и младшего школьного возраста. Для обучения в старших классах школы и позднее такого рода продукты уже мало применимы: затраты на производство таких игр попросту несоизмеримы с предполагаемой пользой, также под сомнение ставится возможность надлежащего покрытия исследуемой области. Тут следует, однако, отметить образовательные продукты, предназначенные для обслуживающего персонала объектов сложной структуры (например, нефтяной станции или кабины самолета), для которых активно разрабатываются эмуляторы рабочей среды. Такие эмуляторы (по крайней мере, некоторые из них) формально можно причислить к образовательным компьютерным программам типа «игра». За исключением такого рода особых ситуаций образовательные системы такого типа считаются неприменимыми в образовании.

Для своего времени первые обучающие программы были весьма существенным и полезным новшеством. Нередко такие программы снабжались некоторым справочным теоретическим материалом, примерами решения и т.п. Уже одно это качество — объединение практического и теоретического материала на одном устройстве в одной программе — в ряде случаев существенно улучшало процесс обучения и/или тренировки: сев за такую программу обучающийся в идеале имеет все необходимое под рукой, он перестает тратить время на поиск и вычленение требуемой информации. Другим немаловажным достоинством такого подхода является доступность. Так составить, опубликовать и выпустить на рынок сборник задач часто сложнее выпуска аналогичной программы, в особенности, когда речь идет о сборнике с малым временем «жизни». В частности по этой причине некоторые ВУЗы страны занимались выпуском программ-задачников (с возможностями проверки ответов и некоторым справочным материалом) для самостоятельной подготовки абитуриентов. Выпуск таких программ порой оказывался менее затратным, а распространение — более простым. В таких вопросах ВУЗ часто в большей степени заинтересован в пользе, которую способны принести соответствующие материалы абитуриентам, нежели в прибыли от их реализации.

Очевидны и определенные недостатки обучающих компьютерных программ:

- Невозможность модификации. Программы такого типа не могут быть дополнены или как-то изменены (лишь в новой версии).

- Ограниченность. Во-первых, такие программы способны покрыть лишь некоторую наперед заданную область знаний и лишь в той мере, в какой это подразумевается исходным заданием. Во-вторых, с определенного момента разработка такого продукта вынуждена придерживаться заданного курса, даже если вскроются какие-то его недостатки, ибо его кардинальное изменение потребует отбросить полученные результаты почти полностью.

- Существенные временные затраты. С учетом первых двух пунктов этот фактор становится также важным недостатком, так как невозможность модификации и ограниченность автоматически уменьшают время жизни продукта, с которым безусловно надлежит соотнести время разработки при оценке целесообразности всего проекта.

Однако, важнейшим результатом опыта разработки обучающих программ можно считать более четкое определение целей систем по организации и поддержке процесса обучения. Тема использования информационных технологий в обучении продолжает развиваться, и так зарождается область «электронного обучения» — e-Learning.

Системы электронного обучения

На сегодняшний день наблюдаются две основные тенденции развития в области e-Learning. Первый путь подразумевает максимальную автономизацию соответствующих продуктов. Такие системы либо полностью исключают наличие активного (с точки зрения воздействия на процесс) преподавателя, либо сводят его к минимуму. В любом случае здесь устраняется прямой контакт обучающегося с преподавателям, а образовательные курсы имеют максимально возможную автономию. Этот путь имеет историческое первенство. Таковы и описанные выше обучающие программы, такими же являются многие системы онлайн-обучения. Так, к примеру, в 2003 году появился Национальный Открытый Университет «ИНТУИТ»[1], предоставлявший возможности дистанционного обучения. Курс здесь состоит из набора печатных лекций, каждая из которых завершается тестом для контроля знаний по данной лекции, а в самом конце пользователя ждет крупный экзамен в виде теста по всему материалу курса. Проверка таких тестов происходит автоматически, вопросы тестов генерируются случайным выбором из множества заранее составленных вопросов. Таким образом, процесс обучения здесь проходит без какого-либо участия живого преподавателя — полная автономия. Такое положение вещей многим кажется весьма привлекательным: гибкий график, устранение личностного фактора, комфортные условия. В то же время ряд положений здесь подвержен серьезной критике. В первую очередь следует остановиться на тестах в качестве способа проверки качества усвоения

[1] Негосударственное образовательное частное учреждение «Национальный Открытый Университет «ИНТУИТ», `www.intuit.ru`

материала. Такая форма контроля является наиболее простой с точки зрения реализации и именно этим обстоятельством оправдывается ее повсеместное использование. Однако, для большинства дисциплин, как точных, так и гуманитарных, она является, мягко говоря, сомнительной. Можно ли, к примеру, установить, насколько хорошо студент овладел некоторым языком программирования, на основе результатов тестирования? Если даже принять сомнительную гипотезу о том, что посредством тестирования возможно оценить уровень усвоения фундаментальных основ искусства программирования, то все еще остается открытым вопрос о способности обучающегося применить полученные знания к решению прикладных задач, что в конечном итоге является целью изучения данной дисциплины. Невозможно научиться программированию, не написав ни строчки кода, равно как и контроль уровня знаний здесь не может быть осуществлен без практических задач. Аналогичным образом дело обстоит с математическими дисциплинами, где применимость тестирования также (если не в еще большей степени) сомнительна. Наиболее общая критика такой формы осуществления контроля заключается в том соображении, что ответ на вопрос теста может быть угадан или получен, исходя из общего здравого смысла, что, вообще говоря, не может служить показателем уровня знаний по предмету.

Подобных систем существует достаточно много, ИНТУИТ — лишь наиболее полная и популярная из них на территории нашей страны. Однако, разработки в области e-Learning, идущие по пути автономизации, зашли уже несколько дальше. Определенное распространение в последнее время получила идея популяризации и демократизации образования, под эгидой которой стали появляться сервисы для размещения онлайн-курсов с ограничениями по времени прохождения и контролем (самостоятельные задания, экзамены), производимым на стороне, предоставляющей курс. В 2012 году запущены два крупных проекта дистанционного обучения, равных которым на сегодняшний день нет. Речь идет о проектах Udacity[2] и Coursera[3]. Отличительной особенностью обеих систем является их структура: они представляют собой площадки для размещения курсов доверенными источниками, к которым предъявляются определенные требования. Здесь по-прежнему отсутствует прямое взаимодействие обучающихся и преподавателей, однако, последние принимают (могут принимать) активное участие в осуществлении контроля там, где этот процесс не может быть автоматизирован. По этой причине строго ограничивается время проведения курса и благодаря этому возможен контроль, адекватный изучаемому предмету.

Как уже говорилось, первый путь — это путь автономии образовательных продуктов от каких-либо официальных систем образования. Следует отметить, что путь этот обладает определенными достоинствами, но также имеет множество противников. Пожалуй, наиболее важным результатом проектов такого типа является доступность. Благодаря упоминавшимся выше проектам Coursera и Udacity множество великолепных, редких, полезных курсов от ведущих преподавателей стало доступно желающим по всему миру, вне зависимости от возраста, пола, профессии, физических возможностей и прочих подобных факторов. Сторонники данного подхода также указывают на все возрастающий спрос на образование, с которым высшие учебные заведения уже плохо справляются, а вскоре перестанут справляться вовсе [1]. Среди преимуществ видео-лекций также нередко называют их воспроизводимость, возможность контролировать скорость, а также то обстоятельство, что такие лекции тщательно готовятся и потому не могут получиться «неудачными». Противники же такого подхода настаивают на невозможности достаточно сконцентрироваться на предмете в условиях онлайн-обучения,

[2] Образовательная организация, проект массового открытого образования, возникший на базе учебной программы по информатике Стэнофрдского университета. Основана С. Труном, Д. Ставенсом и М. Сокольски. `www.udacity.com`

[3] Проект массового открытого образования, основанный профессорами Стэнфордского университета Э. Ыном и Д. Коллер. `www.coursera.org`

когда студент находится вне аудитории, а также на негативном эффекте отсутствия диалога. В ряде исследований (например, в [2]), однако, получены результаты, согласно которым большого различия в качестве усвоения материала между онлайн- и очным обучением нет. В данном исследовании предпочтение отдается иному типу образовательных систем, которые не стремятся изменить саму форму образования, но нацелены на совершенство отдельных аспектов имеющейся системы.

В противоположность первому пути, некоторые системы пошли по пути обеспечения поддержки процесса обучения в рамках учебных учреждений. Необходимо сразу обозначить: в данном случае речь идет скорее о потенциальной возможности внедрения такого рода продуктов в образовательный процесс, чем строгая продиктованность такой возможности характером системы; такие продукты зачастую могут быть использованы и автономно. Формально, основное отличие систем, описанных выше, от продуктов данного типа заключается в том, что последние предоставляют гибкий инструментарий, позволяющий пользователям органично встраивать эти продукты в свои собственные курсы и/или учебные программы. Исключительный интерес это направление представляет по той причине, что ориентация на возможность интеграции таких продуктов с имеющимися образовательными программами подразумевает их высокую степень интеллектуальности (в общебытовом смысле слова) и гибкости, что поднимает ряд интересных задач.

Среди систем второго типа широкое распространение в последнее время приобрели системы управления процессом обучения — Learning Management Systems (LMS). Такие проекты в большинстве своем специализируются на организации формальной стороны вопроса: ведение текущей успеваемости студентов, хранение и управление информацией о курсах, распространение учебного материала и тому подобное. В нашей стране LMS обретают все бо́льшую популярность и используются уже во многих ВУЗах. Аналогично в школах распространяются так называемые электронные дневники.

Однако, в таких системах достаточно малое внимание уделяется (если вообще уделяется) вопросам, касающимся непосредственно проведения занятий — этими вопросами занимаются продукты иного сорта, о них далее и пойдет речь.

Системы поддержки процесса обучения

Вопрос проведения занятий включает два аспекта — аспект преподнесения теории и аспект организации практических занятий, целью которых является разъяснение и тренировка методов, используемых в данной дисциплине. К вопросу о способе преподнесения теоретического материала имеется два подхода: согласно первому материал представляется в текстовом виде, второй подразумевает использование видео-лекций. Последний вариант многими почитается как предпочтительный и получил в наше время широкое распространение (по этому принципу организованы такие системы дистанционного обучения как Cousera, Udacity, edX и многие другие). Этот подход имеет ряд преимуществ перед первым, так как приближается по выразительности к «живым» лекциям. Однако, текстовый материал, будучи дополнен надлежащим образом мультимедиа-данными, получает определенное преимущество за счет возможности контроля скорости и способа преподнесения материала, тогда как видео-курсы навязывают определенную скорость и линейную схему развития лекции. По большому счету, способы преподнесения теории ограничиваются описанными двумя вариантами, и дальнейшие ухищрения сводятся к определению структуры лекции и, возможно, способов комбинирования этих вариантов. Иначе дело обстоит с организацией практических занятий.

С точки зрения практических занятий важнейшую роль играют тренировочные задания — специального вида задачи, разработанные в целях обучения для отработки и закрепления обучающимися полученных теоретических знаний. Эти задания играют ключевую роль в процессе обучения во многих дисциплинах и часто являются для студентов единственным средством приложения изучаемой теории к практике. В последнее время вопросам составления и обработки тренировочных задач в системах электронного обучения уделяется достаточно мало внимания. Как уже говорилось выше, многие системы избрали в качестве формы контроля тесты, задачи же для самостоятельной тренировки часто предоставляются исключительно в виде нескольких примеров, возможно, с ответами. Такое положение дел, однако, может быть улучшено. Если говорить о точных науках, то традиционно тренировочные задачи здесь предоставляются специализированной литературой и, порой, из личных «запасов» преподавателя, которые он формирует на основе собственного опыта. Такой подход обладает рядом недостатков. Во-первых, заметим, что первоочередная цель тренировочных задач состоит в том, чтобы выработать навык отыскания и применения изучаемых студентом методов и правил в конкретных задачах, научить решать любые задачи того же типа. Для достижения этой цели каждому студенту требуется, вообще говоря, какой-то собственный объем тренировочных задач. И объем этот зависит от целого ряда обстоятельств, который едва ли можно формализовать: влияние оказывают и индивидуальные психологические особенности, и исходные способности, и внешняя среда, и многое другое. Традиционный же подход при выработке, к примеру, домашних заданий исходит из понятия среднестатистического студента, объемы заданий зачастую устанавливаются для всех одинаковыми. В рамках такого подхода практически невозможна индивидуализация заданий, так как едва ли возможна разработка индивидуальных программ для каждого из обучающихся в группе. Более разумным критерием оценки достаточности объема нерешенных задач кажется стабильность получения успешного результата. Само по себе понятие стабильности здесь дано неформально и является достаточно расплывчатым. В качестве строгого критерия здесь можно выбрать, например, отношение верно решенных задач на заданную тематику к общему числу решавшихся при установлении некоторого нижнего порога для последнего.

По большому счету, данный вопрос связан с тем обстоятельством, что традиционный подход строится вокруг преподавателя и по определению не может в полной мере учитывать индивидуальные особенности каждого студента. Исправление этого недостатка — одна из важнейших задач систем электронного обучения. На этом заостряется внимание в [3], автор настаивает на том, что в разработке таких систем необходимо задаваться вопросом об индивидуальных характеристиках обучающихся с тем, чтобы система образовывала максимально удобную для обучения каждого среду. Необходимо учитывать индивидуальные особенности и способности каждого студента, и системы электронного обучения могут и должны развиваться также и в этом направлении.

Другим недостатком традиционного подхода является относительно малое число однотипных заданий, предоставляемых источниками. Довольно логично попытаться снабдить каждого студента своим уникальным набором задач, однако, в рамках традиционного подхода это сделать достаточно проблематично, так как для этого потребуется изрядное количество источников, а также придется потратить много времени на поиск задач, удовлетворяющих в надлежащей мере требованию однотипности.

Есть и другие недостатки. Так, осуществление самостоятельного контроля (среди студентов) затрудняется недостаточностью информации об ошибке, когда таковая возникает, не говоря уже о том, что лишнее время тратится на простой поиск ответа для сверки, даже если список ответов представлен в конце задачника. Также во многих случаях найденная и исправленная ошибка не равноценна правильно решенной задаче — это обстоятельство игнорируется в рамках традиционного подхода. Эти и прочие недостатки могут быть так или иначе устранены, решение этого вопроса и составляет по сути предмет данного исследования. Улучшить положение вещей возможно, предоставив способ автоматизации и упрощения процесса составления, распространения и проверки тренировочных задач, а также сопроводительного теоретического материала. Некоторым образом эти идеи были воплощены компанией Waterloo Maple Inc. в продукте Maple T.A.

Maple T.A.

Задача автоматизации проверки задач, ответ к которым так или иначе выражается в математической нотации, напрямую связана с символьными вычислениями. По этой причине серьезную роль в этом вопросе сыграло развитие систем компьютерной алгебры (СКА). Современные системы компьютерной алгебры обычно реализуют операции упрощения выражений, символьные и численные подстановки, вычисление пределов, символьное дифференцирование и многие другие — словом, практически весь математический аппарат, необходимый для решения задачи автоматизации процесса разработки и сопровождения тренировочных задач. Проект Maple T.A. разработан на основе известной СКА того же производителя — Maple. Maple T.A. является прежде всего системой для проведения различного рода проверочных работ по математическим дисциплинам. Данная система позволяет создавать задачи, сопровождаемые всевозможными текстовыми описаниями, организовывать их в курсы и предоставлять студентам. При этом в определении математических выражений допускается использование полной функциональности системы компьютерной алгебры Maple. Это, в свою очередь, позволяет описывать довольно сложные задачи и использовать вычислительные возможности Maple для рандомизации параметров задач, подсчета ответа и так далее. Таким образом, данная система предоставляет мощный инструмент, с помощью которого возможно устранить ряд негативных эффектов традиционного подхода. Так, благодаря возможностям рандомизации параметров задачи сильно упрощается процесс создания однотипных задач, причем породить их можно великое множество, описав лишь один общий шаблон. Это позволяет решить проблемы недостатка задач в традиционных источников и сильно упрощает процедуру составления персонализированных заданий. Далее, в системе реализовано порядка девяти различных типов ответов, среди них — ответ в виде математической формулы. Свои варианты ответов на такие вопросы пользователи-студенты вводят в простейшей машинной математической нотации. В документации к продукту заявлено, что такие ответы обрабатываются автоматически и система способна правильно сравнить пользовательский ответ с истинным (то есть, заданным составителем задачи).

В разработке пользовательского интерфейса Waterloo Maple Inc., очевидно, стремились к максимальной дружественности и простоте. По этой причине основное поле описания задачи имеет вид WYSIWYG[4] редактора со стандартными инструментами форматирования текста и дополнительными инструментами для задания математических выражений и полей ввода ответа. Дополнительный алгоритмический раздел позволяет использовать язык Maple для более тонкого описания задачи. Этот раздел также снабжен рядом панелей-помощников для упрощения взаимодействия пользователя с языком. Весь целиком раздел описания задачи (оригинальное название — Question Designer) довольно прост в использовании и позволяет разрабатывать действительно мощные и интересные задания.

Система Maple T.A. сделала возможным решение ряда вопросов, поднятых выше. Однако, и она не лишена определенных недостатков. Первый — и весьма существенный — недостаток заключается в том, что это коммерческий продукт и он является принципиально платным. Уже самый этот факт наталкивает на мысль поиска или разработки бесплатной альтернативы. Причем проблема здесь заключается не только в самой необходимости покупки как таковой, но также и в том обстоятельстве, что развитие такой системы

[4] Аббревиатура от «What You See Is What You Get». Используется для обозначения типа текстовых редакторов (чаще всего — онлайн html-редакторов), в которых пользователь работает сразу с тем представлением текста и графики, какое он получит в результате работы, а не с исходным кодом или разметкой.

всегда будет в той или иной мере подчинено условиям коммерческой выгоды, что порой идет вразрез с требованиями нормального прогресса продукта, а также означает сокрытие используемых алгоритмов концептуальных решений от широкой общественности. Следующее свойство Maple T.A., вообще говоря, не является полностью недостатком — это вопрос спорный и может стать предметом для серьезных дебатов. Речь идет об использовании языка Maple в системе. С одной стороны, этот язык определяет выразительную мощность описаний задач. С другой же стороны, он в данном случае является избыточным. В силу своего исходного предназначения язык Maple обладает выразительностью, позволяющей с его помощью описывать довольно сложные алгоритмы и проводить серьезные вычисления. В то же время составление тренировочной задачи должно быть максимально упрощено, этот процесс не должен отнимать у составителя слишком много времени и сил, ведь, по большому счету, основной целью такого рода системы является упрощение процессов составления, распространения и проверки тренировочных задач за счет экономии времени преподавателей и студентов на рутинных операциях. С практической точки зрения избыточность языка здесь означает, с одной стороны, его излишнюю синтаксическую сложность, а с другой — чрезмерную свободу пользователя. Первое обстоятельство негативно сказывается на требованиях к уровню познания пользователей-авторов задач. Довольно важно в вопросе разработки системы электронного обучения такого типа стараться удержать наиболее низкий порог входа, то есть, предъявлять минимальные требования к познаниям пользователей и сводить к минимуму необходимость дополнительного обучения правилам пользования системой. Второй аспект — чрезмерная свобода пользователей — может служить потенциальным источником угроз и непредвиденных ошибок. Угрозы могут возникнуть в результате включения вредоносного кода — по злому умыслу или по ошибке — в описание задачи. Такой код может теоретически навредить как серверу системы, так и ее пользователям.

Причин же возникновения непредвиденных ошибок может быть множество: недостаточное знание языка, чисто алгоритмические ошибки, всевозможные переполнения и многое другое. В лучшем случае такие ошибки приведут к неправильной работе задачи. Проблемы угроз и ошибок ведут к необходимости строгого контроля пользовательского кода и излишняя выразительность языка Maple сильно усложняет эту задачу.

Также специфика языка Maple (как, впрочем, и большинства языков такого рода) плохо подходит для использования в рассматриваемом контексте по причине его «ориентированности на вычисление». В задаче описания задачи (ее постановки и хода решения) чрезвычайно полезным является возможность вычислять выражение лишь по мере необходимости. Точнее говоря, зачастую при составлении задач вычисления требуются лишь в описании хода решения и в ответе, тогда как в постановке лучше оставить выражения без вычислений. В таком случае удобно иметь возможность управлять вычислением выражений в описании задачи, чего сложно достичь при использовании языка Maple, так как такая функциональность в него не заложена.

Следующая претензия к проекту Maple T.A. касается способа организации проверки истинности пользовательского ответа типа «математическая формула». Этот вопрос является чрезвычайно важным для системы такого рода. Как уже упоминалось выше, разработчики системы заявляют возможность автоматической проверки пользовательского ответа, заданного в виде математической формулы в простой машинной математической нотации. Причем утверждается, что ответ пользователя-обучающегося не обязан синтаксически совпадать с истинным ответом, то есть, с ответом, заданным автором задачи. Такое утверждение, в свою очередь, звучит несколько сомнительно, так как по сути оно означает возможность автоматического установления семантической эквивалентности двух математических формул. Эта задача полностью эквивалентна следующей: имеется математическое выражение E, требуется установить, является ли оно тождественно равным нулю, то есть верно ли $E \equiv 0$. Известно, что эта задача, вообще говоря, является неразрешимой (соответствующий результат носит название теоремы Ричардсона [4]). Этот момент в документации не описан (по крайней мере, в процессе ее исследования упоминаний об этой проблеме выявлено не было), а значит, в системе таится потенциальная угроза неправильной реакции на пользовательский запрос, причем направлена она исключительно против студента. Данная проблема ставит серьезную преграду в разработке систем электронного обучения с поддержкой ответов, задаваемых в математической нотации, и требует пристального внимания.

В системе Maple T.A. решен ряд принципиальных вопросов разработки интерактивной системы онлайн обучения с поддержкой различных форм задач и она является в своем роде уникальной. Однако, указанные ее недостатки обосновывают целесообразность разработки некоторой альтернативы, в которой будут так или иначе решены проблемы этой системы. Следует отметить, что часть этих проблем едва ли может быть когда-либо разрешена в рамках самой системы Maple T.A., так как они коренятся глубоко в идеологии этой системы, их полное устранение требует изменений на самом базовом уровне, то есть, по большому счету, разработку с нуля. Целью данного исследования является разработка интерактивной системы обучения, ориентированной в первую очередь на поддержку практических занятий по дисциплинам, использующим математический аппарат, и в которой особый упор будет сделан на решение описанных выше проблем традиционного подхода к организации тренировочных задач и исправление ошибок существующих систем такого типа. Данная разработка во многом вдохновляется опытом компании Maple, а система Maple T.A. по некоторым аспектам может рассматриваться в качестве своеобразного ориентира.

В основе разрабатываемой системы лежит идея автоматизации процесса составления, распространения и поддержки (проверки ответов, предложение аналогичных примеров) тренировочных задач.

Важную роль здесь играет следующее соображение. Тренировочные задачи предназначены для применения в процессе обучения с целью отработки некоторого набора специфических методов, правил и т.п.. Таким образом, если мы зафиксируем некоторый набор методов и правил для закрепления, то соответствующие тренировочные задачи будут подчинены некой общей структуре. Это, в свою очередь, позволяет сделать предположение о том, что для таких задач можно привести общее описание — своего рода шаблон, некоторым образом параметризованный, по которому можно получить конкретные задачи на отработку соответствующих приемов. Само собой, едва ли возможно для произвольной области изобрести такой конструктивный шаблон, описанный на некотором языке, который бы покрыл все возможные тренировочные задачи в данной области. Речь в данном случае идет о возможности описывать весьма большие множества тренировочных задач на заданную тематику с помощью относительно небольшого числа общих шаблонов. Реализация данной идеи подразумевает разработку схемы описания таких шаблонов и алгоритма генерации конкретных задач, а также алгоритма проверки ответов — эти задачи и решаются в разрабатываемой интерактивной системе обучения. Эта система призвана устранить ряд проблем традиционного подхода и достигает этого благодаря идее шаблонизации тренировочных задач. Проверка пользовательского ответа к задаче подразумевает сравнение его с истинным и вынесение вердикта относительно его корректности: задача решена верно или нет. Поэтому система позволяет контролировать соотношение правильно решенных задач и задач, на которых испытуемый потерпел неудачу. Это, в свою очередь, позволяет решить заявленную ранее проблему индивидуализации объема заданий: в качестве меры может выступать не общее количество задач, но количество правильно решенных, или же соотношение успехов к неуспехам, или нечто более сложное (в данном контексте правильно решенной считается лишь та задача, для которой верный ответ был получен сразу, то есть успешно прошла проверка ответа системой). Проблема персонализации заданий решается организацией генерации конкретных задач по заданному шаблону случайным образом. При таком подходе каждый отдельно взятый студент наверняка будет взаимодействовать со своим уникальным набором задач, удовлетворяющих требованию однотипности в той мере, какая установлена их общим шаблоном. Снабжение же таких задач описаниями хода решения и справочным материалом по предмету позволит сильно сократить время на поиск и устранение ошибки; сам факт ошибки при этом устанавливается практически моментально, благодаря организации автоматической проверки ответа.

Вопрос об организации автоматической проверки ответов требует особого внимания. Как уже упоминалось выше, эта задача приводит к проблеме установления семантической эквивалентности математических выражений, которая, вообще говоря, является неразрешимой. Вместе с тем, нельзя этот вопрос оставить без внимания, так как автоматическая проверка ответов — основополагающее свойство системы. Для обеспечения символьных вычислений в разрабатываемой системе разумно использовать систему компьютерной алгебры. Такие продукты разрабатываются именно с целью организации символьных вычислений, различных численных алгоритмов и прочих подобных вычислительных возможностей. Коль скоро соответствующая функциональность является непосредственной целью этих продуктов, от ее реализации следует ожидать наивысшего качества, по сравнению с прочими проектами, где такие необходимость решения такого рода задач возникает в качестве побочного эффекта. Так и в разрабатываемой системе электронного обучения вычислительные задачи являются побочным следствием проблемы организации тренировочных задач с автоматической проверкой ответа. Потому использование СКА в качестве вычислительного модуля системы оправдывается ожидаемым качеством реализации требуемых алгоритмов в используемой СКА, а также сильным сокращением временных затрат на организацию вычислений. При этом разумно также использовать систему компьютерной алгебры с открытым исходным кодом, так как это позволит в случае возникновения необходимости внести изменения или дополнения непосредственно в вычислительный модуль, что сильно выигрывает у альтернативного варианта, в рамках которого приходится перенаправлять некоторые вычисления на собственные модули.

Для некоторых СКА эффект неразрешимости задачи сравнения двух выражений выражается в неправильном поведении: два семантически эквивалентных выражения воспринимаются как различные. Большинство СКА в случае возникновения таких проблем не будет вообще выдавать никакого ответа, что соответствует ответу «не знаю», то есть невозможно установить ни равенство, ни неравенство. Так, например, система компьютерной алгебры SymPy версии 0.7.2 не может установить эквивалентность выражений 2^x и $e^{x \cdot ln(2)}$, хотя они, очевидно, полностью эквивалентны. Более естественный и сложный для разрешения различными СКА пример — два разных способа вычисления производной функции $\frac{1}{\sqrt{1-\frac{1}{x^2}}}$. Причем ответ этой СКА в таком случае соответствует именно состоянию «не знаю». С точки зрения обеспечения проверки пользовательских ответов этот эффект означает, что в некоторых случаях система неспособна наверняка установить корректность ответа. Однако, все же, необходимо принять какое-то решение относительно успеха пользователя в данной задаче. Здесь есть две тривиальные стратегии: никогда не засчитывать «неоднозначные» ответы или наоборот — всегда засчитывать. Первая стратегия плоха в первую очередь тем, что ущемляет права студента. Студент будет незаслуженно «наказан», причем у него даже не будет возможности оспорить решение. Это, в свою очередь, будет чрезвычайно негативно сказываться на авторитете такой системы среди студентов, чего никак нельзя допустить. Вторая стратегия с этой точки зрения более безобидна: никто не будет напрямую задет ошибкой системы, несправедливость, скорее всего, просто не будет замечена. Однако и этот подход едва ли может рассматриваться всерьез, так как его основное обоснование

— лишь благосклонность к студенту. Вместе с тем, в условиях, когда мы не можем достоверно установить истинность ответа, кажется наиболее разумным исходить из соображений «ошибки в пользу студента». В таком состоянии неопределенности надлежит провести дополнительную проверку, которая поможет повысить качество принимаемого решения на основании дополнительного сравнения выражений. Такая проверка позволит отсеять некоторые заведомо неправильные решения, что повысит вероятность правдоподобности положительного ответа (на вопрос о корректности пользовательского решения). Разработке соответствующего алгоритма и его обоснованию посвящена часть данного исследования.

Описание тренировочных задач

В обзоре проекта Maple T.A. были сформулированы некоторые соображения о недостатках использования языка системы компьютерной алгебры Maple в описаниях задач. Эти соображения приводят к необходимости разработки собственного языка описания задач в описываемом проекте интерактивной системы обучения. В соответствии с возлагаемыми на него задачами и с учетом описанных выше недостатков реализации Maple T.A. формулируются следующие требования к языку: язык должен предоставить функциональность, необходимую для описания задач по различным математическим дисциплинам, и, по возможности, ею ограничиваться; язык описания задач должен быть как можно более простым, интуитивно понятным, предъявлять минимальные требования к познаниям пользователя и сводить к минимум необходимость дополнительного обучения; язык должен предоставлять широкие возможности рандомизации динамических атрибутов задач, включая разнообразные формы генерации случайных чисел, генерацию функций из различных классов и тому подобное; надлежит строго контролировать возможности языка в целях предотвращения угроз и ошибок — как умышленных, так и случайных, — возникающих в результате его применения пользователями системы; выражения языка должны подразумевать возможность их легкого преобразования в презентативную математическую нотацию (например, TeX, MathML); вычисление выражений должно происходить лишь по мере такой необходимости, причем выражения должны всегда допускать представление как в вычисленной (упрощенной), так и в невычисленной (не упрощенной) формах; В первом пункте ограничение функциональности призвано предотвратить избыточность языка, что позволит максимально упростить его синтаксически и облегчит задачу контролирования возможностей. Также акцент в первом пункте делается на математические дисциплины, поскольку реализация такой функциональности позволит легко распространить возможности системы на другие дисциплины, использующие математический аппарат. В результате, с некоторыми дополнениями, возможно покрыть область точных наук.

Разработка языка, удовлетворяющего указанным требованиям позволит устранить недостатки Maple T.A., связанные с избранной там стратегией. Это, однако, не исключает необходимость разработки специализированного интерактивного конструктора задач, который должен стать основным средством составления тренировочных заданий и позволит максимально снизить необходимость использования языка описания задач напрямую. Все вместе — язык и конструктор — служит цели обеспечения наибольшего удобства для пользователей и наименьшего порога входа.

Умное образование

Многие авторы указывают на то, что в настоящее время мы целенаправленно движемся в сторону так называемых умных технологий. Само понятие «умный», применительно к технологическим решениям, является дискуссионным. В настоящее время продолжаются нет единого представления о том, какие технологии и продукты можно с полной уверенностью назвать «умными», а какие — нет. И тем не менее, многие исследователи соглашаются в том, какие ключевые особенности отличают умные технологии от прочих. Концепция умных технологий лишь совсем недавно была приложена к области образования. Некогда популярное направление электронного образования начало постепенно вытесняться идеями Умного образования, Умной аудитории и Умного университета. Ключевую роль здесь сыграло то соображение, что образовательная среда не должна быть просто наполнена разнообразными достижениями ИКТ — разнообразной техникой и продуктами, каждый из которых выполняет какую-то свою задачу, — но включать набор связанных специализированных компонентов, покрывающих различные аспекты образовательного процесса, нацеленных на увеличение эффективности и продуктивности обучения. Теме умных образовательных технологий и смежным вопросам посвящено множество работ и исследований. Здесь будет приведен обзор наиболее значимых работ по этой теме.

Многие авторы [5, 6, 7] указывают на чрезвычайно высокий показатель роста в сфере информационно-коммуникационных технологий (ИКТ) в течение последних нескольких десятков лет. Уровень развития ИКТ и, в частности, сети интернет, позволяет разрабатывать качественно новые подходы к способам получение, создания и распространения знаний. В. Тихомиров в [6] указывает, что в подобных обстоятельствах чрезвычайно важно, чтобы ВУЗы предоставляли своим студентам доступ к новейшим областям знания и современным технологиям, а также подстраивали свои образовательные программы, чтобы адекватно отвечать на современные запросы и требования к образованию. Основная цель умных технологий в области образования заключается как раз в создании средств, позволяющих должным образом ответить на этот вызов.

Согласно исследованиям [8, 9], термин «умный», применительно к различным технологиям, появился относительно недавно. М. Кокколи и др. в [8] утверждают, что человечество входит в эру «умного нечто», которая сейчас вытесняет эру технологий 2.0, частично собой заменившую ранее «е»-технологии. Согласно М. Кокколи, каждая смена таких технологических «эр» сопровождалась усовершенствованием соответствующих технологий и появлением новых направлений развития, новых требований. В настоящее время префикс «умный» призывает уделить особое внимание удобству и простоте использования, гибкости, мобильности и надежности [8].

Проецирование идей умных технологий на сферу образования привело к появлению таких концепций, как *умное образование* , *умный университет* , *умная аудитория* , *умное учебное пространство* и прочих. Новая область привлекла за последние годы внимание многих исследователей [7, 9, 10, 11]. В то же время — и это не вызывает удивления — у этих терминов до сих пор нет строгих определений. Р. Копер [11] видит умное учебное пространство как специализированное окружение, которое подстраивается под текущие нужды обучающегося и конкретный вид и функциональность которого зависят от текущего контекста. Р. Копер утверждает, что основная задача в разработке умного учебного пространства заключается в том, чтобы сделать обучение более быстрым и качественным. При этом он особо отмечает, что этого нельзя добиться попросту сконцентрировавшись на техническом оснащении аудиторий. Необходим комплексный подход, меняющий и вид, и структуру образовательного процесса. Аналогично, Г.-Дж. Хуанг [10] определяет умное учебное пространство как технологически оснащенное пространство, которое способно адаптироваться под текущие запросы обучающихся и предоставлять поддержку в нужный момент времени и в нужном месте *каждому* обучающемуся индивидуально. Дж. Спектор в поисках ответа на вопрос “Что делает учебное пространство умным?” приходит к выводу, что оно должно обладать широким набор средств планирования и возможностями воплощать разнообразные инновационные подходы, а также — включать какие-то элементы, нацеленные на стимулирование “вовлеченности, эффективности и продуктивности” [12]. З. Зу и соавторы в [7], со ссылкой на [13], указывает такие особенности умных образовательных пространств как персонализированность и предоставление возможности самообучения для студентов. Практически все исследователи указывают на необходимость непрерывного, повсеместного[5] обучения, предъявляя к системам умного обучения требование предоставлять поддержку и сопроводительное руководство в нужное время в нужном месте.

Можно видеть, что в целом понятие умного учебного пространства базируется на «студентоцентричной» парадигме в том смысле, что все средства и технологии должны быть нацелены на удовлетворение учебных потребностей каждого конкретного студента. Общепринятым является представление о том, что умное учебное пространство должно тем или иным образом собирать и анализировать персональные данные студентов в процессе обучения (например, текущий уровень студента в различных областях, различные численные показатели успеваемости, курсы пройденные и изучаемые в данный момент и многое другое), а также прочие персональные характеристики студентов для того, чтобы адаптировать на основе такого анализа собственное поведение, материалы и даже интерфейс так, чтобы предложить каждому из обучающихся наиболее полную поддержку, а также помочь преподавателям, предоставляя им по возможности наиболее целостную картину по каждому из учеников.

Пожалуй, наиболее общим и цельным понятием, возникающим в контексте применения умных технологий к области образования, является понятие *умного обучения*. Множественные обсуждения ведутся вокруг этого термина и того, как следует его определить, однако, понятного и строгого определения на данный момент нет. Многие авторы (например, [7, 13]) соглашаются в том, что под умным обучением следует понимать образовательный процесс с

[5] Оригинальный термин — ubiquitous learning — подразумевает целый набор техник, позволяющих обучающемуся учиться не только в классе, но и в бытовых ситуациях. Наиболее приверженные сторонники такого подхода нередко указывают на необходимость разработки технических средств, которые пронизывали бы практически все сферы жизни ученика, направляя и обучая его повсеместно.

активным использованием различных технических средств, осуществляющих его поддержку и сопровождение, но, в то же время, с фокусом на обучающихся, а не на технологиях как таковых. В своей работе [13] Т. Ким рассматривает умное обучение как парадигму, совмещающую идеи *повсеместного обучения* и *социального обучения*. Социальное обучение характеризуется использованием социальных сетей и разнообразных «умных» технических средств в целях образования. При этом, согласно [13], особое внимание должно быть уделено "человеку и наполнению, а не технологиям и приспособлениям". Г.-Дж. Хуанг описывает идею умного обучения как качественно новый подход к обучению, сочетающий в себе характеристики контекстно-зависимого повсеместного обучения и адаптивного обучения, а также предлагающий нововведения, которым не обладает ни один из перечисленных более ранних подходов. Среди таких нововведений — адаптивные тренировочные задания и совокупный анализ разнообразных персональных факторов и характеристик окружения. В ряде публикаций (например, [8]) можно встретить схожее понятие — *умное образование*[6]. В [8] это понятие определяется как "образование в умном учебном пространстве при поддержке умных технологий, с использованием умных инструментов и устройств"[7].

Для так называемой «е»-эры характерно повсеместное внедрение ИКТ в попытках получить какие-то улучшения. Идеи умного обучения заходят дальше. В контексте технологий в образовании эта новая парадигма отличается особым вниманием к уникальным особенностям образовательного процесса. Персональные параметры студента оказываются в центре внимания. Технологи, как таковые, безусловно, играют важную роль в системах умного обучения, однако являются лишь средством и не должны быть поставлены

[6] В англоязычной литературе терминам «умное обучение» и «умное образование», приводимым в данной работе, отвечают словосочетания «smart learning» и «smart education» соответственно.

[7] Перевод следующего определения из [8]: "education in a smart environment supported by smart technologies, making use of smart tools and smart devices" (англ.)

в центре парадигмы [7]. Авторы [14] замечают, что технологии в образовании должны поддерживать процесс обучения не как инструкторы, но как средства построения и отображения тех или иных знаний; студенты должны учиться *с* их помощью, но не *от* них.

Системы компьютерной алгебры в образовании

Приложению ИКТ к обучению дисциплинам, активно использующим математический аппарат, посвящено множество исследований. Есть несколько причин такого особого внимания именно к математическому образованию. Во-первых, математика, в отличие от многих других дисциплин, работает на достаточно высоком уровне абстракции. Среди обучающихся весьма распространена проблема недостатка визуальной репрезентации математических объектов, которая могла бы помочь понять их природу. В результате студенты испытывают проблемы в обучении, так как не могут корректно вписать изучаемые объекты в свою внутреннюю систему знаний [15]. С этой проблемой помогут справиться широкие возможности визуализации, доступные в наши дни — статическая, динамическая и интерактивная визуализация. Следует, однако, отметить, что и здесь могут возникать достаточно сложные задачи. Проблема визуализации вообще — и математических объектов в частности, — еще недавно решавшаяся в основном методом проб и ошибок локально для конкретных задач, сейчас стала предметом изучения *теории визуализации* и в последнее время все чаще затрагивается специалистами самых разных областей. А. Рубин [15] рассматривает наглядное графическое представление математических объектов как одну из наиболее многообещающих и полезных возможностей, способных усовершенствовать математическое образование. Второй причиной исключительной роли ИКТ в дисциплинах, использующих математический аппарат, является исторический фактор. Долгое время компьютеры использовались исключительно для выполнения математических расчетов — математиками и прочими исследователями, активно использующими математический аппарат. Это, с одной стороны, привело к ситуации, в которой многие начали смотреть на обучение точным наукам с точки зрения компьютерных технологий [15]. Иными словами, технологи вновь оказались в центре внимания, тогда как необходимо напротив рассматривать технологии с точки зрения образования. С другой же стороны, на данный момент существует уже великое множество математических пакетов и полноценных продуктов, предоставляющих функциональность, способную существенно улучшить процесс обучения различным математическим дисциплинам и другим дисциплинам из области точных наук. Ключ к успеху — фокус именно на обучении, с учетом всех специфик и требований этого процесса.

Возможно, наиболее многообещающими и популярными готовыми продуктами, которые могут быть успешно использованы в обучении математическим дисциплинам, являются *системы компьютерной алгебры* (СКА) — программные продукты, основным назначением которых является реализация символьных вычислений [16, 17], то есть, способность манипулировать математическими объектами (производить математические преобразования), заданными в символьном виде. Смежным, но в целом более общим, понятием является понятие *системы компьютерной математики* (СКМ) — прикладной программы, реализующей те или иные математические вычисления. Зачастую СКА рассматривают как вид СКМ [18]. Тем не менее, большинство современных СКА помимо непосредственно символьных вычислений предоставляют множество средств для выполнения численных расчетов и многие сопутствующие инструменты (например, для отображения результатов). Поэтому далее под СКА будем понимать систему, реализующую как символьные, так и численные расчеты. Дополнительные возможности и инструменты будут указываться отдельно.

Системы компьютерной алгебры активно используются исследователями самых разных специализаций, и также использовались в образовании [19, 20]. П. Дрейверс (P. Drijvers) [20] указывает следующие преимущества использования СКА в математическом образовании:

- *Горизонтальная математизация*[8] — преобразование каких-то ситуаций и проблем из реальной жизни в математические задачи и наоборот. Использование СКА позволяет уделить внимание формализации задач в математических терминах и интерпретации результатов.

- *Исследования и вертикальная математизация* — манипуляция с математическими объектами на высоком уровне абстракции. Подразумевает развитие в студентах способности самостоятельно выводить и заново открывать какие-то свойства и теоремы, анализируя результаты специально разработанных под эти нужды задач.

- *Гибкое внедрение различных представлений.* Большинство СКА предоставляют возможность быстро и просто переключаться между различными представлениями одного и того же математического объекта (например, формула, график, таблица).

В то же время, использование СКА в образовании напрямую обнаруживает определенные недостатки. Согласно [20] эти недостатки можно подразделить на следующие три группы:

- *Чрезмерная мощность.* СКА могут оказаться чересчур мощным средством для использования в образовательных целях: если компьютер способен самостоятельно произвести все необходимые вычисления, то зачем тратить собственное время на

[8] Англ. «horizontal mathematization»

изучение соответствующих техник? Если не предпринять надлежащих мер, от этого может пострадать мотивированность студентов.

- *Проблема «черного ящика»*. СКА ведет себя как черный ящик, скрывая методы и алгоритмы, используемые ею при вычислении. Более того, система не только не дает объяснений хода решения той или иной задачи, но и сам конечный результат может иметь неожиданную для человека форму, так как используемые при вычислениях алгоритмы далеко не всегда соответствуют человеческому ходу рассуждений, а наиболее оптимальная форма выражения с точки зрения СКА не всегда будет наиболее удобной для пользователя.

- *Специфические особенности СКА*[9]. Всякая СКА использует свой специальный язык для получения команд от пользователя — со своими особенностями, нюансами. Зачастую, в силу своих широких описательных возможностей, такие языки оказываются достаточно сложными и требуют дополнительного изучения. Изучение такого языка может выходить за рамки предмета, отнимать слишком много полезного времени в ущерб основной цели.

О некоторых других недостатках использования СКА напрямую в обучении пишет З. Лавица [19, 21]. В данных работах исследовался вопрос о том, какое влияние оказывает внедрение СКА в процесс обучения на образовательные программы. Автор обнаруживает, что многие преподаватели различных математических дисциплин относятся негативно к перспективе использования СКА в своих классах в процессе обучения. Основной причиной такого отношения называется необходимость серьезной перестройки курса, что многие преподаватели находят нецелесообразным. Среди других причин — необходимость дополнительного изучения техники работы в СКА

[9] В оригинальной работе [20] данный недостаток носит название «idiosyncrasy» — буквально, отличительная особенность, характерная черта, особенность поведения.

и проблемы оценки знаний: плохо понятно, как оценить настоящие знания студента при использовании СКА, когда он может применять любые автоматизированные алгоритмы [21]. А. Рубин [15] также отмечает опасность ненадлежащего использования: необходимо проконтролировать, чтобы студенты сначала научились выполнять базовые операции самостоятельно, без помощи СКА, прежде чем их можно допустить к использованию СКА или других математических пакетов.

Оценка уровня знаний

Важную роль при разработке программных продуктов, осуществляющих поддержку процесса обучения, играет реализации системы оценивания уровня знаний. В наиболее простых случаях, продукт может вообще не брать на себя обязанности по осуществлению проверок, а, следовательно, и по выставлению оценок. Очевидно, однако, что для представления наиболее полной картины успеваемости каждого студента, необходимо реализовать процедуры построения оценок его текущего уровня знаний. Вычисление такого рода численных характеристик является также необходимой базой для построения алгоритмов, осуществляющих автоматизированную помощь, основываясь на текущем состоянии студента. Многие исследователи [13, 7, 12, 22, 23] отмечают необходимость наличия у систем электронного/умного обучения модулей, ответственных за вычисление показателей текущего уровня знаний каждого пользователя-студента. К этому вопросу предлагалось много различных подходов. В общем случае данную проблему можно свести к вычислению набора элементарных понятий (свойств, правил, определений и тому подобного), которые обучающийся понимает в полной мере и которыми способен оперировать. В случае дисциплин из области точных наук это часто делается посредством специально подобранных задач, каждая из которых требует от испытуемого демонстрации некоторого набора навыков. Здесь возникают вопросы:

- Какие вопросы или задачи нужно задавать?
- В каком порядке?
- В каком случае мы считаем, что соответствующая тема достаточно изучена (необходимый навык приобретен)? С какой степенью достоверности?

При этом, как и всегда, важно по возможности сокращать время, затрачиваемое на подобные проверки, или хотя бы держать его в допустимых пределах, сохраняя эффективность процедуры. Очевидно, что для того, чтобы установить статус обучающегося в некоторой области знаний, вообще говоря, не обязательно задавать ему каждый возможный вопрос в этой области. К примеру, если известно, что некий студент успешно справляется с задачами на вычисление производной от суммы двух основных элементарных функций, это означает, что данный студент заведомо справится с задачами на вычисление табличных производных. Данный пример иллюстрирует следующее общее соображение: между различными задачами, покрывающими некоторую область знаний, имеет место отношение, с учетом которого можно предложить методы оценки уровня знаний, более эффективные, чем простой перебор. В этом ключе, весьма интересный и наиболее строго математически обоснованный подход предлагает так называемая «теория образовательных пространств»[10] разработанная Ж.-К. Фальманем (J.-C. Falmagne) и Ж.-П. Дуаньоном (J.-P. Doignon) [23, 24].

Теория образовательных пространств предлагает формальную математическую модель процесса обучения. Рассмотрим некоторую область знаний. Множество задач[11] (вопросов) в этой области, каждая из которых имеет правильный ответ, называется *доменом*.

[10] Здесь под *пространством* понимается математическая структура

[11] Для дальнейшего изложения важно отметить, что под задачей в данном случае может пониматься не какая-то конкретная задача, а целый класс вопросов и задач, объединенных единой концепцией, нацеленных на отработку одного и того же правила, метода и т.п.

Предметом изучения теории образовательных пространств являются математические структуры над такими доменами. *Состояние знаний* студента определяется как некоторое подмножество задач из домена таких, что студент способен решить любую из них и не способен (или это пока не известно) решить остальные задачи домена. Иначе говоря, множество задач домена, которые данный обучающийся способен решить, определяет его состояние[12]. Понятие состояния знаний является ключевым в данной теории. Оно представляет собой формализацию естественного состояния обучающегося в рассматриваемой области знаний. Множество состояний над заданным доменом формирует математическую структуру, именуемую здесь «структурой знаний». Формально, под «структурой знаний» понимается пара $(Q, \mathcal{K})$, где Q — домена, а $\mathcal{K}$ — семейство подмножеств Q (то есть, множество состояний) такое, что оно содержит хотя бы Q и $\varnothing$. Замкнутая относительно операции объединения *структура знаний* называется *пространством знаний*.

Структура знаний — строго определенный математический объект, формализующий понятия области знаний и возможных состояний студентов в этой области. Тем не менее, в таком виде эта формализация несколько бедна и не вполне пригодна для использования на практике. Для иллюстрации, рассмотрим гипотетического студента Сергея и структуру знаний $(Q, \mathcal{K})$. Пусть Сергей находится в некотором состоянии $K \in \mathcal{K}$, причем $K \neq Q$, что означает, что он может ответить на любой вопрос (или решить задачу) из множества K, но в домене Q есть также задачи, с которыми он не способен справиться. В определенный момент Сергей выбирает задачу $q \in Q \backslash K$ (задачу, которую он пока не может решить), изучает соответствующий материал и, наконец, успешно проходит проверку на способность решать задачу q. В каком теперь состоянии должен оказаться Сергей? В текущей постановке нет ровно никаких гарантий, что состояние $K \cup \{q\}$ (то есть, предыдущее

[12] Здесь и далее, если это не приводит к неоднозначности, будем под «состоянием» понимать «состояние знаний», как оно было определено выше.

состояние с добавленной к нему изученной задачей) принадлежит множеству состояний рассматриваемой структуры знаний. Авторами теории образовательных пространств было предложено [24] два специальных условия, улучшающих понятие структуры знаний:

- *Гладкость обучения.* Для любых двух состояний $K,L \in \mathcal{K}$ таких, что $K \subset L$, существует конечная последовательность вложенных состояний $K = K_0 \subset K_1 \subset \ldots \subset K_p = L$, где $K_j \in \mathcal{K}$ и $|K_{i+1} \backslash K_i| = 1$ для $\forall i = \overline{0,p-1}, j = \overline{0,p}$. Это условие означает, что, если некоторое состояние K включено в другое, более полное состояние L, то последнего можно достичь из первого, изучая по одной задаче/теме за раз.

- *Последовательность обучения.* Для любых двух состояний $K, L \in \mathcal{K}$, таких что $K \subset L$, и для любого $q \in Q$, если $K \cup \{q\} \in \mathcal{K}$, то и $L \cup \{q\} \in \mathcal{K}$. Сами авторы теории определяют это свойство следующим образом: «Большее знание не исключает возможности изучения чего-то нового» [13] [24].

Структура знаний, отвечающая этим двум требованиям, называется *образовательным пространством*. Такая структура имеет гораздо больше смысла с педагогической точки зрения, чем простая структура знаний. На самом деле, понятие образовательного пространства не является принципиально новым в математике и полностью эквивалентно понятию “*антиматроид*”. Этот объект достаточно хорошо изучен, а его свойства позволяют весьма успешно его использовать для формального представления процесса обучения и построения алгоритмов оценивания и предложения путей развития или ликвидации пробелов, основанных на процедурах вычисления (скрытого) состояния и построения путей обучения (определенного вида последовательности задач, изучение которых в конечном итоге приводит обучающегося в конечное состояние, покрывающее весь домен). Другое важное свойство образовательных

[13] Оригинал (англ.): «Learning more does not prevent learning something new»

пространств — связь с предпорядками — определяет удобный способ построения этих пространств на основе отношений, заданных на множестве задач. В [24] рассматриваются различные модификации образовательных пространств.

Задачи и цели исследования

Целью настоящего диссертационного исследования является разработка интерактивной системы обучения, осуществляющей поддержку практических занятий по различным дисциплинам из области точных наук. При этом особый акцент делается на необходимости сохранить традиционную форму обучения на множестве конкретных тренировочных задач. Разрабатываемая система должна отвечать современным требованиям развития и персонификации образования, выявленным во **введении**. Достижение указанной цели подразумевает решение следующих задач:

- разработка модели представления базовых описаний тренировочных задач, по которым возможно порождение конкретных реализаций, определяемых случайной подстановкой, описывающих условие, решение ответ конкретных задач;
- разработка средств для создания базовых описаний тренировочных задач и генератора задач, способного работать с такими описаниями;
- разработка средств сбора и анализа персональных данных обучающихся в рамках их деятельности на практических занятиях с целью построения на их основе персонализированных путей развития и устранения пробелов, автоматического оценивания текущего уровня знаний и предоставления более полной картины по каждому студенту преподавателям;
- разработка веб-приложения по поддержке практических занятий по математическим и прочим дисциплинам из области

точных наук, включающего средства разработки и генерации тренировочных задач, с предоставлением надлежащей инфраструктуры для удобного взаимодействия преподавателей и студентов, и реализующего указанные выше требования.

Глава 1

Тренировочные задачи

Понятие тренировочной задачи лежит в основе практических занятий по любым дисциплинам из области точных наук. В образовательном процессе по математическим дисциплинам тренировочные задачи играют ключевую роль, причем чрезвычайно важны их форма и содержание. Тренируясь на множестве специально подобранных примеров, обучающиеся вырабатывают специальные навыки работы с объектами изучения, учатся их анализировать и применять полученные теоретические знание на реальных примерах. Следует отметить, что подобные навыки особенно важны для инженеров различных специальностей, так как значительная часть их рабочего процесса состоит непосредственно в выявлении некоторых свойств исследуемого объекта и последующем решении типовых задач при проведении необходимых расчетов. Способность увидеть своего рода шаблоны в реальных задачах, разложить сложную задачу на набор типовых — все это необходимые навыки, приобретаемые исключительно за счет тренировок на специальным образом подобранных примерах, имитирующих реальные задачи и призванных воспитать в студенте соответствующие паттерны.

При обучении различным математическим дисциплинам возникает множество вычислительных задач, конечный ответ в которых формулируется в виде некоторого математического выражения (или нескольких математических выражений). Такие задачи составляют практическую базу обучения и играют исключительно важную роль. Как уже упоминалось выше, традиционный подход к организации практических занятий по математическим дисциплинам подразумевает уделение достаточно большого времени именно этой составляющей процесса, что негативно сказывается на другой стороне процесса — исследовательских, теоретических и доказательных задачах. Напомним, что под *традиционным подходом* к организации практических занятий здесь понимается практика, распространенная во всех ВУЗах в России, согласно которой основная часть аудиторных практических занятий посвящается разбору и коллективному решению тренировочных задач, самостоятельная работа студентов организуется с помощью назначаемых преподавателем домашних заданий, промежуточный контроль осуществляется в виде организуемых преподавателем контрольных работ. Основными источниками задач при таком подходе служат специализированная литература (сборники задач, учебники, методические пособия, теоретическая литература) и сам преподаватель (многие опытные преподаватели имеют самостоятельно разработанные материалы). Главным недостатком традиционного подхода являются чрезмерные временны́е затраты на механическую, не несущую практически никакой креативной составляющей работу. Со стороны преподавателей большое время тратится на поиск и составление задач для семинаров, составление домашних заданий, контрольных работ, проверку всех видов работ, поиск и анализ ошибок. Со стороны студентов затрудненным оказывается поиск ошибок, проверка ответов, анализ работ [25, 26]. Важным является также и то обстоятельство, что при таком подходе практически невозможно уделить достаточное внимание каждому студенту — проанализировать индивидуальные ошибки каждого, составить персонализированные планы работ. При составлении домашних и прочих заданий преподавателей вынужден руководствоваться представлением о «среднем» студенте, суммируя и усредняя показатели в группе, а также — временны́ми ограничениями. При этом каждый обучающийся имеет свои индивидуальные особенности, обуславливающие качество и скорость усвоения того или иного материала. Эти и многие другие характеристики обуславливают необходимость и востребованность персонификации образовательного процесса [27, 28]. Поиск и анализ собственных ошибок является весьма важной составляющей образовательного процесса, а потому надлежит стремиться сделать ее наиболее естественной, безболезненной и легкой (в смысле отсутствия сторонних преград). В рамках традиционного подхода возможности самоанализа и самокоррекции оказываются достаточно скудными: столкнувшись с проблемами при решении той или иной задачи, студент в лучшем случае располагает правильным ответом и некоторым набором литературы (или других источников), описывающим общие подходы к решению подобных задач. Всего этого часто оказывается недостаточно для быстрого и эффективного поиска и устранения конкретной ошибки и общего пробела в знаниях. Наличие сразу доступного правильного ответа и возможность коррекции решения одной и той же задаче приводит также к проблемам «подгонки» решения — ситуации, в которой обучающийся стремится получить заранее известный конечный ответ, модернизируя неверное решение. При этом теряется ключевая цель тренировки на задачах. Особенность навыка решения вычислительных задач заключается в умении получить правильный конечный результат, руководствуясь лишь исходными данными. При этом чрезвычайно важно, чтобы обучающийся прошел весь путь от постановки задачи до окончательного ответа полностью — без дополнительных подсказок, каких-либо посторонних маркеров верности избранного пути. Метод «подгонки» решения под известный ответ полностью извращает эту идею и в результате процесс тренировки не достигает поставленных целей. Здесь уместно также остановиться на концепции автоматизации тренировочных задач, основанной на тестировании с выбором правильного ответа из нескольких вариантов. Подобная форма, ввиду простоты реализации, достаточно широко распространена, однако, совершенно неприменима в области вычислительных тренировочных задач по математическим дисциплинам. Критика такой тестовой формы реализации тренировочных задач — сродни критике метода «подгонки». Отличие здесь заключается в том, что испытуемому требуется лишь выяснить, какой из ответов является верным. Это приводит к незавершенным решениям, той же «подгонке», а часто и к выбору ответа на основе здравого смысла. Все это недопустимо при обучении математическим дисциплинам, так как нарушает саму концепцию обучения на множестве специализированных тренировочных задач.

Проблемы, связанные с чрезмерными временны́ми затратами на механические действия и с невозможностью предложить достаточный уровень персонификации, могут быть в значительном объеме решены с помощью автоматизации процессов составления, распространения и обработки тренировочных задач. Освободив преподавателей от рутинной нагрузки, связанной с составлением большого числа конкретных однотипных тренировочных задач, проверкой ответов домашних и контрольных работ, самостоятельным ведением подробной статистики успеваемости, мы высвободим время, которое можно (и до́лжно) потратить на те аспекты обучения, которые действительно требуют непосредственного участия преподавателя и коллективной работы студентов в аудитории. Автоматизация проверки ответов и предложение конкретных шагов решений для тренировочных задач позволит студентам наиболее эффективно тратить свое время, упростит процесс поиска и исправления ошибок. Имея возможность порождать большое количество задач, снабженных всеми необходимыми атрибутами, мы получаем возможность вводить новые характеристики успеваемости, в автоматическом режиме предлагать пути устранения пробелов в знаниях, составлять подробные индивидуальные портреты обучающихся, демонстрирующие их сильные и слабые стороны.

1.1 Шаблоны тренировочных задач

Всякая тренировочная задача[1] имеет три неотъемлемых составляющих — постановку задачи, ход решения и ответ. В постановке задаче формулируются начальные условия и ставится цель. Ход решения описывает шаги, последовательное выполнение которых приводит от начальных условий к достижению поставленной цели — конечному ответу. Так, например, можно сформулировать задачу с условием, требующим вычислить производную заданной функции. Ход решения такой задачи будет состоять из описания последовательных преобразований исходного выражения с применением соответствующих правил, в результате которых получается правильный ответ. Следует отметить, что различных подходов к решению одной и той же задачи может быть много. Более того, нередко различные подходы приводят к синтаксически различным ответам, представляющим один и тот же математический объект. Также, один и тот же подход к решению некоторой задачи может быть сформулирован с разной степенью подробности. Таким образом, под ходом решения задачи мы здесь будем понимать описание некоторого подхода к решению, сформулированное с достаточной степенью подробности.

Любая тренировочная задача нацелена на отработку некоторого набора навыков. В таких задачах формулируется искусственно созданная проблема, решение которой основано на применении соответствующих техник. При этом можно заметить, что задачи, посвященные отработке одного и того же набора навыков, имеют схожую структуру. Возьмем в качестве простого примера задачи на отработку правила дифференцирования композиции — типичные задачи из начального курса математического анализа. В зависимости от уровня сложности такая задача может состоять в вычислении производной от композиции двух, трех или более функций. Целью

[1] Здесь и далее повествование ограничивается рассмотрением вычислительных тренировочных задач, возникающих в процессе обучения математическим дисциплинам, с ответом в виде математического выражения.

всех этих задач будет отработка правила дифференцирования композиции нескольких функций. Условие для подобной задачи можно сформулировать следующим образом:

Вычислите производную функции $f(g(x))$

Здесь f и g — какие-либо функции одной переменной (например, основные элементарные функции[2]). По данному общему примеру можно построить великое множество конкретных задач на отработку указанного правила, заменив f и g на какие-то конкретные функции. Этот пример наглядно иллюстрирует идею шаблонизации тренировочных задач. Задачи, посвященные отработке одного и того же набор навыков (правил, техник и т.п.), имеют общую структуру. Таким образом, для заданного набора однотипных задач можно сформулировать базовое описание — общий *шаблон*, определяющий единую структуру всех задач данного класса (постановку, ход решения и ответ) и определяющий способ порождения конкретных примеров.

Конкретные представители некоторого класса тренировочных задач имеют общую структуру и отличаются конкретными значениями некоторого набора специальных выражений — *динамических параметров*. *Шаблон тренировочной задачи* представляет собой описание структуры класса однотипных тренировочных задач — постановки, хода решения и ответа, — определенное с точностью до набора динамических параметров, каждый из которых принимает значения в определенном множестве значений. В предложенном выше примере f и g играют роль динамических параметров. Указав множества значений этих параметров (например, множество основных элементарных функций), мы получим строгое описание шаблона условия тренировочных задач на отработку правила дифференцирования композиции. Такое описание допускает порождение большого числа конкретных тренировочных задач на эту тему.

[2] Степенная (с любым действительным показателем), показательная, логарифмическая, тригонометрические, обратные тригонометрические функции [29]

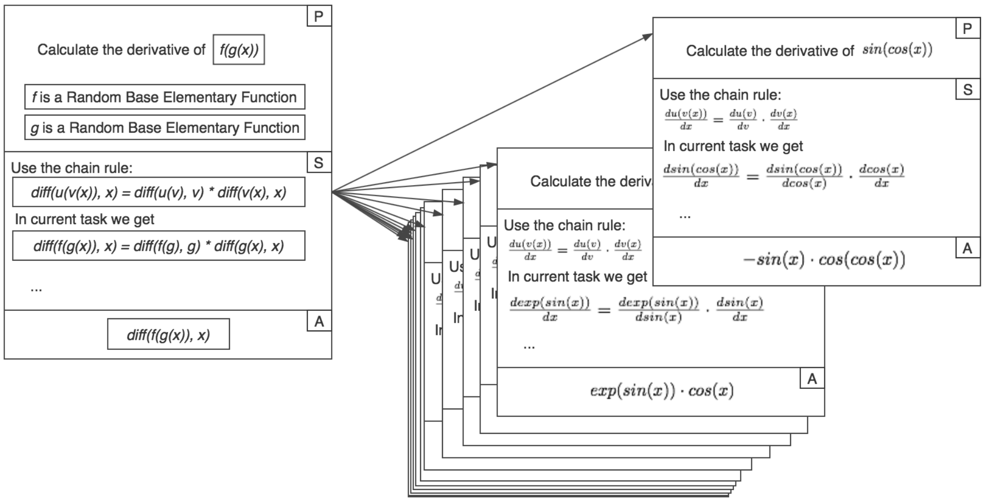


Рис. 1.1: Иллюстрация идеи шаблонизации тренировочных задач

Рис. 1.1 иллюстрирует описанную идею шаблонов тренировочных задач. По одному такому описанию может быть сгенерировано множество однотипных задач на заданную тематику. Все задачи, получаемые с помощью подстановки конкретных значений динамических параметров, являются однотипными и считаются эквивалентными. Эквивалентность понимается здесь в том смысле, который вкладывает в это автор шаблона, в роли которого могут выступать преподаватели, ассистенты и менторы курса. Данное свойство позволяет ввести автоматическую генерацию задач: так как с точки зрения образовательного процесса нет различий между конкретными генерациями, выбор конкретных значений динамических параметров может быть осуществлен с помощью рандомизации по соответствующим множествам значений. Таким образом, концепция шаблонизации тренировочных задач обладает несколькими важными качествами. Во-первых, составление множества тренировочных задач на заданную тематику сводится к описанию базового шаблона задач (или нескольких шаблонов, в случае составной темы), что серьезным образом экономит время составителя. В качестве дополнительного примера рассмотрим небольшую выборку задач из «Сборника задач и упражнений по математическому анализу» Б.П. Демидовича [30] — традиционного сборника, использующегося в большинстве ВУЗов России, — представленную на Рис. 1.2. В данных задачах студенту предлагается вычислить производные указанных функций. Все эти задачи (и великое множество других) могут быть получены с помощью одной-единственной конструкции *“Вычислите производную f”* с определением множества значений f как множество элементарных функций, полученных с помощью не более чем, например, 5 операций сложения, вычитания, умножения, деления или композиции. Во-вторых, назначение задач студентам может осуществляться в автоматическом режиме непосредственно по запросу обучающегося — предоставляется случайная генерация шаблона. В-третьих, снабженные в том числе конкретизированными ходом решения и ответом, такие задачи допускают автоматическую проверку ответа, что позволяет практически полностью исключить преподавателя из процесса обработки конкретных задач. Автоматическая проверка ответов является чрезвычайно важной, неотъемлемой составляющей данного подхода и системы EdLeTS. Данная функциональность, однако, связана с возникновением некоторых теоретических и практических затруднений. Этому вопросу посвящена **??** настоящей работы.

Идея шаблонизации тренировочных задач позволяет ввести выскокий уровень автоматизации в процессы составления, распространения и обработки тренировочных задач в рамках практических занятий по математическим дисциплинам. Строго говоря, данная идея является достаточно общей и область ее применения не ограничивается одними лишь математическими дисциплинами. Достаточно очевидно, что такой подход может быть весьма успешно адаптирован под нужды разнообразных дисциплин из области точных наук и не только. Однако в данном исследовании, как уже упоминалось выше, мы ограничимся рассмотрением математических дисциплин. Реализация предложенной модели в рамках

858. $y=\sqrt[3]{\frac{1+x^3}{1-x^3}}$.

859. $y=\frac{1}{\sqrt{1+x^2}\,(x+\sqrt{1+x^2})}$.

860. $y=\sqrt{x+\sqrt{x+\sqrt{x}}}$.

861. $y=\sqrt[3]{1+\sqrt[3]{1+\sqrt[3]{x}}}$.

862. $y=\cos 2x-2\sin x$.

863. $y=(2-x^2)\cos x+2x\sin x$.

864. $y=\sin(\cos^2 x)\cdot\cos(\sin^2 x)$.

865. $y=\sin^n x\cos nx$. **866.** $y=\sin[\sin(\sin x)]$.

867. $y=\frac{\sin^2 x}{\sin x^2}$. **868.** $y=\frac{\cos x}{2\sin^2 x}$.

869. $y=\frac{1}{\cos^n x}$. **870.** $y=\frac{\sin x-x\cos x}{\cos x+x\sin x}$.

Рис. 1.2: Примеры задач по теме «Производная явной функции» из сборника Б.П. Демидовича

образовательной системы EdLeTS позволяет предоставить необходимые вохможности автоматизации процесса обучения и также позволяет повысить уровень персонализированности обучения за счет сбора подробной персонализированной информации о прогрессе студентов. Следующий раздел настоящей главы посвящен реализации описанной в данном разделе идеи в рамках системы EdLeTS.

1.2 Реализация модели шаблонов тренировочных задач и язык описания динамических параметров

В образовательной системе EdLeTS шаблоны тренировочных задач представлены моделью, имеющей текстовые поля для постановки условия, хода решения и ответа шаблона. Помимо собственно математических выражений, всякая задача содержит сопроводительный текст, разъясняющий постановку или комментирующий ход решения. Для простоты изложения будем считать, что ответом всегда является одном математическое выражение без какого-либо сопроводительного текста. Таким образом, соответствующие поля модели шаблона тренировочной задачи (постановка и ход решения) должны содержать текстовые данные со специальными вставками, описывающими динамические параметры.

В качестве языка разметки для текстовых данных был избран Markdown — простой язык разметки, допускающий преобразование в продвинутые языки разметки (такие как, например, HTML), удобный для быстрой правки и легкой разметки текста; создан Джоном Грубером (John Gruber) в 2004 году. Основным качеством Markdown в данном случае является его простота: использовать его может любой пользователь, располагающий небольшой информацией о правилах этого языка. Изучение этих правил не требует большого времени и не предполагает длительного вникания в суть происходящего. Другим важным качеством этого языка является ограниченность его функциональности. Это позволяет контролировать содержимое текстовых данных задач и серьезно упрощает организацию защиты от инъекции вредоносного кода. Многие современные реализации Markdown поддерживают также и ввод конструкций HTML, однако, эту возможность зачастую можно отключить или ограничить каким-либо набором тегов.

Отдельное внимание необходимо уделить вводу статических математических выражений. В качестве языка описания математических выражений в EdLeTS был избран $\TeX$(его соответствующее подмножество), разработанный Дональдом Кнутом (Donald Knuth) в 1978 году [31]. Математические выражения TeX чрезвычайно проработаны и удобны в использовании и сам этот язык широко распространен среди деятелей образования и науки, занятых в области точных наук. По этой причине поддержка TeX в системе является крайне желательной, многие преподаватели находят удобным ввод математических выражений именно в таком виде. Таким образом, текстовые данные в шаблонах тренировочных задач в EdLeTS представлены размеченным с помощью Markdown текстом, включающим вставки на языке $\TeX$, а также — вставки описаний динамических параметров, о которых далее и пойдет речь.

Для описания динамических параметров задач был разработан специальный язык, получивший название SmallTask. Данный язык полностью посвящен проблеме описания параметров тренировочных задач и разрабатывался с учетом специфики области его применения. Современные языки программирования, использующиеся в разработке реального программного обеспечения (ПО), обладают выразительностью, позволяющей использовать их также и в целях описания динамических параметров задач в рамках описываемого подхода. Все они, однако, обладают рядом существенных недостатков (применительно к данной задаче). В первую очередь, все эти языки являются избыточными для такой задачи. Эта избыточность приводит к таким проблемам, как усложненное описание математических конструкций, необходимость соблюдения специфических синтаксических норм, не связанных напрямую с целью, уязвимость к инъекциям вредоносного кода, необходимость изучения конструкций, не связанных непосредственно с описанием динамических параметров задач. Все эти проблемы, в свою очередь, влекут за собой проблемы безопасности, сложность обработки языковых конструкций и отторжение пользователями, не желающими вникать в нюансы задач, не связанных с их непосредственной деятельностью в рамках системы EdLeTS. Вдобавок ко всему, не будучи нацелены на определение динамических параметров тренировочных задач, обычные языки программирования не обладают необходимым набором типов данных, классов объектов и функций, позволяющих легко и быстро решать задачи определения разнообразных типов параметров задач и рандомизации по множествам их значений. Эта проблема может быть решена разработкой специальной библиотеки, однако, наличие всех прочих недостатков делает использование стандартных языков программирования в рамках поставленной задачи чрезмерно неэффективным.

Другой тип кандидатов на роль языка описания динамических параметров тренировочных задач — языки СКА. Языки этого рода зачастую гораздо больше нацелены на задачи проведения разнообразных математических расчетов и предоставляют широкий набор необходимых типов данных и классов; в них также зачастую сильно упрощена процедура описания математических выражений и преобразований над ними. Избыточность, однако, присутствует и в этих языках, так как область их применения включает сложные математические расчеты, разнообразных моделей и прочие сложные операции, что накладывает определенные требования к выразительным возможностям языка и усложняет его. При этом, безусловно, необходима реализация символьных вычислений над выражениями, задаваемыми в динамических параметрах. Достаточно очевидно, что самостоятельная разработка соответствующих средств является здесь излишней: это трудоемкий и сложный процесс, который выходит за рамки разработки интерактивной системы поддержки практических занятий, причем задача эта уже успешно решена в различных СКА. Системы компьютерной алгебры представляют собой инструмент, который в рамках поставленной задачи может быть успешно использован в качестве вычислительного ядра системы. При этом возникает дополнительное требование к языку описание динамических параметров тренировочных задач — возможность быстрой и надежной трансляции в выражения используемой системы компьютерной алгебры.

Для языка описания динамических параметров задач важно также одно специфическое свойство — возможность контролировать вычисление выражения. Вычислительные задачи по математическим дисциплинам зачастую состоят в выполнении преобразований над исходным выражением, приводящих к конечному результату, или включают такие манипуляции в состав хода решения. Таковы, например, все вычислительные задачи математического анализа, посвященные вычислению пределов, дифференцированию и интегрированию. При описании шаблонов таких задач важно иметь возможность хранить выражения в исходном виде, без вычислений (кроме, быть может, некоторых тривиальных преобразований). Особо это важно для описания хода решения, где описание многих шагов строится по принципу

<исходное выражение> = *<вычисленное выражение>*

Возможность хранить невычисленную форму выражений, претерпевших лишь минимальные преобразования (например, замена "+-" на «-»), играет важную роль при описании тренировочных задач и ее отсутствие сильно усложняет жизнь авторам задач. Традиционные языки программирования и языки СКА не обладают такой возможностью в достаточной мере[3].

Изложенные выше недостатки существующих формальных языков применительно к описываемой задаче отражают ряд требований, предъявляемых к языку в контексте описания шаблонов тренировочных задач. Дополнительные требования накладывает необходимость обеспечить наивысшее удобство пользователей при взаимодействии с системой: даже в том случае, когда пользователю по той или иной причине случится столкнуться с языковыми конструкциями при описании динамических параметров, этот процесс должен пройти наиболее легко, безболезненно, не должен накладывать дополнительных требований к квалификации пользователя (помимо его непосредственных профессиональных навыков), не должен требовать погружения и дополнительного изучения.

[3] Здесь следует уточнить, что некоторые СКА (например, Mathematica) предоставляют инструменты, позволяющие проследить этапы преобразования выражения или даже «заморозить» выражение в текущем его виде. Подобные инструменты способны служить для решения указанной проблемы, однако их использование также усложняет процесс описания динамических параметров и требует от автора знаний в сторонних областях.

Исходя из перечисленных характеристик проблемы описания динамических параметров тренировочных задач и общих требований к организации взаимодействия пользователей с системой, можно сформулировать следующие требования, предъявляемые к языку описания динамических параметров:

- язык должен строго соответствовать задаче описания динамических параметров, сохраняя максимальную синтаксическую и концептуальную простоту, и должен быть лишен излишних синтаксических конструкций;

- все результирующие конструкции языка должны быть представимы в репрезентативной математической нотации;

- должны быть предоставлены средства, осуществляющие трансляцию выражений языка во внутренний язык СКА, используемой в качестве вычислительного ядра системы; язык должен разрабатываться с учетом этих средств и поддерживать такую функциональность на внутреннем уровне;

- все выражения языка должны поддерживать сохранение исходной структуры с минимальным набором тривиальных преобразований;

- основная библиотека должна предоставлять широкие возможности рандомизации по разнообразным множествам;

- язык должен обладать устойчивостью к случайным или злонамеренным инъекциям вредоносного кода, а подобные случаи не должны приводить к перебоям в работе системы.

Указанные требования достаточно однозначно свидетельствуют о необходимости разработки специализированного языка описания динамических параметров шаблонов тренировочных задач. В рамках настоящего диссертационного исследования был разработан язык SmallTask, построенный на основе этих требований.

SmallTask представляет собой контекстно-свободный язык (то есть, язык, описываемый контекстно-свободной грамматикой — тип 2 в иерархии Н. Хомского [32]). Для разработки основного интерпретатора SmallTask был использован автоматизированный генератор анализаторов языков — ANTLR[4] версии 3 [33]. ANTLR позволяет генерировать лексические и синтаксические анализаторы языка на основе заданного описания его грамматики в формате расширенной формы Бэкуса-Наура (РБНФ). Также ПО ANTLR предоставляет средства для генерации анализаторов абстрактных синтаксических деревьев (АСД). Эти средства используются интерпретатором SmallTask для построения внутренней объектной модели выражений по результатам лексического анализа.

В Таблица 1.1 и Таблица 1.2 приведен неполный список основных типов данных и классов объектов, определяемых языком SmallTask. Данный набор легко расширяется за счет описания соответствующих классов в модуле, описывающем типы узлов деревьев выражений языка. В простейшем случае описание нового типа сводится к созданию класса, унаследованного от специального класса функций, удовлетворяющего протоколу именования таких классов и, в случае необходимости, определяющего способ разбора параметров, поступающих при конструкции объекта. Переопределение соответствующих методов позволяет управлять преобразованием выражений описываемого типа в репрезентативную нотацию (TEX), в язык системы компьютерной алгебры, управлять вычислением объектов, определять способ применения подстановок, а также — результаты действия различных операторов. В SmallTask порождение объектов таких типов выполняется либо напрямую вызовом соответствующей функции-конструктора, либо косвенно — как результат выполнения какой-либо функции или операции. Добавление основных типов данных, порождаемых в результате разбора абстрактного синтаксического дерева выражения требует также внедрения соответствующих изменений в грамматику анализатора АСД. Этот

[4] Официальный сайт проекта: `http://www.antlr.org`

процесс является более сложным, однако не представляет серьезных затруднений и происходит достаточно редко.

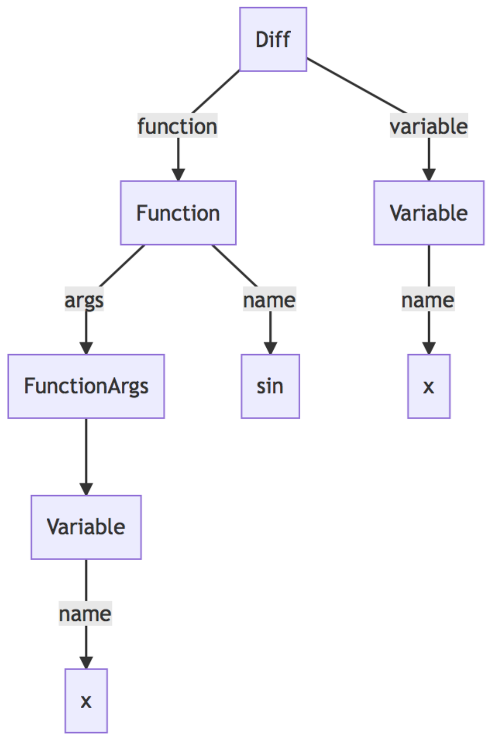


Рис. 1.3: Пример дерева выражения в SmallTask

Всякий объект в SmallTask представлен деревом, которое в случае сложных типов может иметь достаточно сложную структуру. Это представление формируется сразу при распознавании выражения и сохраняется в памяти до окончания сеанса. Пример такого дерева для выражения `diff(sin(x),x )` (задает производную функции $sin(x)$ по x) представлен на Рис. 1.3. Наличие многих из описанных типов продиктовано их специфической ролью в контексте описания тренировочных задач, и ровно в силу этой специфики такие типы данных отсутствуют в традиционных языках программирования и языках различных СКА. К примеру, традиционно мы не будем наблюдать в языке отдельного типа для производной. Вместо этого будет предоставлена функция, вычисляющая производную заданного выражения. Однако требование сохранения исходной формы выражения (в данном случае — невычисленной производной некоторой функции) и возможности работы с этой формой обуславливает необходимость введения соответствующего типа данных, для которого определены специфические правила преобразования, отображения в репрезентативной математической нотации, правила применения подстановок и прочие характеристики. Наличие таких типов в SmallTask позволяет решить указанные во вступлении к данной главе соответствующие недостатки традиционных языков.

Таблица 1.1: Основные типы данных SmallTask

Тип	**Описание**
`Integer`	Целые числа
`Float`	Числа с плавающей запятой
`Variable`	Переменная
`Constant`	Константы (специальный тип)
`Equation`	Уравнение
`String`	Строка
`List`	Список
`Sum`	Сумма
`Prod`	Произведение
`Fraction`	Отношение
`Power`	Степень
`Unary`	Выражение с унарным оператором

Таблица 1.2: Сложные типы данных SmallTask

Тип	**Описание**
`Function`	Функция (математическая)
`Abs`	Модуль
`Poly`	Полином
`Limit`	Предел
`Diff`	Производная

`DefiniteIntegral`	Определенный интеграл
`IndefiniteIntegral`	Неопределенный интеграл
`Sqrt`	Квадратный корень
`Factorial`	Факториал
`Order`	О-большое
`Substitution`	Подстановка
`Plot2D`	Двумерный график функции
`Scatter2D`	График, состоящий из набора точек
`MultiPlot`	Совместный 2-мерный график нескольких функций

SmallTask предоставляет набор основных функций, используемых авторами задач при описании динамических параметров шаблонов. Эти функции можно разделить на 4 основных типа:

- функции порождения и управления выражениями;
- функции для построения графиков;
- функции рандомизации;
- функции для работы с последовательностями и рядами;

Различные функции рандомизации представлены в Таблица 1.3. Множества специальных функций могут быть легко пополнены. Добавление новой функции сводится к определению функции на Python с использованием специализированного декоратора[5], в котором указывается количество ожидаемых аргументов соответствующей функции в SmallTask, с последующим ее добавлением с словарь

[5] В данном случае — способ подключения дополнительной функциональности к описываемой функции в языке Python (реализация одноименного шаблона проектирования), а также функция или объект, предоставляющая соответствующую дополнительную функциональность. Подробное описание декораторов в Python доступно по следующей ссылке: `https://www.python.org/dev/peps/pep-0318/`

дополнительных функций. Функции из указанного словаря становятся автоматически доступны в языке SmallTask и могут быть использованы напрямую.

Разработанный в рамках настоящего исследования язык SmallTask нацелен на устранение описанных недостатков традиционных языков программирования и сконструирован в соответствии с установленными требованиями к языку описания динамических параметров тренировочных задач. Структура языка SmallTask отвечает исключительно этой цели и сохраняет синтаксическую простоту в допустимых пределах. Все типы, предоставляемые языком определяют способ отображения их в TEX, за исключением специальных графических типов, таких как графики, которые отображаются согласно специфическим правилам. Все математические выражения языка могут быть транслированы в язык используемой в EdLeTS СКА, с помощью которой осуществляется вычисление выражений. Способ хранения выражения в виде исходных деревьев позволяет, в частности, в любой момент времени получить доступ к исходной форме выражения — без вычислений. При этом, в процессе построения деревьев выражений, на этапе анализа АСД применяются минимальные преобразования, позволяющие избежать ненамеренного нагромождения операций. Так, например, при случайном выборе некоторого числа из интервала, включающего отрицательные числа, с последующим прибавлением этого значения к некоторому выражению, возникает выражение вида $expr + -a$, где $+-$ не несет смысловой нагрузки, но синтаксически «портит» выражение. Такого рода преобразования в рамках данной работы считаются тривиальными и производятся автоматически.

SmallTask предоставляет широкий набор функций рандомизации, который может быть легко и быстро пополнен разработчиком системы. Набор простых и сложных типов данных также допускает пополнение. Такой возможность располагает исключительно разработчик, который берет на себя ответственность за безопасность использования описанных им функций. Всякое выражение

Таблица 1.3: Список функций рандомизации SmallTask

Имя функции	**Множество, из которого производится случайный выбор**
`Random`	Числа с плавающей точкой в заданном интервале с заданным шагом
`RandomInteger`	Целые числа в заданном интервале
`RandomRational`	Рациональные числа в заданном интервале
`RandomChoice`	Заданный набор
`RandomListChoice`	Заданный список
`RandPoly`	Полиномы заданной степени
`RandIntegerPoly`	Полиномы заданной степени с целочисленными коэффициентами
`RandomParabola`	Квадратичные функции с коэффициентами в заданном отрезке
`RandomIntegerParabola`	Квадратичные функции с целочисленными коэффициентами (в заданном отрезке)
`RandomRationalFunction`	Функция вида $\frac{P}{Q}$, где P и Q — полиномы
`RandomTranscFunction`	Трансцендентные функции (базовые)
`RandomTrigFunction`	Основные тригонометрические функции
`RandomInvTrigFunction`	Основные обратные тригонометрические функции
`RandomHyperbolicFunction`	Основные гиперболические функции
`RandomElemFunction`	Элементарные функции, полученные из основных путем применения указанного числа операций $+, -, /, *$ и композиции
`RandomCombo`	Всевозможные комбинации функций из заданного списка (с операциями $+$, $-$ $,/,*$ и композиции)
`RandomLinCombo`	Всевозможные линейные комбинации заданных функций

SmallTask без исключений проходит через анализатор языка, набор допустимых конструкций которого весьма ограничен. Этот факт делает достаточно прозрачным весь процесс обработки выражений. При этом строгий контроль типов на этапе разбора выражений не позволяет производить инъекции стороннего кода в функции. Возможности SmallTask оказываются весьма ограничены — их достаточно для определения параметров задач, но не более. Это позволяет серьезным образом снизить риск атак, реализуемых недобросовестными пользователями системы. Возможность встраивания вредоносного кода в строковый тип данных SmallTask ограничивается невозможностью экранировать символ завершения строки, так как соответствующий символ не поддерживается языком. Защита от инъекций SQL осуществляется средствами, предоставляемыми используемым каркасом веб-приложений, при записи в БД.

1.3 Выводы

Понятия тренировочной задачи лежит в центре описываемого подхода к поддержке процесса обучения и разработанной в рамках настоящего исследования системы EdLeTS. Концентрирование внимания именно на этой составляющей процесса вызвано, во-первых, тем обстоятельством, что тренировочные задачи играют важнейшую роль в обучении математическим дисциплинам, а выработка соответствующих вычислительных навыков является необходимым условием дальнейшего продвижения в этой области, а, во-вторых, именно процессы, связанные с порождением, распространением и обработкой тренировочных задач, а также — со сбором и анализом всевозможной информации о прогрессе обучающихся, допускают высокую степень автоматизации. Последнее обстоятельство позволяет освободить преподавателей и студентов от большого объема рутинной работы, не отражающей истинные цели обучения, а являющейся, скорее, препятствием, негативно сказывающимся на эффективности этого процесса. Предлагаемый подход позволяет освободить время практических занятий для тех видов деятельности, которые действительно требуют участия преподавателей и коллективной работы студентов.

Описанная в разделе 1.1 настоящей главы модель шаблонов тренировочных задач позволяет устранить такие недостатки традиционного подхода к построению практических занятий, как проблемы построения необходимого количества задач, отсутствие конкретных указаний при возникновении проблем у студентов, затрудненный поиск ошибок, проблема исправленной ошибки и многие другие. Возможности, которые сулит реализация такой модели, включают предложение более высокого уровня персонификации — за счет введения более показательных мер оценки проработанности той или иной задачи и широких возможностей по сбору и анализу прогресса каждого студента в отдельности. В главе 3 приводятся описания различных способов построения заключений, на основе действий и ответов студента.

Предлагаемый метод реализации писанной модели шаблонов тренировочных задач окончательно формализует это понятие. Выбор языка разметки Markdown обусловлен его предельной простотой и интуитивной понятностью. Набор базовых возможностей стилизации текста, поддерживаемый данным языком, вполне достаточен для целей описания шаблонов задач. Язык SmallTask, разработанный специально для нужд описания динамических параметров задач берет на себя все функции, связанные непосредственно с содержательной частью задачи. Данный язык достаточно успешно справляется с недостатками других языков программирования и разработан согласно выявленным требованиям к такому языку. Его специфика делает его применимым исключительно в описываемой области — и в этом его преимущество, так как высокая специализированность позволяет максимально упростить синтаксические конструкции и обеспечить надлежащий уровень безопасности.

Глава 2

Автоматическая проверка ответов

Задача автоматической проверки ответов непосредственно связана с проблемой установки *семантической эквивалентности* двух математических выражений. Два математических выражения будем называть семантически эквивалентными, если они оба описывают один и тот же математический объект. Если два выражения имеют одинаковую репрезентацию, то есть, их запись совпадает полностью, их называют синтаксически эквивалентными. Очевидно, что синтаксически эквивалентные выражения всегда являются семантически эквивалентными, однако, обратное неверно: к примеру, выражения 2^x и e^{xln2} являются семантически эквивалентными, но синтаксической эквивалентности нет.

Легко показать, что задача установления семантической эквивалентности двух математических выражений полностью эквивалентна задаче сравнения выражения с нулем (так как $f_1 \equiv f_2$ тогда и только тогда, когда $f \overset{\Delta}{=} f_1 - f_2 \equiv 0$). Данная задача, известная как проблема тождественности (математических) выражений, является частным случаем более общей проблемы тождества слова в некоторой формальной системе. П.С. Новиков [34] доказал неразрешимость проблемы тождества слов в группах в общем случае. Позже Д. Ричардсон в [4] рассмотрел вопрос об установлении семантической эквивалентности математических выражений, отвечающих элементарным функциям[1]. Им было доказано, что для класса элементарных функций, включающем $ln2$, π, e^x, $\sin x$ и $|x|$ не может существовать алгоритма, устанавливающего тождество соответствующих математических выражений. Соответствующий результат получил название теоремы Ричардсона [35]. В дальнейшем этот результат был усилен в работе М. Лацковича [36], где было показано, что в кольце, порождаемом целыми числами и выражениями x, $\sin x$, $\sin(x \cdot \sin x^n)$ $(n = 1,2,\ldots)$, неразрешима задача доказательства/опровержения существования вещественного корня функции.

Данное обстоятельство — невозможность представить алгоритм, достоверно сравнивающий два математических выражения — поднимает большую проблему с точки зрения разработки системы электронного обучения, поддерживающей математический аппарат. Автоматическая проверка ответов, заданных в математической нотации — ключевая и неотъемлемая составляющая рассматриваемой системы. И в то же время, нет возможности гарантировать корректность такой проверки, полагаясь лишь на средства сравнения выражений, предоставляемые СКА. Может показаться, что подобные проблемы на практике возникают достаточно редко. Это отчасти верно, однако исследования показали, что проблемы возникают в реальных задачах и, вместе с тем, непредсказуемо, что делает необходимым поиск метода корректной обработки таких ситуаций. Показательным является следующий пример: равны ли тождественно выражения $\frac{sec(arccsc(x))}{x^2\sqrt{1-\frac{1}{x^2}}} \cdot tg(arccsc(x))$ и $\frac{-1}{\left(1-\frac{1}{x^2}\right)^{3/2} \cdot x^3}$? Некоторые из исследованных СКА не смогли дать правильного ответа на этот вопрос. Вместе с тем, оба эти выражения выражают производную функции $\frac{1}{\sqrt{1-1/x^2}}$ (по переменной x), причем каждый

[1] Элементарные функции — функции, которые можно получить с помощью конечного числа арифметических операций и композиций основных элементарных функций (степенные, показательные, логарифмические, тригонометрические, обратные тригонометрические) [29]

из этих результатов соответствует определенному пути вычисления. Данный пример был обнаружен при исследовании причин нарушения работоспособности системы в процессе ее использования. Следует также отдельно отметить, что подобное проявление некорректной проверки ответа при использовании системы в образовательных учреждениях недопустимо ввиду возможности возникновения негативных последствий как для обучающихся (с точки зрения показателей успеваемости), для рассматриваемого продукта в контексте приятия его сообществом преподавателей и студентов.

Алгоритм проверки пользовательского ответа к задаче должен возвращать положительный или отрицательный результат для правильного или неправильного ответа соответственно. В большинстве СКА невозможность установления эквивалентности двух выражений приводит к остановке алгоритма сравнения выражений без возврата положительного или отрицательного ответа. Таким образом, не может возникнуть ситуации ложного результата. При этом какой-то однозначный ответ все же необходимо дать[2]. Есть два тривиальных способа разрешения этой проблемы: всегда трактовать неоднозначный ответ как правильный или, наоборот, всегда трактовать неоднозначный ответ как неправильный. Второй путь может ввести студента в замешательство, а также нарушает его права (в особенности, при использовании рассматриваемой системы в качестве средства контроля успеваемости и оценки знаний). Подобное поведение системы исключительно негативно влияет на ее оценку пользователями, что негативно сказывается на успехе внедрения. С этой точки зрения первый вариант кажется более предпочтительным, так как не оказывает непосредственного негативного влияния на студентов, а большую часть времени возникающие проблемы остаются попросту незамеченными. Вместе с тем, очевидно, что незамеченные таким образом ошибки могут негативно сказываться

[2] Известен также и другой метод обхода подобных проблем в задачах, генерируемых случайным образом — перегенерировать задачу до тех пор, пока проблема не исчезнет. Очевидны недостатки такого подхода: отсутствуют гарантии достижения успеха, а также теряется всякая полезная информация от «проблемного» сеанса.

на обучении конкретных студентов. Более удачным методом обхода проблем, возникающих вследствие невозможности достоверной проверки ответа средствами СКА, является дополнительная проверка, позволяющая установить корректность ответа с некоторой степенью достоверности. В данном разделе будет предложен алгоритм дополнительной проверки, позволяющий повысить вероятность корректной обработки ответа.

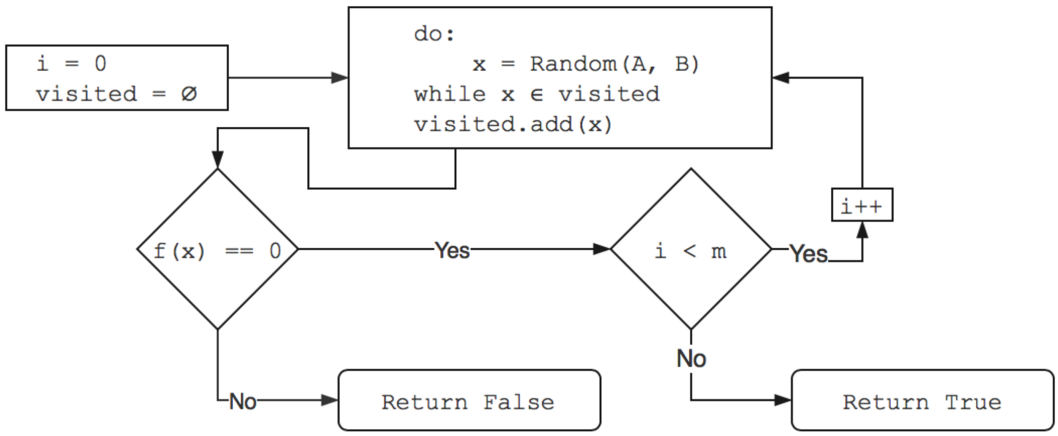


Рис. 2.1: Упрощенная блок-схема алгоритма дополнительной поточечной проверки

2.1 Алгоритм дополнительной проверки

Рассмотрим конкретную задачу T, ответом к которой является элементарная функция одного аргумента. Обозначим через $f_{comp}(x)$ ответ к задаче T, вычисленный программно, а через $f_{user}(x)$ — ответ, полученный пользователем. Задача проверки правильности пользовательского ответа в таком случае сводится к вычислению функции $f(x) \triangleq f_{comp}(x) - f_{user}(x)$ и сравнению ее с нулем: $f \stackrel{?}{\equiv} 0$. Пусть это тождество не может быть доказано или опровергнуто средствами СКА. Для разрешения этой проблемы предлагается *алгоритм* *поточечной проверки* . Данный алгоритм принимает на вход функцию и проверяет ее значения, сравнивая их с нулем, поочередно в каждой точке из набора точек, выбранных случайным образом в наперед заданном отрезке $[A,B]$. Если в очередной точке x_i значение $f(x_i)$ оказывается отличным от нуля, алгоритм возвращает отрицательный ответ. Если же во всех точках набора рассматриваемая функция принимает нулевое значение, алгоритм возвращает положительный ответ. Максимальное количество точек проверки задается заранее, точки выбираются случайным образом из равномерного распределения на отрезке $[A,B]$ без повторений (так как нет необходимости проверять значение функции в одной и той же точке дважды). Схема алгоритмма дополнительной поточечной проверки прдествалена на Рис. 2.1. Перед началом работы происходит инициализация необходимых объектов — объекта, предоставляющего доступ к равномерно распределенным на $[A,B]$ числам, множества «использованных» точек, и переопределение функции f таким образом, чтобы ограничить время ее вычисления в заданной точке (вопрос об обработке ситуации с чрезмерным временем вычисления значения будет рассмотрен в подразделе **??** данной главы). Далее, на каждой итерации цикла от 0 до $m-1$ (где m — максимальное количество точек проверки) осуществляется генерация очередного случайного числа d из заданного отрезка, отличного от уже проверенных, и вычисляется $f(d)$. Если по той или иной причине это значение вычислить не удается, счетчик цикла уменьшается на единицу (данная проверка не учитывается). Затем значение $f(d)$ сравнивается с нулем и, если это значение оказывается отличным от нуля, алгоритм завершает работу, возвращая отрицательный результат проверки (False). В противном случае алгоритм продолжает работу. Если все итерации основного цикла пройдены успешно, алгоритм возвращает положительный результат (True).

Несмотря на свою относительную простоту, предлагаемый алгоритм обладает рядом преимуществ. В первую очередь, следует отметить, что алгоритм работает «в пользу студента» в том смысле, что отрицательный результат не будет получен, если ответ пользователя на самом деле верен. Результат работы алгоритмы носит вероятностный характер, однако, «ошибиться» он может лишь в случае, когда пользовательский ответ на самом деле неверен, а все точки проверки по случайному стечению обстоятельств попали в нули функции f. При этом достаточно очевидно, что при достаточно большом количестве точек проверки вероятность такой ситуации достаточно мала. В предельном случае, при $m \to \infty$, вероятность ошибки стремится к нулю. Однако на практике количество точек проверки должно быть строго ограничено, причем так, чтобы время, затрачиваемое на осуществление проверки, в конечном итоге оказалось приемлемым с точки зрения пользователя. Ниже приведен анализ вероятности ошибки предложенного алгоритма.

2.2 Вероятность ошибки алгоритма дополнительной поточечной проверки

В данном разделе рассматривается вероятность получения неправильного результата при использовании предложенного алгоритма дополнительной поточечной проверки, выводится ее оценка и рассматриваются некоторые конкретные примеры, дающие представление о том, как рассматриваемый алгоритм ведет себя на практике. Будет показано, что полученная теоретическая оценка вероятности ошибки алгоритма делает возможным и целесообразным использование этого алгоритма на практике, а результаты экспериментов это подтверждают.

Пусть, как и выше, $f = f_{comp} - f_{user}$ — элементарная функция, принимающая свои значения на множестве вещественных чисел. Пусть также $A,B \in \mathbb{R}$ таковы, что $A < B$ и функция f является аналитической на отрезке $[A,B]$. Будем считать, что значения функции f вычисляются точно в том смысле, что проверка равенства

$f(d) \stackrel{?}{=} 0$ для некоторого $d \in [A,B]$ всегда производится корректно. Имеет место следующее утверждение.

Утверждение 1. *Если на самом деле $f \equiv 0$, то алгоритм выдает правильный результат.*

Действительно, если $f \equiv 0$, то, каковы бы ни были границы отрезка $[A,B]$, для любой точки $d \in [A,B]$ будет выполнено равенство $f(d) = 0$. Следовательно, после выполнения m итераций основного цикла алгоритм завершит выполнение, вернув положительный ответ.

Пусть теперь $f \not\equiv 0$ и пусть $k \in \mathbb{N}$ — такое натуральное число, что $|x \in [A,B] : \quad f(x) = 0| < k$. Иными словами, k задает некоторую верхнюю оценку на число нулей функции f в отрезке $[A,B]$. Как и выше, через m будем обозначать максимальное число точек проверки. Будем различать следующие два случая:

- $k < m$ — количество точек проверки превосходит максимальное возможное число нулей исследуемой функции

- $k \geq m$ — количество нулей функции f на отрезке $[A,B]$ может превосходить максимальное число точек проверки

Утверждение 2. *Если $k < m$, рассматриваемый алгоритм всегда возвращает правильный результат.*

Согласно утверждению 1, если рассматриваемая функция тождественно равна нулю, то алгоритм возвращает правильный ответ. Если же $f \not\equiv 0$, то ошибочный результат — положительный ответ. Для этого необходимо, чтобы на каждой итерации основного цикла алгоритма выбирались точки исключительно из множества нулевого уровня исследуемой функции. Пусть $d_0, d_1, \ldots, d_{m-1}$ — случайная выборка из равномерного распределения на отрезке $[A,B]$. Алгоритм вернет положительный ответ в том и только том случае, когда $\forall i = \overline{0, m-1}\ f(d_i) = 0$. При этом, как обсуждалось выше, все элементы выборки различны: $d_0 \neq d_1 \neq \ldots \neq d_{m-1}$. Далее, так как $m > k$ и $k > |\{x \in [A,B] : \quad f(x) = 0\}|$, то найдется хотя бы одна точка $\tilde{d} \in \{d_0, d_1, \ldots, d_{m-1}\}$ такая, что $f(\tilde{d}) \neq 0$: в худшем случае, первые k случайно выбранных точек окажутся нулями функции f, но, в силу ограничения мощности множества ее нулей и запрета повторений среди исследуемых точек, d_k неизбежно такова, что $f(d_k) \neq 0$. Таким образом, при $m > k$ алгоритм возвращает правильный ответ (причем не более чем за $k + 1$ шаг).

Единственная оставшаяся возможность некорректной работы рассматриваемого алгоритма — случай $m \leq k$. В этом случае возможно, что все m случайно выбранных в процессе работы алгоритма точек окажутся нулями исследуемой функции. Для оценки корректности алгоритма дополнительной поточечной проверки проведем вычисление и анализ вероятности такой ситуации.

Напомним, что функция f предполагается аналитической на отрезке $[A,B]$. Таким образом, f не может иметь бесконечное число нулей внутри этого отрезка (если только она не равна нулю тождественно на всей области определения). В непрерывном случае (когда $[A,B]$ — непрерывный отрезок вещественной прямой, а точки проверки выбираются из непрерывного равномерного распределения на этом отрезке) вероятность ошибки алгоритма описывается вероятностью попасть в заданное конечное дискретное подмножество непрерывного множества и, следовательно, равна нулю. Однако, в контексте программной реализации рассматриваемого алгоритма, мы имеем дело не с непрерывной вещественной прямой, а с дискретной сеткой чисел с плавающей запятой. Отрезок $[A,B]$ превращается в последовательность точек, определяемую используемой системой чисел с плавающей запятой, расстояния между которыми определяются соответствующей процедурой округления. Обозначим через AB множество чисел с плавающей точкой, аппроксимирующих отрезок $[A,B]$. Пусть $M = |AB|$ — мощность этого множества, а $Z = \{x \in [A,B] : \quad f(x) = 0\}$. И пусть, для определенности, $|Z| = k$. Тогда вероятность ошибки алгоритма описывается вероятностью выбрать m точек из k-подмножества множества из M элементов.

Вероятность выбрать одно число из множества AB так, чтобы оно также попадало в некоторое определенное k-подмножество множества AB задается выражением (2.1)

$$\mathbb{P}(d \in Z) = \frac{k}{M} \tag{2.1}$$

После того, как выбрано первое число d_0 (так, что $d_0 \in Z$), останется $k-1$ не покрытых нулей функции f (в исследуемом отрезке) и $M-1$ вариантов выбора следующей точки всего (с учетом запрета на повторения). Таким образом, условная вероятность выбора второй точки из множества нулевого уровня исследуемой функции задается выражением (2.2):

$$\mathbb{P}(d_1 \in Z | d_0 \in Z) = \frac{k-1}{M-1} \tag{2.2}$$

Тогда совместная вероятность выбора подряд двух точек из множества нулевого уровня функции j имеет вид (2.3):

$$\mathbb{P}(d_1 \in Z, d_0 \in Z) = \frac{k \cdot (k-1)}{M \cdot (M-1)} \tag{2.3}$$

Продолжая эти рассуждения, получим формулу (2.4), задающую вероятность выбрать подряд m точек из множества AB так, что все они попадут в множество Z:

$$p_{M,m,k} = \mathbb{P}(d_{m-1} \in Z, \ldots, d_0 \in Z) = \prod_{i=0}^{m-1} \frac{k-i}{M-i} = \frac{k!(M-m)!}{M!(k-m)!} \tag{2.4}$$

Выражение (2.4) задает вероятность ошибки алгоритма при $m \leq k \leq M$. Очевидно, что при $k > M$ алгоритм заведомо возвращает некорректный результат, а при $k < m$, как было показано выше, ошибки быть не может. Таким образом, окончательное выражение для вероятности ошибки алгоритма дополнительной поточечной проверки имеет вид (2.5):

$$P_{err} = \begin{cases} 0, & k < m \\ p_{M,m,k}, & m \leq k \leq M \\ 1, & k > M \end{cases} \tag{2.5}$$

Формула, заданная выражением (2.5) зависит от трех параметров:

- M — количество чисел с плавающей точкой в отрезке $[A,B]$,
- m — максимальное количество точек проверки,
- k — верхняя граница числа нулей исследуемой функции в отрезке $[A,B]$

Для предоставления численных оценок вероятности ошибки алгоритма необходимо сначала предоставить оценки для этих параметров. Единственный параметр, которым можно относительно свободно управлять — это параметр m. Очевидно, что уменьшение количества точек проверки положительно сказывается на скорости получения результата, однако, при этом увеличивается и вероятность ошибки (это понятно эмпирически, а также непосредственно следует из формулы (2.4)). В следующем разделе анализируется зависимость вероятности ошибки от количества точек проверки при фиксированных остальных параметрах и приводятся соответствующие численные оценки. Отдельный интерес представляет параметр k, так как оценка числа нулей в заданном промежутке по ряду причин не может быть автоматизирована, а следовательно, строго говоря, нет общего подхода, позволяющего на каждом конкретном запуске алгоритма автоматически подстраивать, например, количество точек проверки. Наконец, и сам выбор границ отрезка $[A,B]$ оказывается нетривиальной задачей, так как, на практике, функции, с которыми мы сталкиваемся, могут иметь различные особенности в наперед выбранном промежутке, что затрудняет работу алгоритма. Все эти вопросы рассматриваются в следующем разделе.

Анализ вероятности ошибки

Для анализа поведения программной реализации алгоритма дополнительной поточечной проверки, необходимо обратиться к машинной арифметике, рассматривая особенности сетки чисел с плавающей запятой, аппроксимирующей отрезок вещественной прямой, на котором проводится проверка. В данном случае необходимо построить нижнюю оценку количества чисел с плавающей в заданном интервале. Всякая система чисел с плавающей точкой характеризуется, в первую очередь, двумя параметрами: β (основание) и p (точность). В такой системе некоторое вещественное число $\tilde{x}$ будет представлено в виде[3] $R(\tilde{x}) = x = d_0.d_1d_2\ldots d_{p-1}\cdot\beta^e$ (см. [37]). Абсолютная погрешность такого представления может достигать значения $(\frac{\beta}{2}\beta^{-p})\cdot\beta^e = \varepsilon\cdot\beta^e$ (см. [38]), где через ε обозначено машинное эпсилон в данной системе — число, задающее расстояние между единицей и следующим отличным от нее числом. Легко заметить, что расстояние между узлами сетки чисел с плавающей запятой, как функция от модуля числа, монотонно возрастает (нестрого): если $A = x_0 < x_1 < \ldots < x_M = B$ — сетка чисел с плавающей запятой, аппроксимирующая отрезок $[A,B]$ $(0 < A < B)$, то верно следующее неравенство (2.6):

$$x_1 - x_0 \leq x_2 - x_1 \leq \ldots \leq x_M - x_{M-1} \tag{2.6}$$

Таким образом, расстояния между числами с плавающей запятой, содержащимися в заданном отрезке, можно оценить сверху расстоянием между правой границей этого промежутка (точнее, его аппроксимацией) B и следующим узлом сетки. Пусть $\tilde{M}$ — истинное количество чисел с плавающей точкой внутри отрезка $[A,B]$, и

[3] Через R будем обозначать функцию округления вещественного числа к числу с плавающей запятой, т.е. $R(\tilde{x})$ — представление вещественного числа $\tilde{x}$ в заданной системе с плавающей запятой

пусть функция $ulp(\tilde{x})$ для $\tilde{x} \in \mathbb{R}$ задает расстояние между двумя ближайшими к $\tilde{x}$ числами с плавающем запятой: $ulp(\tilde{x}) = b - a$, где $a \leq \tilde{x} \leq b$, $a \neq b$ и $\forall \tilde{x}'$ либо $R(\tilde{x}') = a$, либо $R(\tilde{x}') = b$. Тогда верна следующая оценка (2.7):

$$M = \left[\frac{B - A}{ulp(B)}\right] \leq \tilde{M} \tag{2.7}$$

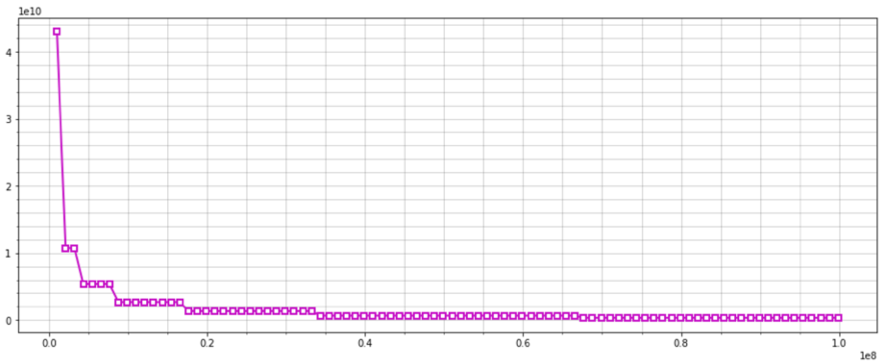


Рис. 2.2: Оценка количества узлов сетки чисел с плавающей запятой в отрезках вида $[A, A + 5]$, как функция A

Данная оценка может оказаться достаточно грубой для достаточно широких отрезков, однако, в данном случае она удовлетворяет необходимым требованиям. Рассмотрим теперь следующий простой способ вычисления расстояния от данного числа с плавающей запятой до ближайшего следующего: зададимся некоторым начальным приближением (например, в качестве такого приближения можно выбрать и само исходное число, так как ширина сетки возле него заведомо меньше) и будем делить его на 2 до тех пор, пока сумма заданного числа и текущего приближения не окажется равной исходному числу (в результате округления до числа с плавающей точкой) — типичный метод дихотомии. Псевдокод этого способа представлен ниже 1. При таком способе вычисления искомого расстояния истинное значение может оказаться меньше, но не более, чем в два раза. В то же время, именно в таком виде алгоритм гарантирует выполнение неравенства[4] $x + ulp_{estimate}(x) > x$.

Algorithm 1 Простой метод подсчета расстояния от данного числа с плавающей запятой до ближайшего следующего

```
function ULP_ESTIMATE(x)
    eps = x
    result = eps
    while x + eps != x do
        result = eps
        eps = epse / 2
    end while
    return result
end function
```

Использование этого метода для подсчета $ulp(x)$ при вычислении оценки M обеспечивает выполнение неравенства (2.6). На Рис. 2.2 изображен график зависимости предложенной оценки M от левой границы отрезка для отрезков ширины 5 для типа данных *NumPy.float64*[5]. Из графика видно, что количество точек в промежутке фиксированной ширины уменьшается по мере удаления от нуля, и этот спад носит экспоненциальный характер. При этом абсолютные значения достаточно велики: к примеру, в промежутке $[10^8, 10^8 + 5]$ оценка количества узлов имеет порядок 10^7. Как уже упоминалось выше, вероятность ошибки $p_{M,m,k}$ убывает как функция M. В связи с этим, предпочтительным является выбор промежутка проверки относительно широким, с наименьшей по модулю

[4] Можно было бы модифицировать алгоритм, возвращая в результате значение *eps* вместо *result*, что соответствует нижней оценке истинного значения $ulp(x)$. Однако выполнение указанного неравенства удобно, например, при прямом подсчете (оценки) количества точек в исследуемом отрезке через последовательное вычисление расстояний между узлами сетки.

[5] NumPy — популярный вычислительный пакет для Python. NumPy.float64 — тип данных, определяемый пакетом NumPy для чисел с плавающей запятой, удвоенной точности.

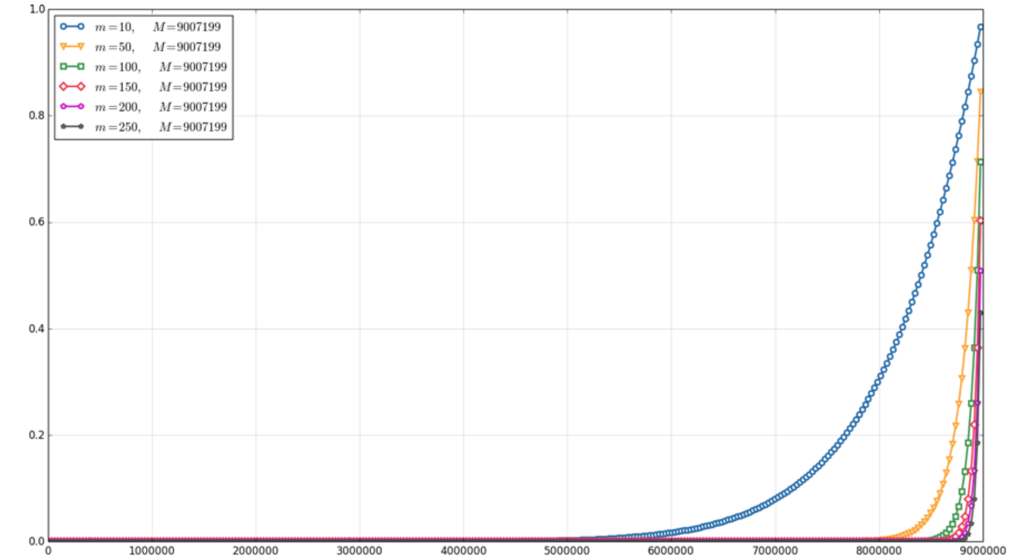
границей вблизи нуля. Однако, как можно видеть, и для небольших отрезков, расположенных достаточно далеко от нуля, значение оказывается достаточно большим — много большим, чем потенциальное число нулей исследуемой функции. Конкретные примеры значений вероятности ошибки алгоритма будут рассмотрены ниже.

На Рис. 2.3(а) представлены графики $p_{M,m,k}$ как функции k для шести различных значений m (10, 50, 100, 150, 200 и 250), при $M = 9007199$ (что отвечает отрезку $[10^9, 10^9 + 1]$ для типа *NumPy.float64*), на Рис. 2.3(б) изображены те же графики в промежутке $[0, 1.1 \cdot 10^{-3}]$ по оси ординат. Из графиков видно, что даже при весьма малом количестве точек проверки $m = 10$, если оценка количества нулей k лежит левее 10^6 вероятность ошибки составит менее 10^{-5}. По мере же увеличения числа точек проверки порог появления существенных вероятностей сдвигается вправо: так, например, при $m = 250$ даже если исследуемая функция будет иметь $8 \cdot 10^6$, вероятность ошибки останется заведомо меньше 10^{-5}. Напомним, что нули исследуемой функции в контексте задачи сравнения ответов — это точки, в которых ответ, полученный пользователем, и ответ, посчитанный системой, совпадают, при условии, что в целом они не равны. Учитывая природу неправильных ответов (разнообразные арифметические ошибки, незнание или неверная трактовка каких-либо вычислительных правил, невнимательность и прочее), возникновение неверного ответа, совпадающего с истинным в большом числе точек, очень маловероятно.

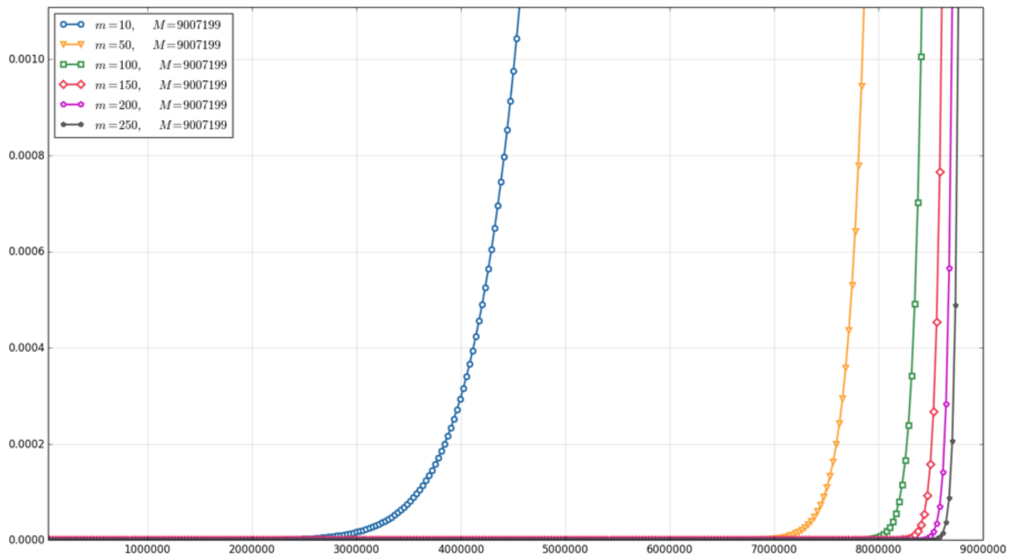
Для демонстрации значений вероятности ошибки алгоритма при запуске на более естественных отрезках воспользуемся следующим свойством (2.8):

$$\ln p_{M,m,k} = \ln\left(\prod_{i=0}^{m-1} \frac{k-i}{M-i}\right) = \sum_{i=0}^{m-1} \ln\frac{k-i}{M-i} \tag{2.8}$$

Последнее выражение в равенстве 2.8 вычисляется точнее и быстрее, чем непосредственно значение $p_{M,m,k}$, так как такие вычисления не приводят к чрезмерно большим числам, и поэтому



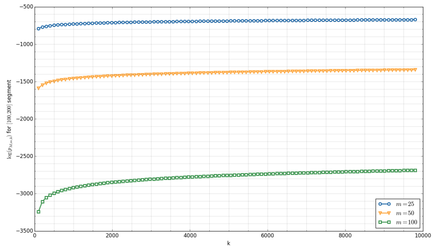
(а)



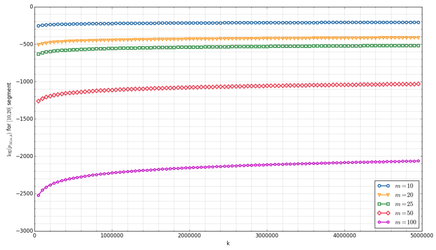
(б)

Рис. 2.3: $p_{M,m,k}$ как функция k для отрезка $[10^9, 10^9+1]$, для различных значений m

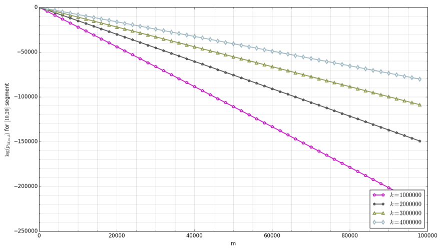
его удобно использовать при построении графика вероятности ошибки алгоритма, что также играет существенную роль при организации возможности настройки параметров алгоритма конечным



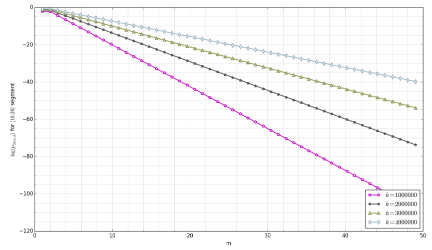
(а) $\log(p_{M,m,k})(k)$ для отрезка $[100, 200]$, $m \in \{25, 50, 100\}$



(б) $\log(p_{M,m,k})(k)$ для отрезка $[10, 20]$, $m \in \{10, 20, 25, 50, 100\}$



(в) $\log(p_{M,m,k})(m)$ для отрезка $[10, 20]$, $k \in \{10^6, 2{\cdot}10^6, 3{\cdot}10^6, 4{\cdot}10^6\}$



(г) $\log(p_{M,m,k})(m)$ для отрезка $[10, 20]$, $k \in \{10^6, 2{\cdot}10^6, 3{\cdot}10^6, 4{\cdot}10^6\}$

Рис. 2.4: $\log(p_{M,m,k})$ как функция k (а, б) и как функция m (в, г).

пользователем-разработчиком задач (подробнее этот вопрос будет рассмотрен ниже). Графики, иллюстрирующие зависимость $\ln p_{M,m,k}$ от m и от k при фиксированном значении M представлены на Рис. 2.4. Важный вывод из этих данных — порядки, о которых идет речь при рассмотрении вероятности ошибки алгоритма. Можно видеть, что, например, для отрезка проверки [10,20] вплоть до значений $k = 5 \cdot 10^6$ всего лишь ста проверок достаточно для того, чтобы вероятность ошибки не превысила значения e^{-2000}. Из Рис. 2.4(в) видно, как убывает исследуемая вероятность с ростом количества точек проверки: кривые на графике убывают практически линейно (хотя, строго говоря, это не совсем так). Это, в свою очередь, означает, что $p_{M,m,k}$ как функция m убывает примерно экспоненциально (на самом деле, еще быстрее). Даже для неправдоподобно больших значений k можно добиться исчезающе малых значений вероятности ошибки алгоритма при сравнительно

m	k	$\ln p_{M,m,k}$
10	1000000.0	−222.281473312
10	2000000.0	−215.349979006
10	3000000.0	−211.295320425
20	1000000.0	−444.563046624
20	2000000.0	−430.700008012
20	3000000.0	−422.590674183
30	1000000.0	−666.844719939
30	2000000.0	−646.050087019
30	3000000.0	−633.886061275
40	1000000.0	−889.126493256
40	2000000.0	−861.400216026
40	3000000.0	−845.181481701
50	1000000.0	−1111.40836658
50	2000000.0	−1076.75039503
50	3000000.0	−1056.47693546

Таблица 2.1: Некоторые значения логарифма вероятности ошибки

небольшом количестве проверок — в том числе, для весьма удаленных узких отрезков. Некоторые конкретные значения логарифма вероятности ошибки алгоритма приведены в таблице 2.1.

До сих пор в стороне оставался вопрос о выборе отрезка исследования функции и также не рассматривалась зависимость вероятности ошибки предложенного алгоритма проверки как функции от M. Трудность здесь заключается в том, что исследуемая функция может иметь различного рода особенности, появление которых в отрезке проверки приводит к нарушению приведенных выше рассуждений. Первое и очевидное требование к такому отрезку — исследуемая функция должна быть на нем определена. Так, бессмысленно запускать алгоритм дополнительной поточечной проверки для функции $f(x) = \ln(-x) - \sin(x)$ на отрезке, лежащем правее нуля. При этом, в ряде случаев автоматическое вычисление области определения исследуемой функции может быть затруднено. И хотя, например, наличие изолированных точек разрыва (при достаточном количестве точек проверки и возможности ограничить время вычисления значения функции) не приведет к нарушению работоспособности алгоритма и сохранит приведенные оценки вероятности его ошибки, таких ситуаций все же лучше избегать. Другие специфические характеристики исследуемых функций на отрезке проверки — такие как, например, наличие осцилляций с быстро возрастающей частотой — могут приводить к появлению большого числа нулей, что соответствующим образом повысит вероятность ошибки. Подобные «проблемные» участки исследуемых функций оказывается затруднительно определить автоматически, однако, для человека, составляющего тренировочное задание, эта задача обычно не составляет проблемы. К примеру, преподаватель дисциплины «Математический анализ» при составлении задачи без особого труда укажет промежуток, на котором ответ будет определен и не будет иметь «лишних» особенностей. Именно по этой причине вопрос о выборе отрезка, на котором будет производиться дополнительная проверка ответа в случае возникновения неопределенности, необходимо оставить на усмотрение автора задачи. В программном продукте, разработанном рамках данного диссертационного исследования пользователям предлагается возможность настройки всех, параметров, влияющих на вероятность ошибки алгоритма дополнительной поточечной проверки — количества точек проверки (m), оценки потенциального числа нулей при проверке (k) и отрезка проверки ($[A,B]$). Для текущего набора параметров строится два графика вероятности ошибки алгоритма — по m и по k в некоторой окрестности выбранных значений, а также демонстрируется непосредственно посчитанная оценка вероятности ошибки. Все это представляет собой исчерпывающий набор, позволяющий тонко контролировать поведение алгоритма дополнительной проверки, подстраивая его под конкретную задачу.

Ограничение времени вычисления значения функции

Еще один вопрос, требующий отдельного внимания — ограничение времени вычисления значения исследуемой функции в алгоритме дополнительной поточечной проверки. Данное ограничение, в первую очередь, служит способом защиты от непредвиденных ситуаций, связанных как с особенностями исследуемой функции, так и с потенциальными вычислительными нюансами используемой системы компьютерной алгебры. Подобные аспекты могут приводить к нецелесообразному увеличению времени вычисления одного значения, вплоть до входа в бесконечный цикл. Подходы к реализации такого рода функциональности различаются в зависимости от конкретной задачи и средств, предоставляемых языком программирования. В рассматриваемой системе для решения этой задачи был применен механизм сигналов в языке Python (подробное описание — в [39]) и менеджер контекстов. Модуль работы с сигналами в языке Python позволяет сгенерировать сигнал заданного типа по истечении определенного времени и определить функцию-обработчик сигнала требуемого типа. В контексте рассматриваемой задачи обработчик сигнала должен сгенерировать исключение, которое просигнализирует о превышении ограничения на время вычисления значения функции. В случае же, если значение оказывается вычисленным прежде, чем закончится отведенное на его вычисление время, сигнал о превышении не будет послан и алгоритм продолжит работу в нормальном режиме. Все, что после этого остается — добавить в алгоритм проверки обработку исключения, связанного с превышением ограничения по времени. Код функции, отвечающей за ограничение времени вычисления значения заданной функции — `limit_execution_time`, — представлен в листинге ниже:

```
import signal
from contextlib import contextmanager


class FunctionExecutionTimeout(Exception):
```

```
    pass


@contextmanager
def limit_execution_time(seconds):
    def signal_handler(sig_num, frame):
        raise FunctionExecutionTimeout
    signal.signal(signal.SIGALRM, signal_handler)
    signal.alarm(seconds)
    try:
        yield
    finally:
        signal.alarm(0)
```

files/limit_execution.py

Класс `FunctionExectutionTimeout` задает исключение, генерируемое при превышении ограничения на время вычисления. Декоратор `contextmanager`, предоставляемый модулем `contextlib`, позволяет использовать `limit_execution_time` как менеджер контекста в `with`-выражении. Введение ограничения на время вычисления значений исследуемой функции позволяет избежать нежелательных задержек при проведении дополнительной проверки. На практике, в случайной выборке точек из заданного отрезка могут возникать такие, в которых вычисление значения заданной функции может затянуться. Такой подход, однако, не спасает от проблем, связанных с ошибками, допускаемыми как авторами задач, так и студентами при получении ответа. Так, например, автор задачи может случайно задать такой промежуток проверки, что истинный ответ не будет на нем определен. С другой же стороны, ответ, полученный и введенный для проверки студентом, может оказаться достаточно сложным для вычисления, причем это далеко не всегда означает, что он неправильный. При возникновении подобных проблем нельзя допустить, чтобы алгоритм дополнительной проверки слишком долго (или вовсе бесконечно долго) перебирал точки проверки, отбрасывая одну за другой точки, в которых вычисление функции занимает слишком много времени. Поэтому необходимо ограничить также и количество отбрасываемых точек. Один из возможных путей решения этой проблемы — прекращать дополнительную проверку при возникновении «неудобной» точки и помечать ответ к задаче как *верный с некоторой вероятностью*, где оценка вероятности дается, исходя из количества уже проверенных точек и других параметров, заданных автором задачи. Такая задача может быть потом проверена непосредственно преподавателем, отвечающим за данный предмет, который может вынести окончательное решение — полностью зачесть или отклонить ответ. Этот подход особенно хорошо себя проявляет в реализации контрольных и прочих проверочных работ, где чрезвычайно важны высокая точность и справедливость оценки.

2.3 Выводы

Задача автоматической проверки ответов представляет собой ключевую и неотъемлемую часть интерактивной системы обучения, нацеленной в первую очередь на тренировочные задачи и поддержку практических занятий. Невозможность достоверной аналитической проверки можно обойти с помощью дополнительной численной проверки, но лишь частично. Анализ вероятности ошибки предложенного алгоритма дополнительной поточечной проверки показывает, что этот алгоритм может быть весьма успешно использован для проверки ответа при возникновении проблемных ситуаций. Нескольких десятков точек проверки на практике оказывается достаточно для того, чтобы сделать вероятность ошибки ничтожно малой — при условии, что неправильный ответ имеет относительно небольшое число нулей. Возникает закономерный вопрос: как можно оценить число нулей функции, которая еще до конца не известна (так как одна ее часть — неизвестный до непосредственно момента проверки ответ студента)? Здесь необходимо отметить, что под этой оценкой понимается характеристика самого истинного ответа с учетом ошибок, которые могут допустить студенты. Безусловно, для всякой наперед заданной функции можно подобрать другую такую, что их разность будет иметь сколь угодно большое число нулей на заданном отрезке (хотя и это в ряде случаев может оказаться затруднительным в элементарных функциях). Однако, невозможно произвести такой подбор, не зная исходной функции — истинного ответа. Если же неправильный ответ получен в результате настоящей ошибки (или нескольких ошибок), он с высокой вероятностью не совпадет с истинным ответом в чрезмерно большом числе точек. Вместе с тем, как показывает приведенный выше анализ, даже когда значения k исчисляются миллионами на достаточно узком промежутке, ста точек проверки достаточно для того, чтобы «удерживать» вероятность ошибки ниже e^{-2000}. Для того, чтобы прочувствовать, что это означает, можно сравнить это значение с вероятностью везения на экзамене с билетами. Пусть на некотором экзамене имеются экзаменационные билеты в количестве 20 штук, а испытуемый (студент) идеально подготовил ответ лишь на один из них. Вероятность его успеха на таком экзамене составит $\frac{1}{20}$. При этом подобная схема проведения экзаменов широко распространена и считается достаточно надежной. На этом фоне вероятность ошибки рассматриваемого алгоритма выглядит ничтожной. Действительно большое число нулей, хоть как-то сравнимое с количеством чисел с плавающей запятой внутри отрезка проверки, может возникать при наличии особенностей исследуемой функции внутри отрезка проверки. Именно поэтому необходимо предоставить автору задачи возможность настраивать параметры дополнительной проверки для своих задач, так как лишь он способен быстро и надежно проанализировать задачу и выбрать наиболее оптимальную комбинацию этих параметров.

Несмотря на то, что предложенный алгоритм демонстрирует хорошие показатели как в теории, так и на практике, потенциально проблемы могут все же возникать. Как было указано ранее, появление точек, в которых вычисление значения исследуемой функции занимает продолжительное время, может в конечном итоге потребовать вмешательства преподавателя, так как дополнительная проверка не сможет быть реализована в полной мере. Причины этих проблем могу лежать в каких-то особенностях реализации элементарных функций в используемой системе компьютерной алгебры, в ответе, полученном студентом, или в неправильно заданных параметрах проверки. В любом случае преподаватель должен иметь возможность скорректировать результат проверки. Для этого всякая попытка (конкретная задача, ответ студента и результат проверки), потребовавшая дополнительной численной проверки, помечается специальным образом и хранит оценку вероятности ошибки; результат проверки может быть изменен преподавателем. Таким образом, преподаватель имеет возможность вынести окончательный вердикт по каждому такому вопросу, а студент — оспаривать результаты автоматической проверки.

В ряде случаев у задачи имеется несколько характерных подходов к решению, которые приводят к синтаксически различным, но семантически равным результатам. При этом конечный ответ к такой тренировочной задаче в системе будет задан каким-то одним способом. В таком случае, если автору задачи заранее очевидны другие варианты ответа, он имеет возможность перечислить их. При этом в ходе проверки, в случае невозможности проверить правильность пользовательского ответа средствами системы компьютерной алгебры, в первую очередь будут проведены сравнения с другими вариантами ответа и лишь затем — численная проверка (если ни один из вариантов не может быть проверен).

Глава 3

Вычисление и оценка текущего уровня знаний

Важным аспектом обучения является оценка текущего уровня знаний. При этом необходимо определить набор навыков и компетенций, которыми обладает на данный момент испытуемый, в некоторой области знаний. Задачи такого рода возникают при необходимости удостовериться в приобретении необходимых знаний в результате обучения, при оценке возможности испытуемого стать участником какого-либо проекта или поступить на какой-либо курс, при корректировке программы обучения и во множестве других ситуаций.

В первом разделе данной главы обсуждается способ структурирования задач — установления определенного порядка, описывающего отношение вида родитель-последователь, получившего название «отношение предшествования». Подобная структура может быть получена естественным образом и составитель некоторого множества задач как носитель необходимых конмпетенций вполне может справиться с определением такой структуры. Наличие определения этой структуры на множестве тренировочных задач позволяет в автоматическом режиме строить предположения о пробелах в знаниях студента (если у того обнаружены затруднения при решении какой-то задачи) и предложения по дальнейшему развитию (если студент показывает хорошие результаты на некотором уровне).

Отношение предшествования оказывается тесно связанным с объектом изучения теории образовательных пространств. Эта связь позволяет использовать некоторые результаты данной теории в рамках разработанного подхода в организации задач. Раздел, посвященный описанию алгоритма вычисления скрытого состояния знаний студента, во многом основан на работе [24] и других работах данных авторов. Эксплуатация связи теории образовательных пространств с естественной структурой в разработанной системе EdLeTS позволяет применить на практике результаты этой теории в рамках системы.

3.1 Структура задач

Рассмотрим множество тренировочных задач в некоторой заданной области. Практически всегда эти задачи не существуют сами по себе, отдельно друг от друга, но связаны неким отношением, а именно — отношением предшествования. Наличие такого отношения на множестве задач означает, что более сложные задачи требуют как минимум знаний, необходимых для решения некоторых более легких задач. Будем говорить, что задача A *предшествует* задаче B, если для успешного решения задачи B необходимо (но, вообще говоря, не достаточно) обладать знаниями, которые требуются для решения задачи A. Записывать это будем следующим образом: $A \preceq B$. Легко видеть, что отношение предшествования, определенное на множестве тренировочных задач, является транзитивным (если $A \preceq B$ и $B \preceq C$, то $A \preceq C$) и рефлексивным ($A \preceq A$). Таким образом, отношение предшествования есть предпорядок (по определению). Задачу A будем называть *прямым предшественником* задачи B, если $A \neq B$ и не существует[1] задачи C такой, что $A \preceq C \preceq B$. Набор всех прямых предшественников заданной задачи будем называть ее *предусловием.*

[1] в рассматриваемом домене

Во множестве случаев отношение предшествования на домене задач может быть построено естественным образом. Так, например, не вызывает сомнений тот факт, что задачи на вычисление табличных производных предшествуют всем прочим задачам дифференциального исчисления, так как без этого знания невозможно полноценное изучение специальных техник дифференцирования. В некоторых же случаях, такое отношение может не быть столь очевидным. В качестве примера рассмотрим теорию пределов и дифференциальное исчисление. С теоретической точки зрения знание и понимание теории пределов является необходимым условием для изучения дифференциального исчисления хотя бы потому, что определение производной вводится через предел. С другой же стороны, на практике, студент может владеть техникой дифференцирования, не зная ровным счетом ничего о пределах. Однако, в конечном итоге, тем или иным образом всегда возможно либо определить отношение предшествования на некотором множестве задач, либо разбить исходной множество на несколько подмножеств и определить имеющие смысл отношения на каждом отдельно. Такого рода структурирование задач всегда должно оставаться на усмотрение преподавателя, который способен надлежащим образом разбить весь массив заданий на разделы и упорядочить в них задачи по отношению предшествования.

Тот факт, что отношение предшествования является преподрядком, обуславливает сильную связь с теорией образовательных пространств. Отношение такого типа может быть преобразовано к специального вида структуре знаний — математическому объекту, описывающему множество состояний уровня знаний обучающегося в некоторой области знаний. Такие структуры, в свою очередь, используются при построении алгоритма автоматической оценки уровня знаний, который реализует процедуру подбора задача и анализа ответов испытуемого для вычисления его состояния в заданном пространстве. Авторы теории образовательных пространств предлагают алгоритмический фреймворк для построения таких пространств посредством опроса преподавателей [см. 24]. В данном разделе будет предложен иной подход, основанный на анализе отношения предшествования, заданного на множестве задач. Данный подход оказывается более удобным и практичным в контексте рассматриваемого проекта. Отношение предшествования является достаточно естественным с человеческой точки зрения и может быть достаточно просто установлено преподавателем в небольшой группе задач. Его также достаточно просто хранить и изменять, а многие его свойства помогают при анализе поведения студента при решении задач.

Наличие такой структуры на множестве задач позволяет предлагать персонализированные подсказки и помощь студентам. Пусть, к примеру, некоторый студент систематически проваливает решение некоторой задачи. И в то же время замечено, что он не слишком хорошо себя показал при решении некоторых предшествующих задач (например, решил их достаточно мало, или процент успехов оказался невысок). В такой ситуации логично предположить, что у испытуемого остались какие-то проблемы с такими предшественниками текущей задачи, и следует предложить ему поработать над ними вновь. Такого рода подсказки могут быть вычислены автоматически при наличии отношения предшествования на множестве задач и статистики успехов и неудач по каждой задаче для данного студента. На этой идее основана функциональность системы автоматических подсказок по ликвидации пробелов: система анализирует прогресс студента и определяет потенциальные пробелы, предлагая затем соответствующие задачи студенту для дополнительной работы. В то же время, если, наоборот, некий студент показал себя очень положительно при выполнении некоторого набора задач, которые формируют предусловие какой-то новой задачи, последняя может быть ему предложена в качестве варианта дальнейшего развития. Данные идеи автоматической помощи по ликвидации пробелов и по дальнейшему развитию формируют функциональность, привносящую дополнительную персонализированность и адаптивность в систему — одни из наиболее важных свойств современных систем умного обучения.

3.2 Основные понятия теории образовательных пространств

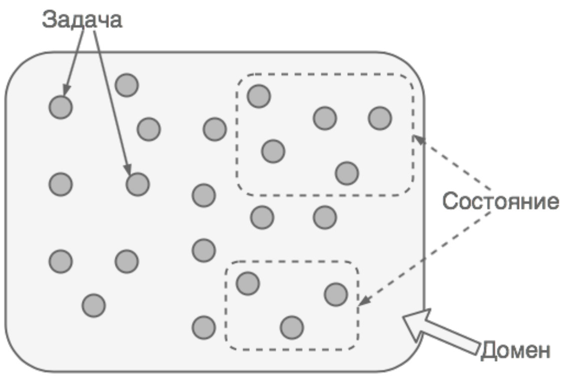


Рис. 3.1: Домен, задачи и состояния

Напомним, что под структурой знаний мы будем понимать пару $(Q, \mathcal{K})$, где Q — множество задач (домен), а $\mathcal{K} \subseteq 2^Q$ — множество состояний, причем такое, что как минимум $Q \in \mathcal{K}$ и $\varnothing \in \mathcal{K}$. Отдельно заметим, что множество $\mathcal{K}$ не обязано содержать все возможные подмножества Q (и, на практике, зачастую не содержит). Пример структуры знаний задан равенствами (3.1). Можно видеть, множество $\mathcal{K}$ удовлетворяет всем писанным выше требованиям и содержит не все подмножества домена.

$$\begin{aligned} &Q = \{a, b, c, d\} \\ &\mathcal{K} = \{\varnothing, \{a\}, \{a, b\}, \{a, c\}, \{a, b, c\}, \{a, b, c, d\}\} \end{aligned} \tag{3.1}$$

Структура знаний $(Q, \mathcal{K})$ называется *конечной*, если Q (и, соответственно, $\mathcal{K}$) конечно. Далее будем рассматривать конечные структуры. Будем говорить, что семейство множеств $\mathcal{K}$ *замкнуто* *относительно объединения*, если $\forall \mathcal{F} \subseteq \mathcal{K}$ объединение всех множеств $\mathcal{F}$ также принадлежит $\mathcal{K}$: $\bigcup\limits_{F \in \mathcal{F}} F \in \mathcal{K}$. Структура знаний, замкнутая относительно объединения называется *пространством знаний*. *Образовательным пространством* называется пространство знаний, обладающее свойствами гладкости и последовательности, описанным в разделе **Оценка уровня знаний**. Образовательное пространство можно определить немного иначе. Пусть $d(A,B) = |A \Delta B| = |A \backslash B| \cup |B \backslash A|$. Рассмотрим семейство множеств $\mathcal{F}$. Будем говорить, что оно *хорошо ранжировано* если $\forall K, F \in \mathcal{F}$, таких, что $K \neq F$, существует конечная последовательность $K = K_0, K_1, \ldots, K_p = F$ такая, что $K_i \in \mathcal{F}\ \forall i = \overline{1,p}$, и $d(K_{i-1}, K_i) = 1$ $1 \leq i \leq p$, и $d(K,F) = p$ [24, 40]. Тогда образовательное пространство можно определить как хорошо ранжированное пространство знаний [41, 40]. Последовательность $K_0, \ldots, K_p$ в определении выше называется *плотным путем*. Свойство хорошей ранжированности усиливает свойство гладкости обучения, а последовательность в определении свойства гладкости является частным случаем плотного пути.

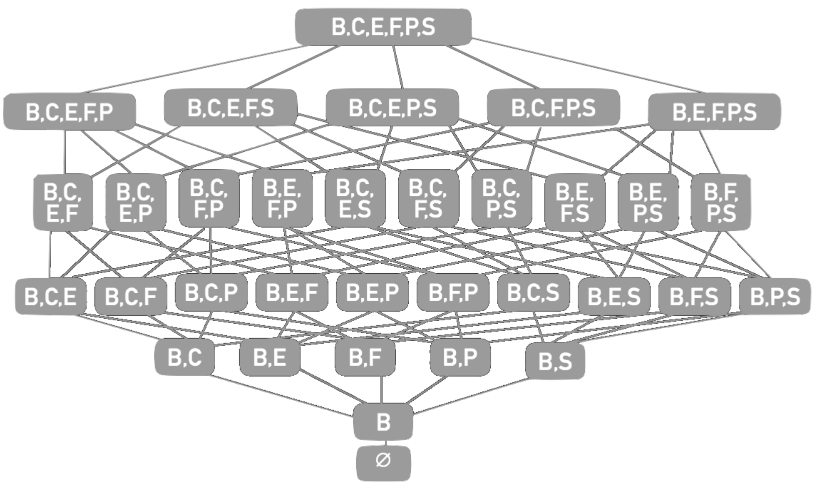


Рис. 3.2: Пример образовательного пространства для задач дифференциального исчисления

На Рис. 3.2 изображен пример образовательного пространства, построенного на домене задач дифференциального исчисления функций одной переменной. На диаграмме использованы следующие обозначения: B — задачи на вычисление табличных производных (производных от основных элементарных функций), C — задачи на вычисление производной композиции функций, E — логарифмическое дифференцирование, F — задачи на вычисление производной частного двух функций, P — задачи на вычисление производной произведений функций, S — задачи на вычисление производной суммы функций. Каждая вершина данного графа представляет собой состояние, в котором может пребывать обучающийся. Ребра графа демонстрируют возможность перехода из одного состояния в другое на один шаг. Из этого графа можно, например, заключить, что любому обучающемуся необходимо владеть техникой дифференцирования основных элементарных функций (задача B), чтобы претендовать на знание какой-либо другой задачи: все состояния, кроме пустого, содержат B.

Упомянутая ранее процедура оценивания работает с такими структурами знаний, описанными выше. Такая модель оказывается достаточно удобной для анализа и сама по себе является достаточно иллюстративной, однако, процесс ее построения «с нуля» оказывается достаточно трудоемким, а также имеются определенные трудности с хранением подобных структур. Поэтому особо важна для данной работы связь между отношениями предшествования и образовательными пространствами. Согласно теореме Биркгофа [42] имеет место взаимооднозначное соответствие между множеством всех предпорядков на множестве Q и пространствами знаний, замкнутых относительно операции пересечения, на том же множестве Q [24]. Этот факт, в свою очередь, позволяет использовать отношение предшествования как нативный формат для описания и хранения информации о структуре задач в заданном множестве и иметь при этом возможность использовать алгоритмы, предлагаемые теорией образовательных пространств. В следующей секции описана процедура построения пространства знаний по заданному отношению предшествования.

3.3 Построение пространства знаний

Пусть имеется некоторое множество задач, на котором задано отношение предшествования, определенное выше. Тогда для данного отношения, как показано, например, в [24], существует пространство знаний, замкнутое относительно пересечения, причем, единственное. Следовательно, это пространство можно построить. В данном разделе предлагается простой прямой алгоритм построения пространства знаний по заданному отношению предшествования на множестве задач. Данный алгоритм не является наиболее эффективным, так как в процессе своей работы может проводить множество лишних операций, но при этом он достаточно удобен для демонстрации и понятен.

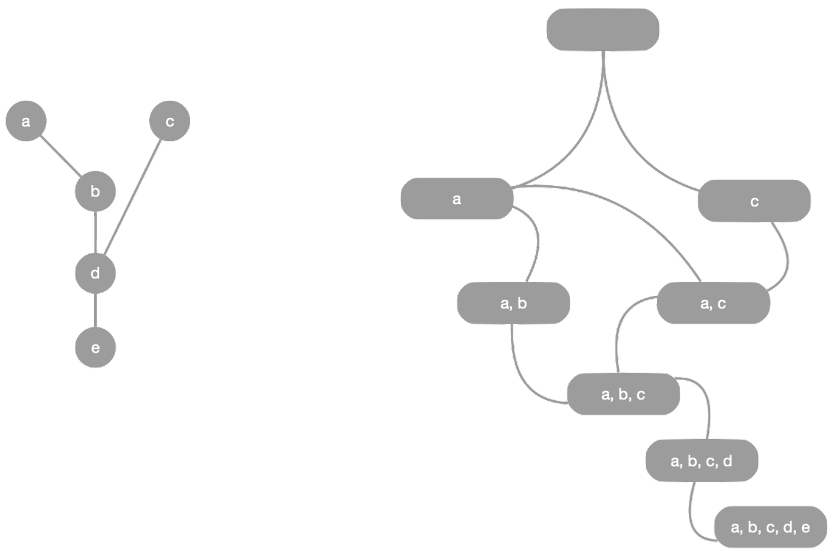
Для того, чтобы построить искомое пространство по заданному набору задач и отношению предшествования на нем, необходимо определить все состояния, возможные при таком отношении. Для этого можно пошагово расширять уже построенные состояния, добавляя к ним по одной задаче при условии соблюдения требований, накладываемых отношением. При этом за начальное состояние надлежит принять пустое множество. Иными словами, такой алгоритм проходит по каждому возможному пути обучения в данном пространстве, запоминая образуемые на этом пути состояния. Всякий такой путь начинается в пустом состоянии и заканчивается финальным состоянием, содержащим весь домен. Рассмотрим функцию, представленную в псевдокоде 2. Как и ранее, обозначим через Q весь домен задач. Пусть имеется функция $parent(q)$, которая для всякого $q \in Q$ возвращает список прямых предшественников задачи q, то есть, набор задач, непосредственно предшествующих q в отношении предшествования, заданном на Q. Функция $parent$ введена для упрощения изложения. Строго говоря, здесь необходимо лишь иметь доступ к структуре отношения, иметь возможность определить прямых предшественников любой задачи. Также для простоты изложения в алгоритме 2 предполагается наличие глобально определенных переменных Q (домен) и $\mathcal{K}$ (состояния).

Algorithm 2 Простой алгоритм построения пространства знаний по заданному отношению предшествования

```
function BuildStates(s)
    K.add(s)
    for q ∈ Q \ {s} do
        if parents(q) ⊆ s then
            BuildStates(s ∪ {q})
        end if
    end for
end function
```

Для того, чтобы построить пространство знаний, запустим `BuildStates`($\varnothing$). Тогда во множество состояний будет добавлено пустое множество. Далее, среди множества задач будет найдена первая ($\tilde{q}_1 \in Q$), не имеющая прямых предшественников, и для нее будет рекурсивно выполнена функция `BuildStates` с аргументом $\tilde{q}_1$. Таким образом в конструируемое множество состояний будут добавлены все одноэлементные множества, каждое из которых содержит задачу, не имеющую прямых предшественников. В свою очередь, для каждого такого одноэлементного состояния будут построены новые состояния, получаемые путем добавления одной задачи, согласно указанным правилам. На очередном вызове `BuildStates` от некоторого состояния s s добавляется в $\mathcal{K}$, затем ищутся такие $q \in Q \setminus s$, что s содержит прямых предшественников q в качестве подмножества. Процесс остановится, когда будут пройдены все итерации основного цикла первого вызова `BuildStates`. В результате, в $\mathcal{K}$ будут записаны все возможные состояния, порождаемые заданным отношением предшествования на Q.

В качестве иллюстрации рассмотрим пример отношения, заданного на домене $Q = \{a, b, c, d, e\}$ (Рис. 3.3a). Первый вызов



(а) Отношение предшествования

(б) Пространство знаний

Рис. 3.3: Пример отношения предшествования и соответствующего пространства знаний

`BuildStates` приведет к добавлению $\varnothing$ в множество состояний и двум рекурсивным вызовам — `BuildStates`$(\{a\})$ и `BuildStates`$(\{c\})$. Выполнение `BuildStates`$(\{a\})$ приведет к добавлению состояния $\{a\}$ и последовательным вызовам `BuildStates` $(\{a, b\})$ и `BuildStates` (a, c), так как, помимо b и c, все остальные элементы Q имеют более строгие «требования» (имеют других прямых предшественников). В конечном итоге, будет получено множество состояний $\mathcal{K} = \{\varnothing, \{a\}, \{c\}, \{a, b\}, \{a, c\}, \{a,b,c\}, \{a, b, c, d\}, \{a, b, c, d, e\}\}$. Соответствующий граф пространства знаний изображен на Рис. 3.3б.

3.4 Марковская процедура оценивания

Помимо рассматривавшихся в предыдущих главах задач тренировки на множестве специально подобранных примеров и проверки проработанности материала, важную роль в системах сопровождения практических занятий играет задача вычисления текущего уровня знаний студента в заданной области. Эта задача появляется, например, при оценке способности обучающегося двигаться дальше по программе, поступить на тот или иной факультатив или заняться каким-либо иным видом деятельности, требующим наличия определенного набора квалификаций. При этом, при сохранении надлежащего качества оценки уровня знаний, предпочтительно провести эту проверку наиболее оптимальным, коротким способом.

Авторами теории образовательных пространств были предложены алгоритмы для вычисления состояния студента в заданном пространстве, в том числе — *марковская процедура оценивания*[43, 24]. Существенным преимуществом от связи естественного отношения предшествования с пространствами знаний является возможность применения предложенного в рамках этой теории алгоритма вычисления состояния знаний студента. Данный раздел посвящен обзору марковской процедуры оценивания и ее реализации в рамках системы EdLeTS. Идея данного алгоритма заключается уточнении корректности оценки текущего состояния знаний испытуемого на основе его ответов. Так, в начале эксперимента можно задаться некоторым распределением вероятностей на множестве состояний структуры знаний: если об испытуемом не известно ровным счетом ничего, то следует принять равномерное распределение (он может прибывать в любом состоянии с равной вероятностью); если же нам доступна какая-то информация о знаниях студента, то мы можем оценить, какие состояния являются более вероятными. Получив от испытуемого ответ на очередной вопрос, можно уточнить текущее распределение, основываясь на корректности ответа: в случае правильного ответа — повысить вероятность состояний, содержащих этот вопрос, и понизить вероятность остальных состояний.

Соответственно, аналогичным образом следует поступить при получении неправильного ответа. Предлагаемый алгоритм описывает процедуру выбора вопросов и уточнения вероятностей на основе полученных ответов по такому сценарию.

Описываемый алгоритм представляет собой пошаговую процедуру, на каждом шаге которой выбирается очередная задача (или вопрос) и анализируется ответ студента. Эта процедура представляет собой эволюцию определенного марковского процесса. Для его описания необходимо дополнить предложенное ранее определение структуры знаний. Пусть $(Q,\mathcal{K})$ — некая структура знаний. Рассмотрим функцию $L:\mathcal{K}\to[0,1]$ такую, что $\sum\limits_{K\in\mathcal{K}} L(K)=1$. Функция L задает вероятностное распределение на $\mathcal{K}$. Тройка $(Q,\mathcal{K},L)$ называется *вероятностной структурой знаний*. Аналогичным образом вводится вероятностное пространство знаний.

Далее обозначим через Λ_+ множество всех положительных вероятностных распределений на $\mathcal{K}$. Пусть $L\in\Lambda_+$ — некоторое распределение на $\mathcal{K}$. Рассмотрим вероятностную структуру знаний $(Q,\mathcal{K},L)$. Пусть испытуемый находится в некотором состоянии $K_0\in\mathcal{K}$, которое не известно и не меняется в течение всей процедуры. Это состояние будем называть *скрытым состоянием* студента. Задача процедуры оценки — вычислить это состояние с достаточной степенью достоверности. Введем марковский процесс (R_n,Q_n,L_n), где:

- n — номер шага;
- Q_n — случайная величина (с.в.), описывающая вопрос, задаваемый испытуемому на шаге с номером n;
- $R_n\in\{0,1\}$ — с.в., задающая правильность ответа, полученного от испытуемого на заданный ему вопрос $q=Q_n$: $R_n=1$ — правильный ответ, $R_n=0$ — неправильный ответ;
- L_n — вероятностное распределение на $\mathcal{K}$; $L_n(K)$ для $K\in\mathcal{K}$ — с.в., задающая вероятность состояния K.

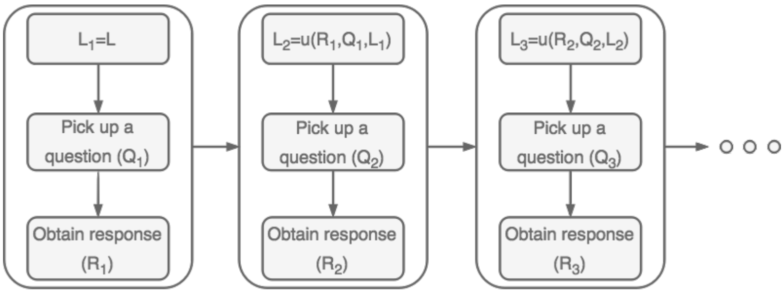


Рис. 3.4: Схема марковской процедуры вычисления неизвестного состояния знаний

Эволюция описываемого процесса зависит от трех составляющих: *правила выбора вопросов*, ответов испытуемого и *правила обновления*. На первом шаге ($n = 1$) выбирается начальное распределение $L_1 \in \Lambda_+$. Затем вопрос выбирается на основе правила выбора вопросов, которое задает вероятность выбрать тот или иной вопрос при текущем вероятностном распределении на состояниях. Правило выбора вопросов задается функцией

$$\Psi : \quad Q \times \Lambda_+ \mapsto [0,1] \tag{3.2}$$

$\Psi(q, L)$ задает вероятность выбрать (задать) вопрос $q \in Q$ при распределении вероятности состояний $L \in \Lambda_+$. После выбора вопроса, согласно описываемой процедуре, этот вопрос задается испытуемому. Получив ответ, мы обновляем распределение вероятностей на состояниях согласно *правилу обновления*:

$$\begin{gathered} u : \quad \{0,1\} \times Q \times \Lambda_+ \mapsto \Lambda_+ \\ L_{n+1} = u(R_n, Q_n, L_n) \text{ (почти наверное)} \end{gathered} \tag{3.3}$$

Таким образом, на каждом шаге происходит выбор вопроса на основе правила выбора вопросов, затем вопрос (или задача) задается студенту, после чего, на основе ответа студента, самого вопроса и текущего распределения вероятностей на состояниях, с помощью правила обновления уточняется распределение вероятностей. Схема эволюции процесса показана на Рис. 3.4. Авторы теории образовательных пространств и описываемого алгоритма оценки знаний студентов вводят следующие три аксиомы, выполнение которых гарантирует корректность работы алгоритма [24]:

$$\begin{aligned} &[\mathrm{U}]\colon \mathbb{P}(L_{n+1} \in B | W_n) = \mathbb{1}_B(u(R_n, Q_n, L_n)) \\ &\qquad \forall B \subseteq \Lambda_+, n = 1{,}2, \ldots \\ &\quad \mathbb{P}(L_1 = L) = 1 \\ &[\mathrm{Q}]\colon \mathbb{P}(Q_n = q | L_n, W_{n-1}) = \Psi(q, L_n) \quad \forall q \in Q \\ &[\mathrm{R}]\colon \mathbb{P}(R_n = \mathbb{1}_{K_0}(q) | Q_n = q, L_n, W_{n-1}) = 1 \end{aligned} \tag{3.4}$$

Здесь и далее под $\mathbb{1}_A(x)$ понимается индикатор множества A: $\mathbb{1}_A(x) = 1 \Leftrightarrow x \in A$, в противном случае $\mathbb{1}_A(x) = 0$. Через W_n в (3.4) обозначена история процесса до n-го шага:

$$W_n = ((R_n, Q_n, L_n), (R_{n-1}, Q_{n-1}, L_{n-1}), \ldots, (R_1, Q_1, L_1)). \tag{3.5}$$

Данные аксиомы накладывают ограничения на правила выбора вопросов и обновления, гарантируя осмысленность процесса с точки зрения оценки уровня знаний обучающегося. В частности, эти аксиомы гарантируют, что никакое состояние не будет иметь нулевую вероятность, а вероятность всякого состояния K будет повышаться при получении правильного ответа на вопрос $q \in K$ или неправильного — на вопрос $q \notin K$, а в оставшихся двух случаях — понижаться. Определенный таким образом процесс, удовлетворяющий аксиомам [U], [Q] и [R] будем называть *стохастическим процессом оценивания* для вероятностного пространства знаний $(Q, \mathcal{K}, L)$, параметризованным u, Ψ и K_0. Легко показать, что этот процесс является марковским.

Для того, чтобы эволюция данного процесса в действительности приводила к желаемому результату — определению скрытого состояния знаний испытуемого с некоторой степенью достоверности, — необходимо, чтобы вероятность скрытого состояния K_0 повышалась в течение работы соответствующей процедуры. Иными словами, для того, чтобы K_0 можно было вычислить с помощью заданного стохастического процесса оценивания (R_n,Q_n,L_n), требуется, чтобы было выполнено условие (3.6):

$$L_n(K_0) \xrightarrow[n\to\infty]{} 1 \text{ (почти наверное)} \tag{3.6}$$

Если условие (3.6) выполнено, то говорят, что состояние K_0 *вычислимо* (в оригинале — “uncoverable”) с помощью соответствующего стохастического процесса оценивания [24]. Выполнение этого свойства зависит от конкретного вида u и Ψ. Очевидно, что далеко не всегда это свойство будет выполнено. Авторы [24] описывают несколько видов правил обновления и правил выбора вопросов, для которых можно доказать выполнимость свойства (3.6). В настоящей работе мы ограничимся рассмотрением *выпуклого* правила обновления и *разделяющего* правила выбора вопросов[2]. Правило обновления называется выпуклым с параметром $\theta_{q,r}$ $(0 < \theta_{q,r} < 1)$, если оно $\forall K \in \mathcal{K}$, $L_n = l \in \Lambda_+$, $Q_n = q$, $R_n = r \in \{0,1\}$ имеет следующий вид:

$$\begin{gathered} u_K(r,q,l) = (1-\theta_{q,r})\cdot l(K) + \theta_{q,r}\cdot g_K(r,q,l), \text{ где} \\ g_K(r,q,l) = \begin{cases} r\frac{l(K)}{l(\mathcal{K}_q)}, & \text{если } K \in \mathcal{K}_q \\ (1-r)\frac{l(K)}{l(\mathcal{K}_{\bar{q}})}, & \text{если } K \in \mathcal{K}_{\bar{q}} \end{cases} \end{gathered} \tag{3.7}$$

Через $\mathcal{K}_q$ и $\mathcal{K}_{\bar{q}}$ обозначаются, соответственно, состояния, содержащие и не содержащие задачу q. u_K — проекция u на состояние K.

Правило выбора вопросов является разделяющим, если оно имеет вид:

$$\Psi(q, L_n) = \frac{\mathbf{1}_{S(L_n)}(q)}{|S(L_n)|}, \tag{3.8}$$

где $S(l) \subseteq Q$ для всякого распределения $l \in \Lambda_+$ задает множество вопросов, доставляющих минимальное значение выражению

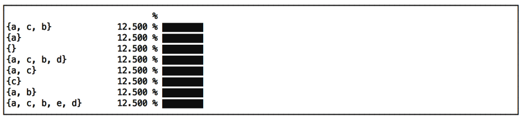
$$|l(\mathcal{K}_q) - l(\mathcal{K}_{\bar{q}})| = |1 - l(\mathcal{K}_q)| \tag{3.9}$$

[2] В оригинальной работе [см. 24] эти правила носят названия *convex updating rule* и *half-split questioning rule* соответственно.

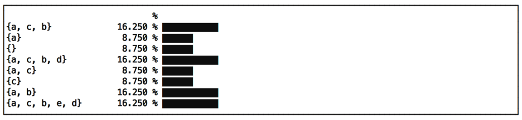
Проще говоря, разделяющее правила стремится выбрать очередной вопрос таким образом, чтобы суммарные вероятности множеств состояний $\mathcal{K}_q$ и $\mathcal{K}_{\bar{q}}$, получающиеся в результате разбиения множества всех состояний на содержащие и не содержащие q, оказались как можно более близки.

Доказано, что скрытое состояние вычислимо с помощью стохастического процесса оценивания с выпуклым правилом обновления и разделяющим правилом выбора вопросов [см. теорему 13.6.7 24]. В рамках разработки интерактивной системы обучения EdLeTS был разработан модуль, реализующий марковскую процедуру оценивания с константным выпуклым правилом обновления ($\theta_{q,r} = \theta = const$) и разделяющим правилом выбора вопросов. Данный модуль поддерживает задание параметра θ и определение начального распределения вероятностей на множестве состояний заданного пространства знаний. Модуль может быть использован автономно через специально разработанный интерфейс командной строки, с помощью которого можно считать описание пространства знаний из файла и запустить марковскую процедуру оценивания с заданными параметрами. На каждом шаге пользователю, управляющему процессом, помимо следующего вопроса демонстрируется также и текущее распределение вероятностей на состояниях заданного пространства.

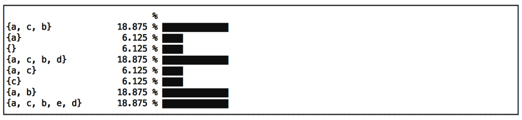
В качестве иллюстрации рассмотрим пример, введенный ранее в данной главе (Рис. 3.3). Пусть испытуемый находится в состоянии $K_0 = \{a, b, c\}$, скрытым от экзаменатора. Предположим также, что невозможно высказать какое-либо обоснованное предположение о состоянии студента в исследуемой области. Таким образом, подразумевается равномерное начальное распределение на множестве состояний. Всего имеется восемь состояний, следовательно, $L(K_i) = \frac{1}{8}\ \forall K_i \in \mathcal{K}$. Задача b доставляет минимум выражению (3.9). Если испытуемый находится в (скрытом) состоянии $\{a,b,c\}$, то он должен ответить на данный вопрос правильно. Первые шесть шагов моделирования работы марковской процедуры оценивания для примера Рис. 3.3 при $\theta = 0.3$ показаны на Рис. 3.5. На Рис. 3.6 продемонстрирована работа того же алгоритма со значением параметра $\theta = 0.7$.



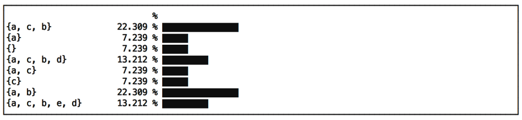
(а) Шаг 1. $q_1 =' b'$, $r_1 = 1$



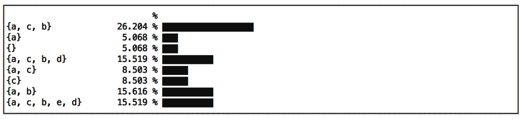
(б) Шаг 2. $q_2 =' b'$, $r_2 = 1$



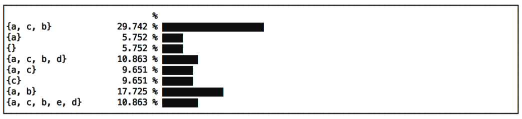
(в) Шаг 3. $q_3 =' d'$, $r_3 = 0$



(г) Шаг 4. $q_4 =' c'$, $r_4 = 1$



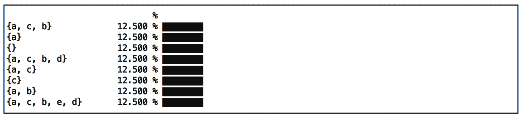
(д) Шаг 5. $q_5 =' d'$, $r_5 = 0$



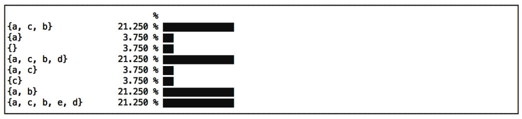
(е) Шаг 6. $q_6 =' b'$, $r_6 = 1$

Рис. 3.5: Шаги работы марковской процедуры оценивания с выпуклым правилом обновления ($\theta = 0.3$) и разделяющим правилом выбора вопроса. В основном поле каждой диаграммы перечислены состояния и указаны соответствующие им вероятности на данном шаге (до получения ответа и обновления распределения). Мнимый испытуемый отвечает в соответствии с состоянием $\{a, b, c\}$.

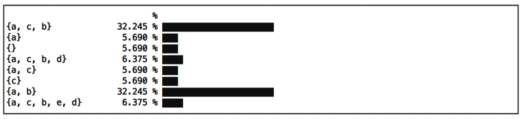
Можно видеть, что в обоих случаях вероятность скрытого состояния повышается по мере получения ответов на вопросы. При этом, уже на третьем шаге алгоритма с $\theta = 0.7$ удается добиться вероятности порядка 30%, тогда как при $\theta = 0.3$ это значение достигается после шестого шага. Параметр θ отвечает за то, какой вклад значение очередного ответа вносит в обновление вероятностей. Большие значения этого параметра к быстрому выделению скрытого состояния, но лишь в том случае, когда испытуемый действует в строгом соответствии со своим состоянием, то есть, отвечает всегда правильно на вопросы из своего состояния и неправильно —



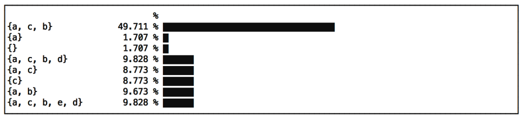
(а) Шаг 1. $q_1 =' b'$, $r_1 = 1$



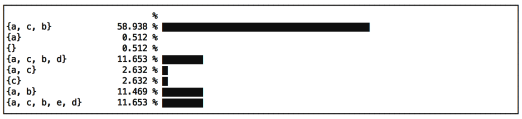
(б) Шаг 2. $q_2 =' d'$, $r_2 = 0$



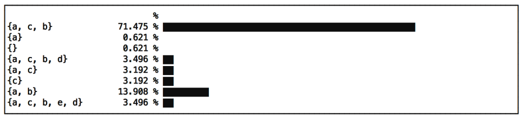
(в) Шаг 3. $q_3 =' c'$, $r_3 = 1$



(г) Шаг 4. $q_4 =' b'$, $r_4 = 1$



(д) Шаг 5. $q_5 =' d'$, $r_5 = 0$



(е) Шаг 6. $q_6 =' c'$, $r_6 = 1$

Рис. 3.6: Шаги работы марковской процедуры оценивания с выпуклым правилом обновления ($\theta = 0.7$) и разделяющим правилом выбора вопроса.

на все прочие. Малые же значения параметра обуславливают высокую «инертность» алгоритма — слабое влияние новых ответов на текущее распределение. Важно понимать, что, будучи живым человеком, испытуемый может допускать ошибки или, с другой стороны, догадываться до правильного ответа. При этом одна-две ошибки явно не могут служить показателем того, что испытуемый не владеет соответствующими знаниями. При $\theta = 1$ и равномерном начальном распределении история $\{(b, 1), (d, 0), (c, 0)\}$ (второе значение в каждой паре соответствует правильности полученного ответа на вопрос, заданный первым значением пары) приводит к вероятности состояния $\{a{,}b\}$, равной, практически, 100%. При этом вполне возможно, что неправильный ответ на вопрос c стал результатом простейшей оплошности, никак не иллюстрируещей отношения испытуемого с этой задачей. В то же время, в случае $\theta = 0$ исходное распределение вообще не будет изменяться. Выбором конкретного значения параметра θ можно управлять чувствительностью алгоритма к правильным и неправильным ответам. Такая настройка является специфической для каждой конкретной области и также для человека, проводящего оценку (от целей, которые он преследует, и от его собственного представления о том, что можно считать достаточным уровнем владения материалом). В более общем случае выпуклого правила обновления с параметром $\theta_{q,r}$, зависящем от вопросов и ответов, возможно произвести более тонкую настройку, определяя чувствительность алгоритма к правильному и неправильному ответу отдельно для каждой задачи из домена.

3.5 Выводы

Описанные в данной главе алгоритм построения пространства знаний по заданному отношению предшествования и алгоритм вычисления скрытого состояния испытуемого позволяют привнести в систему EdLeTS функциональность автономного определения возможностей пользователей-студентов, которая находит применение в таких задачах, как проведение промежуточного контроля (на предмет проверки достижения необходимого состояния), проверка наличия необходимых компетенций для входа в команду проекта, поступления на какой-либо факультатив и т.п., самостоятельный контроль студентами процесса своего развития в заданной области. Организация некоторого набора задач по введенному в данной главе отношению предшествования является достаточно естественным процессом для преподавателей. Занятые в подборе заданий и анализе успехов и неудач студентов в процессе обучения, они имеют представление о том, какие задачи являются более сложными, из чего состоят ко́мплексные задачи, как следует упорядочить материал и соответствующие задачи для изложения. Эти знания позволяют преподавателям, ассистентам или менторам курсов без особых затруднений установить предпорядок на некотором множестве задач. И этого оказывается достаточно для реализации множества весьма полезных возможностей системы. Как было показано в разделе **Структура задач**, наличие описанной структуры на заданном множестве задач позволяет строить автоматические подсказки для студентов, базируясь на информации об их успехах и неудачах. Так, в случае, если студент испытывает трудности при решении некоторой задачи, то велика вероятность, что корень проблемы находится где-то среди задач предыдущего уровня — предусловия данной. В противном случае проблема испытуемого связана с недостаточным пониманием теоретического материала, касающегося специфики именно данной задачи (типа задач), и здесь, если не исправляет ситуацию предоставленный ход решения, может помочь лишь преподаватель. Если же проблема все же связана с недостаточной проработанностью одной или нескольких задач-прямых предшественников данной, выданная системой на основе отношения предшествования подсказка поможет испытуемому устранить пробелы. Задачи в такой подсказке упорядочиваются по возрастанию отношения числа успехов к общему числу попыток (с нижней границей на общее число попыток). Таким образом, первыми в подсказке будут указаны наименее проработанные задачи. Следуя аналогичным рассуждениям, можно сформировать механизм предложений по дальнейшему изучению и развитию: если информация о достижениях студента показывает, что тот готов к изучению и решению каких-то задач новых типов, ему можно продемонстрировать соответствующее предложение. Показателем такой готовности для некоторой новой задачи служит хорошая проработанность всего предусловия данной.

Возможность преобразования отношения предшествования в пространство знаний позволяет использовать предлагаемый в рамках теории образовательных пространств алгоритм вычисления неизвестного состояния испытуемого. Как уже неоднократно упоминалось выше, отношение предшествования является достаточно естественным способом организации задач, чего нельзя сказать о структуре пространства знаний. Авторы теории образовательных пространств предлагают специальный алгоритм, позволяющий построить такое пространство на основе специального опроса эксперта [см., например, гл. 15 в 24]. Необходимость представления подобных алгоритмов обусловлена сложностью составления такой математической структуры вручную — тем более невозможно заставить заниматься такой работой преподавателей. В настоящей работе не ставится цели привести все задачи в системе к объектам теории образовательных пространств. В силу указанных выше свойств отношения предшествования и самой идее составления малых групп задач преподавателями, использование именно отношения предшествования представляется наиболее разумным. При этом в случае необходимости задействовать свойства различных образовательных структур, они могут быть легко составлены с помощью указанного алгоритма. Марковская процедура оценивания — один из таких примеров. Данный алгоритм показывает себя весьма убедительно при моделировании и тестирование, что позволяет рассчитывать на его успешное применение при массовом использовании системы.

Глава 4

Структура системы и описание ее компонентов

В данной главе описана структура разработанной в рамках настоящего диссертационного исследования системы поддержки практических занятий EdLeTS, обсуждается реализация идей и моделей, предложенных в предшествующих главах и то, как в совокупности предложенные подходы решают проблемы традиционного подхода к построению практических занятий по математическим дисциплинам и общие задачи персонификации и повышения эффективности образования. EdLeTS представляет собой веб-приложение, предоставляющее инструменты для создания и распространения тренировочных задач, сбора информации о прогрессе обучающихся, организации групп студентов и средства по оценке текущего уровня знаний с предложениями по ликвидации пробелов. Серверная часть EdLeTS разработана на языке Python 2.7 с использованием каркаса веб-приложений (веб-фреймворка) Django версии 1.7. При разработке системы был решен ряд технических вопросов, специфических для данной области. Разработанный продукт является уникальным и не имеет полных аналогов на рынке образовательного ПО.

4.1 Модель тренировочных задач

В терминологии MVT[1] — шаблона проектирования, используемого в Django для организации структуры приложения — под *моделью* понимается компонент системы, отвечающий за предоставление некоторых данных и их обработку. Модели в Django представляются специального вида классами, описывающими некоторый набор данных. Модели отображаются в базу данных (БД) посредством механизма *объектно-реляционного отображения* (англ. object-relational mapping, ORM), устанавливающего взаимооднозначное отображение объектов модели на множество записей соответствующей таблицы реляционной БД. Поля модели задач в EdLeTS описаны в таблице 4.1 (модель Task). В полях `name` и `description` содержаться название и некоторое описание задачи соответственно. Поле `permalink` содержит уникальную короткую строку для ссылки на задачу. Поле `author` ссылается на пользователя системы, разработавшего данную задачу. Значение этого поля формируется автоматически при составлении задачи пользователем. Поля `head`, `problem`, `solution` и `answer` хранят описание структуры шаблона задачи. В поле `head` содержатся декларативные описания динамических параметров задачи на языке SmallTask. Это специальное поле, содержимое которого в подавляющем большинстве случаев генерируется автоматически графическим дизайнером задачи на основе соответствующих действий пользователя (подробнее о дизайнере задач в 4.3). Поля `problem` и `solution` содержат текст, размеченный в нотации Markdown, с возможными вставками математических выражений, описанных в TEXи со вставками

[1] MVT — аббревиатура от «Model-View-Template» — представляет собой производный от более общего шаблона проектирования MVC («Model-View-Controller», «Модель-Представление-Контроллер»), предписывающего разделение приложения на три отдельных компонента — модель (отвечает за данные), представление (отображение данных пользователю, интерфейс) и контроллер (организует взаимодействие модели и представления). Различные варианты MVC широко распространены в разработке масштабных веб-приложений. Более подробное описание этого шаблона проектирования можно найти, например, в [44].

динамических параметров на SmallTask. Значение поля `answer` - выражение на языке SmallTask, описывающее конечный ответ задачи. Поля `max_check_points`, `pw_check_interval_from`, `pw_check_interval_to` и `zeros_estimate` отвечают максимальному количеству точек проверки, левой и правой границе интервала проверки и оценки на потенциальное число нулей у функции, задаваемой разностью ответов — параметрам алгоритма дополнительной поточечной проверки, запускаемого в случае невозможности достоверно проверить ответ средствами СКА (глава 2).

Таблица 4.1: Модель тренировочной задачи в EdLeTS

Task	
`name`	CharField
`description`	TextField
`permalink`	CharField
`author`	ForeignKey
`head`	TextField
`problem`	TextField
`solution`	TextField
`answer`	TextField
`max_check_points`	IntegerField
`zeros_estimate`	IntegerField
`pw_check_interval_from`	FloatField
`pw_check_interval_to`	FloatField
`created_at`	DateTimeField
`updated_at`	DateTimeField

Модель Task служит для хранения шаблонов тренировочных задач. Пользователи системы, играющие роль преподавателей, имеют возможность создавать новые задачи, редактировать или удалять задачи, авторами которых они являются. Пользователи, выполняющие роль студентов, взаимодействуют лишь с конкретными генерациями этих шаблонов — объектами класса ConcreteTask.

ConcreteTask, в отличие от Task, не является моделью (в терминологии MVT), а объекты этого класса не подлежат изменению и не отображаются в используемую реляционную БД. Генерация задачи Task с порождением соответствующего объекта ConcreteTask происходит при поступлении соответствующего запроса от пользователя (при вызове задачи). При этом конкретизируются значения всех динамических параметров запрошенного шаблона (посредством рандомизации по соответствующим множеством), а полученные таким образом конкретные постановка, решение и ответ инициализируют соответствующие свойства ConcreteTask. Каждый порождаемый таким образом объект ConcreteTask кэшируется на запущенном в том числе и для этих нужд кэш-сервере Memcached[2] по определенной схеме. Объект ConcreteTask может быть позже восстановлен из кэша для дальнейшей обработки. При запросе задачи пользователем, он получает лишь сгенерированное условие. Подобная схема обусловлена, в первую очередь, тем, что необходимо иметь возможность хранить посчитанный ответ и описание хода решения, начиная с момента запроса задачи и до ввода окончательного ответа пользователем, не предоставляя ему эти значения. Таким образом, конкретные ход решения и ответ — вре́менные данные, к которым необходим быстрый доступ с последующим удалением. Время жизни этих данных чрезвычайно мало, по сравнению со средним временем жизни, например, самих шаблонов тренировочных задач. Это обстоятельство делает кэширование наиболее предпочтительным способом хранения таких данных. Кэширование в Memcached построено на основе хеш-таблиц. Для ключей таблицы возможно установка ограничения на время жизни — время хранения, по истечении которого ключ уничтожается. Наличие такого порога позволяет избежать пустого «простоя» задач и переполнения памяти. В случае, если пользователь не введет ответ до истечения срока давности ключа, попытка будет считаться проваленной.

[2] Memcached — сревис кэширования данных (то есть, сохранения данных в буфере с быстрым доступом) в оперативной памяти.

За учет попыток решить ту или иную задачу отвечает соответствующая модель — TaskSolvingAttempt. Данная модель хранит ссылку на оригинальную задачу (шаблон), ссылку на пользователя, осуществившего запрос (то есть, предпринявшего попытку решения), время начала попытки, время ее окончания (если попытка уже завершена) и результат — успех или неудача. Набор таких данных для каждого пользователя позволяет следить за его прогрессом и строить на их основе специфические критерии оценки качества проработки материала. Доступ к персональной статистике пользователя, помимо самого пользователя, ограничен.

4.2 Организация пользователей

Пользователи в EdLeTS представляются соответствующей моделью, хранящей базовый набор стандартных полей с информацией о человеке. Основной структурной единицей группирования пользователей в системе EdLeTS являются *учебные комнаты* (УК) — объекты класса-модели Classroom. Учебные комнаты включают группу студентов и группу преподавателей, один из которых является учредителем и основным администратором «комнаты». Преподаватели учебной комнаты, в зависимости от значения модификатора доступа, могут добавлять и удалять студентов, других преподавателей, а также задачи и группы задач. Учредитель учебной комнаты располагает всей полнотой прав. Модификатор доступа УК управляет способом добавления студентов в УК и видимостью ее в общем пуле. В EdLeTS определены шесть типов УК:

- Публичная — любой пользователь может присоединиться в качестве студента, УК отображается в общем пуле;
- Защищенная — только преподаватели могут приглашать студентов, УК видна в общем пуле;

- Защищенная скрытая — только преподаватели могут приглашать студентов, УК скрыта;
- Частная — лишь учредитель может приглашать участников, УК видна в общем пуле;
- Частная скрытая — только учредитель может приглашать участников, УК скрыта;
- Закрытая — никто не может присоединиться, УК не отображается в пуле, никто, кроме учредителя, не может посещать УК (используется во время разработки контента УК).

Задачи в УК организуются в иерархические структуры по директориям. Все студенты УК получают доступ к ее задачам и могут решать их как в режиме свободного треннинга, так и в рамках заданий.

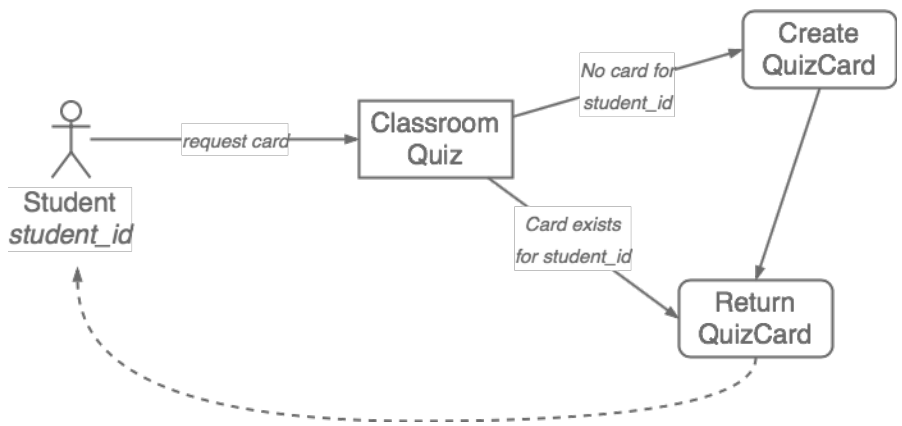


Рис. 4.1: Диаграмма запроса задания контрольной

В рамках учебных комнат EdLeTS предоставляется функциональность проведения контрольных работ. Контрольные работы, представленные моделью Quiz, позволяют в рамках заданной УК определить список задач (шаблонов), участвующих в контрольной, а также некоторые прочие поля, среди которых — время окончания контрольной (день и время, когда доступ к контрольной закрывается). При поступлении первого запроса доступа к заданиям контрольной работы от студента УК происходит генерация всех задач контрольной и порождается объект QuizCard — вариант контрольной работы, закрепленный за данным студентом. С этих пор объект QuizCard оказывается закрепленным за студентом и другого варианта он не получит (Рис. 4.1). Каждая задача в варианте контрольной представляет собой специальный объект — объект модели QuizCardTask, — который хранит постановку, решение и два ответа задачи (пользовательский и посчитанный системой). Студент УК, получив вариант контрольной, заполняет поля ответов для каждой задачи, после чего отправляет свою работу на проверку. При этом в БД сохраняются все данные о контрольной и ответах студента. Время жизни таких записей в БД соответствует времени жизни УК, в рамках которой была проведена контрольная. Необходимость так подробного сохранения и строгого учета результатов контрольных работ обусловлена их природой и основным назначением — проверить накопившиеся знания и оценить их. Результаты контрольных работ зачастую влияют на различные формальные показатели успеваемости студентов (накопленная оценка, рейтинг и т.п.), поэтому проведение подобных работ должно подчиняться строгому регламенту, а результаты должны быть доступны на протяжении всего курса и должны допускать апелляцию.

Пользователи, выполняющие роль преподавателей в некоторой УК, имеют доступ к информации о прогрессе, успехах и неудачах студентов этой УК. Каждый студент имеет доступ к своей персональной статистике. Пример персональной статистики студента в режиме свободной тренировки на задачах по математическому анализу показан на Рис. 4.2. Рис. 4.3 демонстрирует пример графиков успехов и неудач студента в УК. Наблюдения за поведением кривых прогресса каждого студента позволяют преподавателям выстроить более полную картину происходящего как для каждого студента

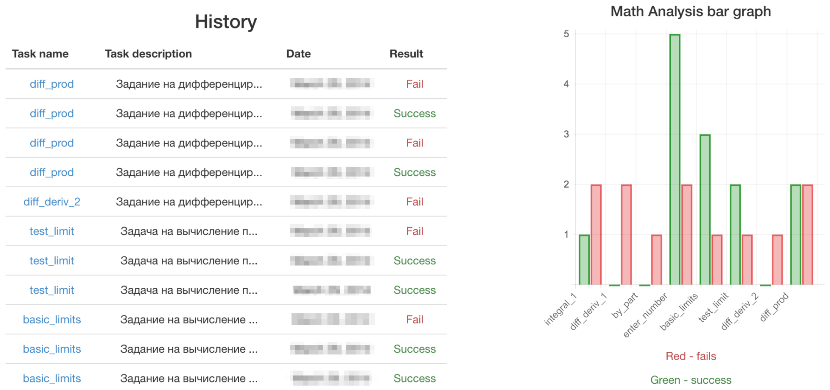


Рис. 4.2: Пример персональной статистики решения задач студента

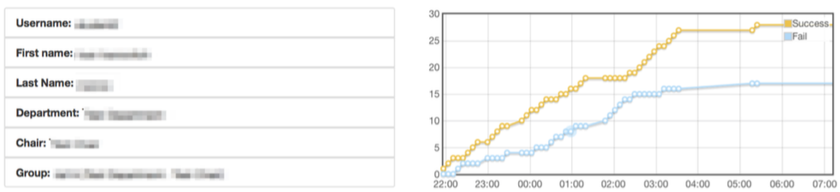


Рис. 4.3: Пример графика прогресса студента в УК

персонально, так и для группы в целом. Персональные рекомендации в случае обнаружения тех или иных проблем у студента могут заключаться в предложениях уделить внимание соответствующим разделам теории и в рекомендации дополнительной отработки соответствующих задач. При этом, модель шаблонов тренировочных задач позволяет предлагать более информативные меры оценки качества проработанности материала. Традиционно преподаватели назначают некий объем конкретных задач, считая, что *в среднем* этого объема достаточно для поддержания допустимого уровня усвоения соответствующих техник и методов. Вместе с тем, очевидно, что каждый обучающийся может нуждаться в своей личной схеме отработки: иному может понадобиться прорешать в несколько раз больше задач на заданную тематику, чем большинству других его одногруппников, для достижения высокого уровня усвоения.

Это может зависеть от огромного набора факторов — как постоянных, так и вре́менных и случайных. Более показательной мерой для оценки степени проработанности какой-либо темы может служить отношение правильно решенных задач к общему числу попыток. При этом стоит отметить, что необходимо ввести некоторый нижний порог на общее число решенных задач: один-два примера, пусть и успешных, мало говорят об уровне усвоения материала.

4.3 Дизайнер тренировочных задач

В разделе, посвященном описанию языка определения динамических параметров тренировочных задач, SmallTask, приводились основные требования к такому языку. Среди этих требований указывались максимальная простота, отсутствие лишних конструкций, минимизация порога входа и необходимости изучения документации. Все это обусловлено необходимостью сделать процесс составления задач в EdLeTS наиболее простым и удобным. Успех попыток внедрения системы в образовательный процесс в различных учебных заведениях во многом зависит от того, насколько легко пользователем будет привыкнуть к этому новому инструменту и адаптировать учебные программы под его использование. При этом процесс составления тренировочных задач играет очень важную роль, так как для многих преподавателей важно иметь возможность обучать своих подопечных на тех задачах, на которых им это удобно или привычно. По этой причине процедура освоения основных принципов построения описаний тренировочных задач и непосредственно составление этих описаний должна быть наиболее безболезненной и интуитивно понятной. Этой цели нельзя добиться, предложив пользователям исключительно возможность запрограммировать поведение динамических параметров задач — пусть и на простом и понятном языке. Практика показывает, что для большинства людей (в частности, это касается и преподавателей) гораздо более предпочтительным является графический интерфейс взаимодействия с каким-либо ПО. При этом многие предпочитают свести к минимуму необходимость вообще что-либо печатать или как-то иначе взаимодействовать с клавиатурой. В связи с этим обстоятельством, EdLeTS предлагает специальный *дизайнер тренировочных задач* с графическим интерфейсом, позволяющим пользователям избежать необходимости вникать в конструкции языка описания динамических параметров.

Дизайнер задач строится на принципах максимального удаления пользователя от языковых конструкций, при сохранении эффективности и скорости описания динамических параметров. Дизайнер задач предлагает пользователю три вкладки — для определения постановки задачи, хода ее решения и ответа. В боковой панели располагаются элементы, отвечающий определенным пользователем динамическим параметрам. Для добавления динамического параметра пользователь открывает диалоговое окно с помощью соответствующей кнопки, в котором ему доступны различные варианты динамических параметров, область ввода (адаптирующаяся под выбранный тип), панель математических выражений и область предпросмотра. Варианты задания динамических параметров отвечают типам языка SmallTask, разнообразным способам рандомизации и некоторым часто используемым конструкциям. Задав все необходимые значения, пользователь подтверждает создание параметра, в результате чего формируется соответствующее выражение на языке SmallTask, которое добавляется в область заголовков задачи (поле `head`, описанное выше в разделе, посвященном модели тренировочных задач). Графические объекты, отвечающие динамическим параметрам, в боковой панели дизайнера можно менять местами, редактировать и удалять. Добавление динамического параметра в текст задачи может быть осуществлено перетаскиванием соответствующего графического объекта в нужное место в тексте задачи (постановки, хода решения или ответа). При этом простое перетаскивание приводит к включению соответствующего выражения в текст в исходном виде. Зажав предварительно специальную клавишу (`Cmd` — для компьютеров под управлением OS X или MacOS, `Ctrl` — для компьютеров под управлением Windows), пользователь может перетащить объект для добавления динамического параметра в вычисленном виде. Пример с описанием хода решения тренировочной задачи на вычисление производной частного двух тригонометрических функций представлен на Рис. 4.4.

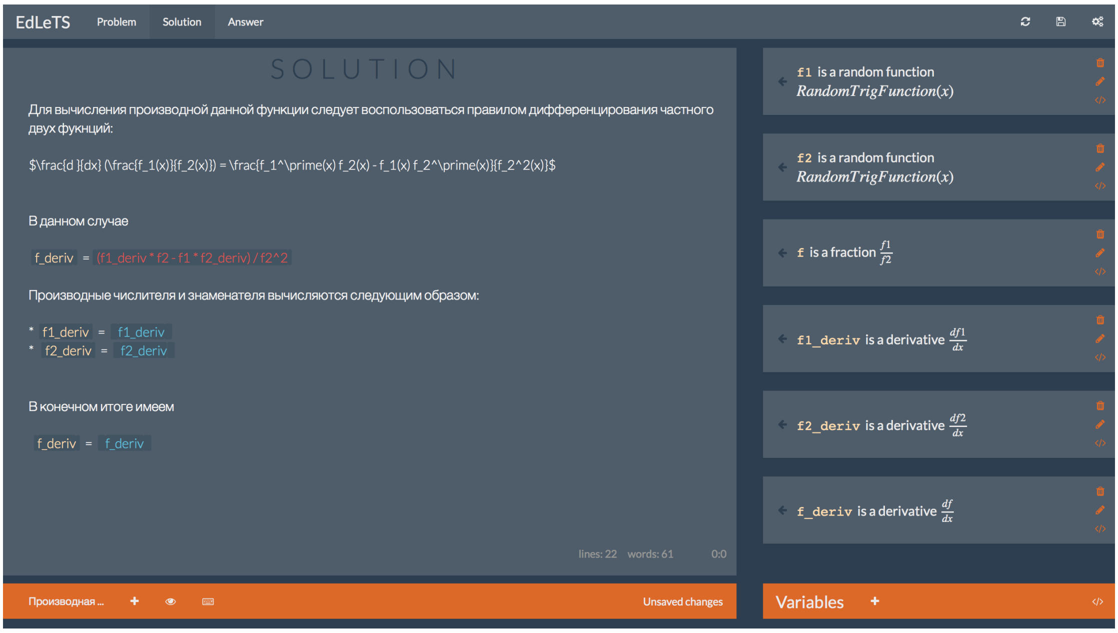


Рис. 4.4: Пример хода решения задачи в дизайнере задач

При разработке дизайнера тренировочных задач был использован интерфейсный фреймворк RactiveJS[3]. RactiveJS — MMVM-фреймворк[4] на Javascript, организующий сборку элементов страницы

[3] Официальный сайт проекта: `https://ractive.js.org`

[4] MMVM — аббревиатура от Model-View-ViewModel — шаблон проектирования, используемый для разделения модели данных и представления. В отличие от упомянутых ранее MVC и аналогичных шаблонов проектирования, данный подразумевает связывание данных с визуальными элементами в обе стороны; при этом внесение изменений в модель и преставление должны быть согласованы, и такая связь данных и представления, строго говоря, противоречит концепции MVC [45].

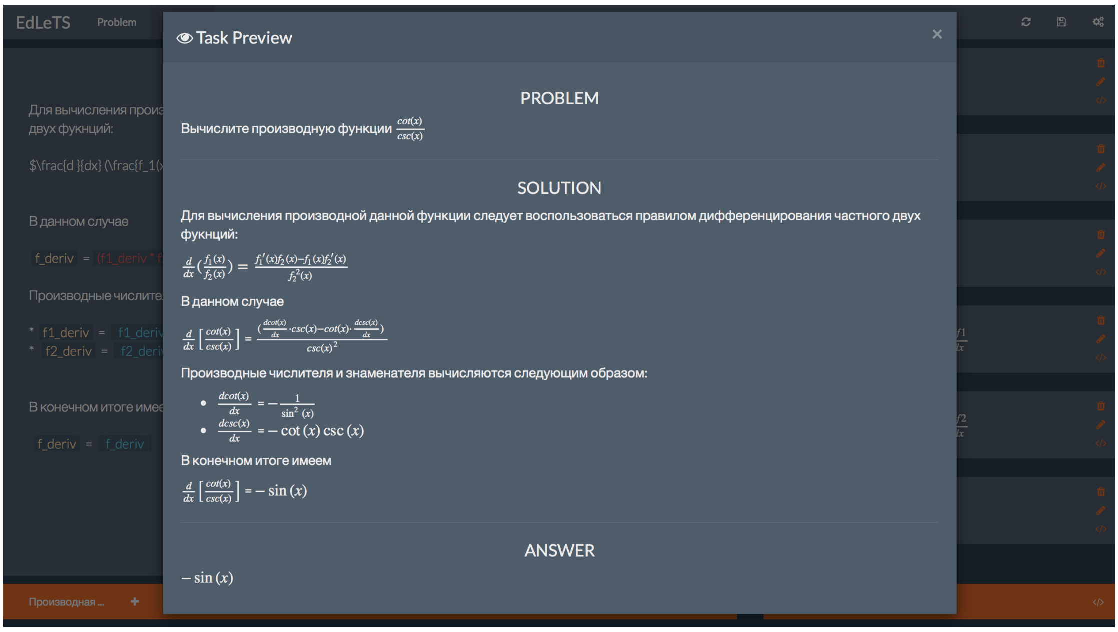


Рис. 4.5: Предпросмотр случайной генерации задачи в дизайнере задач

(или всей страницы) из шаблонов, связывая представление с данными, определенными в модели. Две основные составляющие приложения на Ractive — модель (данные) и шаблоны. В модели описываются данные приложения и их поведение, определяются вычисляемые параметры (зависимые от основных данных). Шаблоны представляют собой декларативное описание представления приложения. При порождении объекта Ractive происходит сборка представления (рендеринг, render) в указанный элемент DOM[5] на странице. Предоставляемая данным фреймворком функциональность двусторонней связности (англ. two-way binding) позволяет легко поддерживать модель в актуальном состоянии при изменениях, вносимых пользователем через интерфейс. Каждый динамический параметр в

[5] DOM — Document object model, объектная модель документа, единый интерфейс доступа к содержимому XML/HTML-документов. В DOM документ представляется деревом, узлами которого являются элементы, отвечающие каждому тегу в документе, или текстовые узлы.

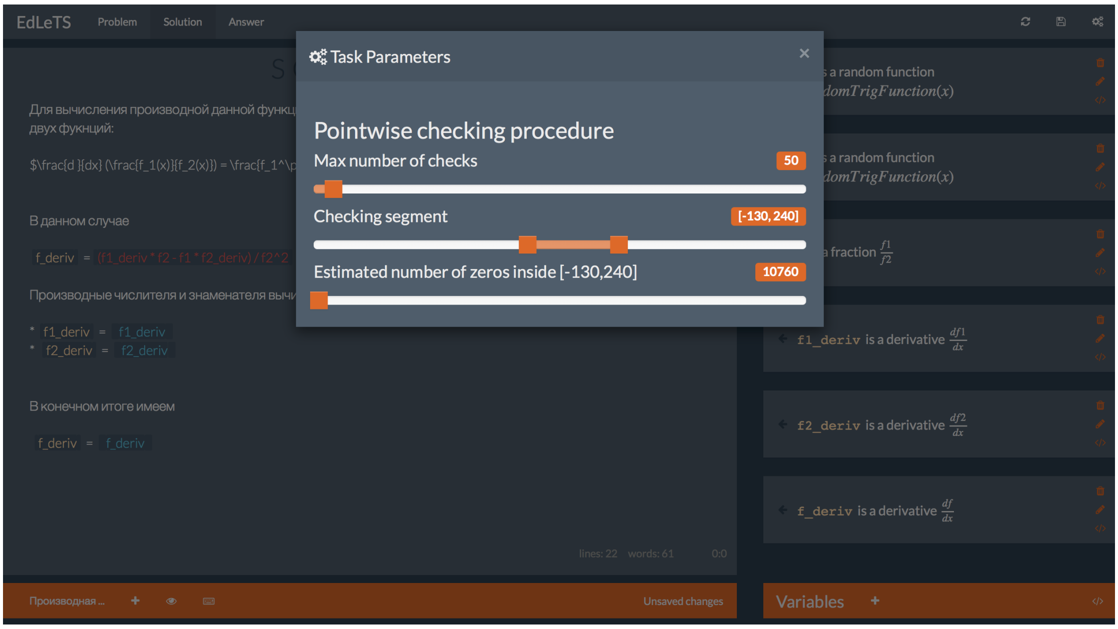


Рис. 4.6: Настройка параметров дополнительной поточечной проверки в дизайнере задач

этой схеме представлен самостоятельным компонентом — объектом Ractive, определяющим собственное отображение, внутреннюю модель, методы и вычисляемые параметры, зависимым от основного объекта, отвечающего за весь дизайнер. Для каждого типа динамических параметров разработан отдельный тип компонентов, отражающий специфику инкапсулируемых данных и реализующий необходимые методы. Также отдельные компоненты были разработаны для различных типов ввода динамических параметров, областей предпросмотра и некоторых других технических объектов. В такой реализации поле `head` всей задачи становится вычисляемым и всегда зависит от множества объявленных динамических параметров: его значение вычисляется как конкатенация выражений, определяемых компонентами динамических параметров.

Использование RactiveJS (как и аналогичных фреймворков) позволяет избежать прямых манипуляций с DOM при разработке интерфейса. При правильно построенных модели и представлении операции изменения представления могут быть практически полностью отданы на откуп внутренним инструментам Ractive, а требуемого эффекта можно добиться лишь манипулируя данными и самим объектом Ractive. Использование сложной структуры с самостоятельными компонентами и повторяемыми областями, привносит масштабируемость и возможность легко и быстро изменить или исправить разработанный дизайнер. При этом программный интерфейс самого RactiveJS является весьма удобным и достаточно простым, что позитивно сказывается на скорости и качестве разработки.

Дизайнер задач также включает специальную реализацию анализатора SmallTask на Javascript. Для построения этого анализатора был использован автоматический генератор анализаторов ANTLR версии 3 с целевым языком Javascript. В результате была разработана библиотека smalltaskjs, включающая транслятор выражений SmallTask непосредственно в TEX, а также интерпретатор, преобразующий входное выражение на SmallTask в дерево в виде простейшего объекта Javascript. Smalltaskjs предоставляет возможности, позволяющие организовать моментальный предпросмотр текущего состояния выражений и корректно преобразовывать исходные данные динамических параметров задач в Ractive-компоненты дизайнера.

Область определения текстовых описаний — текстовый редактор — построена на базе библиотек showdown.js[6] и SimpleMDE[7] в связке с CodeMirror[8]. Полученный редактор поддерживает подсветку синтаксиса и графическое оформление текста (например, подстройка размеров шрифтов для заголовков), распознавание и

[6] Конвертер Markdown в HTML. Проект на GitHub: `https://github.com/showdownjs/showdown`

[7] Текстовый редактор Markdown. Официальный сайт проекта: `https://simplemde.com`

[8] Базовый текстовый редактор на javascript, поддерживающий подсветку синтаксиса, нумерацию строк, темы и многие другие функциональные возможности. Официальный сайт проекта: `https://codemirror.net`

мелкий анализ выражений SmallTask (подсветка не определенных переменных, всплывающие подсказки).

Помимо описанных выше функций, дизайнер задач также предусматривает возможности запроса случайной генерации разрабатываемой задачи (Рис. 4.5), настройку параметров алгоритма дополнительной поточечной проверки (Рис. 4.6).

4.4 Выводы

В представленной в настоящей главе системе поддержки практических занятий EdLeTS реализованы подходы и идеи, представленные в предыдущих главах. Модель тренировочных задач и связанные с ней механизмы предоставляют пользователям системы функциональность автоматической генерации и проверки тренировочных задач. При этом реализуется ключевая идея тренировки на таких примерах: обучающийся в процессе тренировки раз за разом получает все новые примеры одно и того же типа, исследуя конкретизированные ходы решений (в случае необходимости), отрабатывая таким образом необходимый навык. Собираемая системой параллельно информация о прогрессе каждого студента позволяет им самим и их преподавателям анализировать успехи, более четко представлять персональные проблемы и сильные стороны каждого. Функциональность учебных комнат является необходимым, но чисто техническим аспектом организации пользователей. По сути своей, учебные комнаты сродни группам в социальных сетях и курсам или контекстам в различных LMS.

Описанный дизайнер задач позволяет пользователям практически полностью избежать контакта с синтаксическими конструкциями языка SmallTask. Не смотря на простоту и интуитивность этого языка, в подавляющем большинстве случаев использование его напрямую будет излишним и также вызывает недовольство составителей задач (инструкторов, преподавателей и др.) Возможность орудовать практически исключительно мышью и оперировать графическими представлениями объектов является чрезвычайно важной для разрабатываемой системы, так как значительно упрощает процесс создания описаний тренировочных задач, а также позволяет пользователям-составителям быстрее и глубже вникнуть в суть происходящего. При этом также важно, что описанная структура дизайнера задач допускает простое и быстрое расширение заложенной в него функциональности в рамках компонентов для языковых конструкций SmallTask. Все вместе представляет собой удобный инструмент создания описаний тренировочных задач, лишенный описанных ранее недостатков аналогичных систем, задействующих традиционные языки программирования.

Глава 5

Интеграция в учебный процесс

Данная глава посвящена вопросам интеграции системы EdLeTS в образовательный процесс в рамках практических занятий по математическим дисциплинам. Описываемые здесь подходы и результаты являются достаточно общими, однако, сосредоточены на внедрении в начальный курс математического анализа, и приводимые здесь результаты исследований касаются именно этого предмета. Полноценное использование системы на практических занятиях требует перестройки логики курса с учетом новых возможностей, которые эта система открывает перед преподавателями и студентами. Отдельного внимания заслуживают вопросы места EdLeTS в системе оценивания результатов работы студентов. Вопросы эти носят не только практический, но и этический характер, поэтому при внедрении подобной технологии необходимо соблюдать особую осторожность. Результаты анализа процесса интеграции системы, приведенные ниже в этой главе, можно рассматривать в качестве общих рекомендаций по включению систем поддержки практических занятий по математическим дисциплинам, позволяющих избежать возможных негативных последствий.

5.1 Внедрение в образовательный процесс

Разработанный в рамках настоящего диссертационного исследования подход к организации процесса тренировки на специальных задачах и реализующая его система избавляют пользователей от недостатков традиционного подхода к этому вопросу и также предлагают некоторые дополнительные функциональные возможности, реализация которых ранее была невозможна. Однако внедрение системы в образовательный процесс сопряжено с рядом затруднений, эффект которых необходимо свести к минимуму во избежание отторжения и негативной оценки.

Сам метод оттачивания вычислительных навыков на множестве специальным образом составленных задач хорошо себя зарекомендовал со времен введения такой практики и не нуждается в принципиальных изменениях. Основным недостатком традиционного подхода к организации процесса такой тренировки и практических занятий в целом является чрезмерность затрачиваемого времени на пустую механическую работу, что, в свою очередь, не позволяет посвятить достаточное время тем аспектам обучения, которые действительно требуют коллективной работы в группах с активным участием преподавателей. Среди таких аспектов — разнообразные подходы и методы доказательства теорем, анализ математических объектов и вывод их свойств, основы исследовательской деятельности, в рамках которой вновь открывают какие-либо характеристики и свойства объектов, формулируют и обосновывают собственные утверждения. Интеграция в образовательный процесс системы EdLeTS позволяет освободить время практических занятий для такого рода деятельности, так как EdLeTS способна взять на себя значительную часть рутинной работы. При этом, разумеется, необходимо разработать программу этих новых занятий — тех аспектов, изучение которых займет высвободившееся время. Таким образом, программа учебной дисциплины, привлекающей помощь EdLeTS, должна быть переработана с учетом открывающихся возможностей.

При внедрении системы в образовательный процесс в рамках некоторой дисциплины, необходимо в первую очередь очертить круг тем и компетенций, которые надлежит покрыть с использованием EdLeTS. В ряде случаев необходимо провести анализ существующих примеров конкретных тренировочных задач по каждой из выделенных тем и произвести обобщение, получив в результате общие структуры задач. Важно определиться с формой тренировок и способами контроля результатов студентов. Предоставляемая EdLeTS функциональность контрольных работ позволяет достаточно легко и просто организовать проверку на некотором наборе задач, а хранимая история прогресса каждого студента является сама по себе достаточно показательной. Тем не менее, вопрос о недобросовестных студентах и различного рода махинациях обучающихся остается за рамками данной работы. Так, например, нельзя исключать ситуацию, в которой некоторую работу с аккаунта некоторого студента выполняет другой человек. Подобные проблемы могут быть решены проведением проверочных работ в дисплейных классах или с помощью каких-то дополнительных мер. Сама система EdLeTS не ставит своей задачей представление надежного способа защиты от мошенничества, поэтому, в зависимости от задач, которые ставит перед собой ВУЗ или преподаватель, такие вопросы должны решаться отдельно.

5.2 Тестирование основной модели

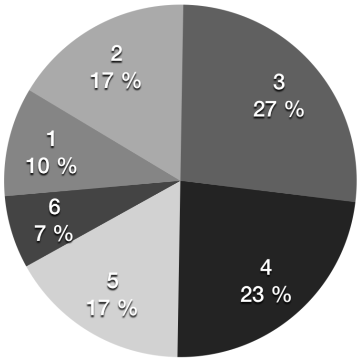
В рамках исследования эффектов внедрения системы в образовательный процесс и апробации предложенной модели было проведено несколько экспериментов. Результаты, изложенные в данном разделе, представлены также в [26]. Первое тестирование основной функциональности системы EdLeTS было проведено на группе из 30 студентов, проходящих курс по математическому анализу и теории функций комплексного переменного. Целью этого теста стала оценка эффективности предлагаемого подхода и установление возможности применения его на практике для развития вычислительных навыков у обучающихся. Для проведения пробных исследований были избраны четыре небольших темы из разных разделов дисциплины (Таблица 5.1).

Таблица 5.1: Темы, использованные для тестирования EdLeTS

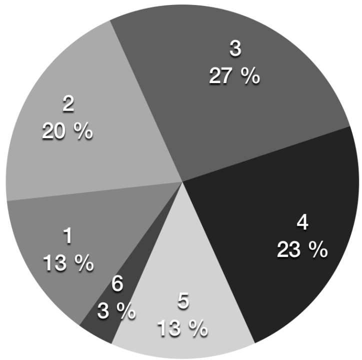
Тема	**Описание**	**Пример задачи**
Вычисление пределов с использованием замечательных пределов и «теоремы о двух милиционерах»	Задачи призваны развить в обучающемся способности находить и определять возможности использования замечательных пределов при решении конкретных задач и/или определять границы функции (мажоранту и миноранту), позволяющие применить соответствующую теорему к решению задачи.	Вычислить предел функции $\lim\limits_{x\to\infty} \frac{5x+\sin 2x}{2x-1}$
Основные правила дифференцирования	Задачи на основные правила дифференцирования элементарных функций, их сумм, разностей, отношений, произведений и композиций.	Вычислить производную функции $2^{\sin x}$
Интегрирование по частям	Задачи на отработку правила интегрирования по частям: $\int u(x)v'(x) = u(x)v(x) - \int u'(x)v(x)dx$.	Вычислить неопределенный интеграл $\int \sec^3 x dx$

Комплексное интегрирование с помощью теоремы о вычетах	Задачи на использование техники вычисления криволинейных интегралов от аналитических функций вдоль замкнутой кривой, с использованием теоремы о вычетах: $\oint_C f(z)dz = 2\pi i \sum_{k=1}^{n} Rez(f, z_k)$, где C — положительно ориентированная простая кривая, $z_1, \ldots, z_n$ — изолированные особые точки функции f в области, ограниченной контуром C, $Rez(f, z_k)$ — вычет функции f в особой точке z_k.	Вычислить интеграл $\oint_{\|z-1\|=2} \frac{\sin z}{z} dz$

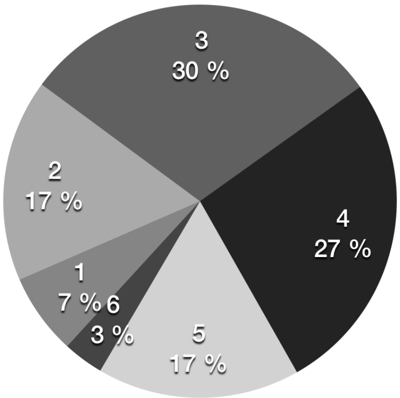
Каждый этап тестирования проводился после изложения соответствующего теоретического материала на лекционных занятиях. Работа проводилась в дисплейном классе за компьютерами, на которых была запущена тренировочная версия системы. Процесс контролировался двумя преподавателями. Каждый студент работал с окном, в котором выдавались задачи соответствующего типа. Получив условие, испытуемые приступали к решению, закончив которое вводили ответ в соответствующее поле. По результатам проверки ответа студент получал сообщение об успехе или неудаче и ход решения (в последнем случае). Кнопка запроса новой задачи приводила к генерации другой задачи того же типа. Система вела запись каждой попытки каждого студента. В рамках тестирования производился подсчет номера попытки, начиная с которой студент показывал стабильный успех — три подряд правильно решенные задачи. Результаты тестирования представлены на Рис. 5.1 в виде



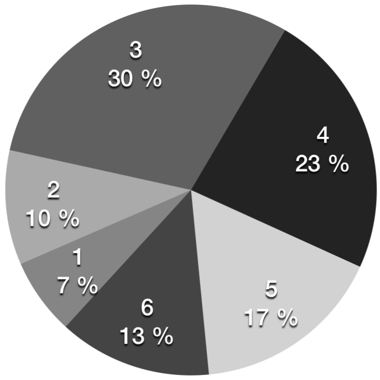
(а)



(б)



(в)



(г)

Рис. 5.1: Результаты тестирования EdLeTS на группе студентов по задачам: (а) вычисление пределов с использованием специальных пределов и теоремы о двух милиционерах; (б) вычисление производных элементарных функций; (в) интегрирование по частям; (г) комплексное интегрирование с помощью вычетов.

секторных диаграмм. Каждая диаграмма на рисунке отвечает одной из тем. Номер каждого сектора на диаграмме отвечает числу попыток до достижения стабильного успеха. Площадь сектора отражает процент студентов, которым понадобилось соответствующее число попыток для достижения стабильного успеха.

Наиболее удивительным и интересным результатам этого исследования оказалось то, что во всех случаях, не более шести попыток потребовалось испытуемым, чтобы добиться понимания схемы решения соответствующих задач. Причем значительная часть студентов справлялась с заданиями быстрее. Так, около 70% учеников суммарно добились успеха не более чем за 4 попытки с задачами на применение теоремы о вычетах (Рис. 5.1(г)). Несмотря на то, что указанные задачи являются весьма алгоритмическими и достаточно простыми, результаты превзошли ожидание. Они наглядно демонстрируют состоятельность идеи обучения вычислительным навыкам на множестве случайно генерируемых примеров: начиная с какого-то момента, студент понимает, в чем схожи все предлагаемые ему задачи и в чем их различия, понимает, как устроено решение таких задач и как детали влияют на это решение. Следует, конечно, отметить, что более сложные задачи обнаруживают бóльшие требования к знаниям и умениям студентов, а потому процесс усвоения процедуры решения таких задач может требовать большего времени и большего числа попыток. Но общая идея сохраняется.

Опрос студентов

С целью выявления оценки пользователями-студентами различных аспектов работы с системой EdLeTS был проведен эксперимент на группе из 17 студентов первого года обучения, проходящих курс математического анализа. Студенты использовали систему в специально отведенное время на практических занятиях в течение трех недель для тренировки изучаемых приемов. Результаты тренировок никак не протоколировались и не влияли на формальные показатели успеваемости обучающихся. По окончании эксперимента студентам было предложено пройти опрос, призванный выяснить, насколько удобно им было пользоваться системой и каков общий уровень одобрения подобной инициативы среди них. Опрос состоял из следующих вопросов:

1. Насколько полезными были подсказки, разъясняющие ход решения, появляющиеся в случае неправильного ответа?

    1. совершенно бесполезны
    2. почти бесполезны
    3. иногда полезны
    4. в целом полезны
    5. очень полезны

2. Какой подход в обучении Вы бы предпочли: традиционный (традиционные задачники, домашние задания, обычные контрольные и т.д.) или с использованием EdLeTS?

    1. Только традиционный и никак иначе
    2. Традиционный подход лично мне ближе
    3. Не могу ответить
    4. Подход с использованием EdLeTS лично мне ближе
    5. Определенно, подход EdLeTS лучше

3. Насколько удобно/просто Вам было пользоваться системой EdLeTS (оцените интерфейс, процесс получения задачи, проверку ответа, разбор решения)?

    1. очень сложно/неудобно
    2. не очень удобно
    3. могло бы быть и лучше
    4. достаточно удобно
    5. очень удобно

4. Порекомендовали бы Вы EdLeTS для использования на других занятиях?

    1. точно нет
    2. вряд ли
    3. не могу ответить
    4. скорее всего, да

5. точно да

Результаты опроса представлены на Рис. 5.2. Предложенные варианты ответов на вопросы анкеты упорядочены по возрастанию положительности отклика. Таким образом, студентам было предложено оценить различные аспекты системы по шкале от 1 до 5: 1 отвечает достаточно низкой оценке соответствующего аспекта, 5 — высокой оценке. Опрос проводился анонимно. На Рис. 5.2 каждая группа из двух столбцов диаграммы соответствует ответам одного студента на два вопроса. В обеих диаграммах анкеты упорядочены одинаково, таким образом, ответы на все четыре вопроса анкеты одного студента находятся в одной колонке (например, результаты условно первой анкеты представлены первой парой столбцов диаграммы 5.2(а) и первой парой столбцов диаграммы 5.2(б)). Результаты опроса, представленные на Рис. 5.2(а) показывают, что, в большинстве своем, студенты настроены положительно в отношении возможности использовать систему EdLeTS в обучении. Большинство респондентов находит достаточно полезной возможность просмотреть конкретизированный ход решения в случае, если возникла проблема с решением задачи. Все опрошенные указывают на то, что находят предложенный подход более привлекательным, по сравнению с традиционным. Вопросы 1 и 2 призваны выявить оценку самого подхода и способа его подачи в системе EdLeTS. Судя по результатам, предлагаемый подход и его реализация отвечают на некоторые запросы студентов, делая образовательный процесс для них более удобным, привлекательным. Вопрос 3 касается непосредственно реализации системы — удобства интерфейса, его очевидности, наличия или отсутствия каких-то затруднений при работе. Ответы на этот вопрос также оказались весьма положительными. У испытуемых не возникло особых сложностей в процессе взаимодействия с приложением, хотя, стоит отметить, использованная в рамках эксперимента функциональность была весьма ограничена. И все же, для современного студента, выросшего в окружении великого множества веб-приложений (таких как социальные сети, разнообразные текстовые и графические редакторы, обучающие приложения и многие другие), интерфейс EdLeTS в целом является достаточно естественным, так как наследует множество типичных для современных приложений приемов. Последний, четвертый, вопрос нацелен на выявление общего уровня одобрения данной инициативы. Ответы студентов показывают, что в среднем они настроены положительно.

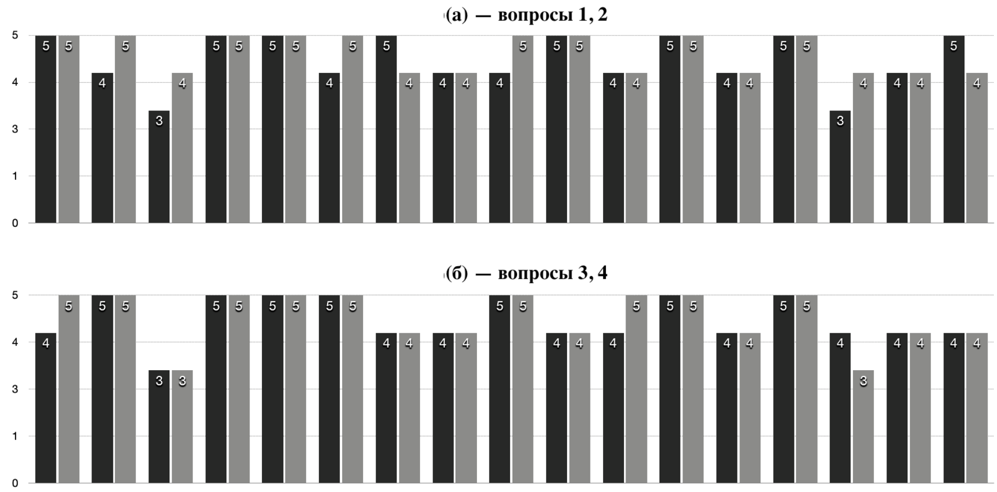


Рис. 5.2: Результаты опроса студентов. Каждая пара столбцов соответствует ответам одного студента на два вопроса: (а) — вопросы 1 и 2, (б) — вопросы 3 и 4.

Тестирование алгоритма поточечной проверки

Отдельное исследование было посвящено тестированию алгоритма дополнительной поточечной проверки, представленного в главе 2. Напомним, что данная процедура является стохастической и результат может с некоторой вероятностью оказаться неверным. В разделе 2.2 главы 2 было показано, что даже при неправдоподобно больших значениях оценки числа нулей исследуемой функции

Процент запусков	**Количество шагов**
$\sim 73\%$	1 шаг
$\sim 12\%$	2—10 шагов
$\sim 8\%$	11—30 шагов

Таблица 5.2: Распределение числа шагов алгоритма дополнительной поточечной проверки на тестовых запусках

в заданном отрезке, с помощью относительно небольшого количества проверок можно добиться исчезающе малой вероятности ошибки предложенного алгоритма. Тем не менее, возникает вопрос, насколько часто все же этот алгоритм может ошибаться. Другим немаловажным вопрос является вопрос о том, сколько в среднем требуется проверок (и, соответственно, как долго в среднем работает алгоритм). Очевидно, что, в силу случайности выбора точек для проверки, количество проверяемых точек варьируется от запуска к запуску даже для одинаковых исходных данных.

Был проведен ряд тестов на различных примерах. В каждом тесте строилась задача со статичным ответом (не изменяемым от генерации к генерации), заданным некоторой функцией. Для каждой такой задачи проводилось множество тестов, в каждом из которых для проверки предлагалась функция, так или иначе близкая к настоящему ответу, но с ним не совпадающая. В случае, если запускался алгоритм дополнительной поточечной проверки, записывался результат проверки (корректность) и количество точек проверки, задействованных алгоритмом. В рамках исследования ни один тест не дал неправильного результата. При этом в большинстве случаев проверка была завершена на первом же шаге. Более 85% проверок завершились менее, чем за 10 шагов. Результаты тестирования показаны в таблице 5.2. Результаты показывают, что на практике предложенный алгоритм способен выдавать корректный результат достаточно быстро.

5.3 Выводы

В данной главе были подняты вопросы о внедрении системы EdLeTS в процесс обучения в рамках практических занятий по математическим дисциплинам и приведены результаты тестирования возможностей системы и самого разработанного подхода. В процессе интеграции системы в учебный процесс важно четко и правильно указать статус этого приложения. Ни один программный продукт, в том числе и EdLeTS, не может и не должен заменить преподавателя. Цель разработанной в рамках данного диссертационного исследования продукта — в первую очередь, облегчить жизнь преподавателям и студентам, избавляя их от недостатков традиционного подхода к организации практических занятий, и повысить эффективность образовательного процесса. Новые возможности, предлагаемые системой, призваны повысить точность оценок, привнести новый уровень персонификации. При этом, однако, нисколько не должна уменьшаться роль преподавателя, за которым всегда остается самая важная часть обучения.

Разделы данной главы, посвященные тестированию основной модели системы и анкетированию студентов, близко следуют опубликованной ранее работе [26]. Полученные результаты показывают, что предлагаемый подход достаточно успешно работает на практике и позволяет в достаточно короткие сроки получить осязаемый результат (в прогрессе студентов). Опрос студентов показал, что внедрение EdLeTS в образовательный процесс в целом воспринимается среди студентов положительно. При этом важно упомянуть, что анкетирование проводилось в группе, тестировавшей функциональность системы без каких-либо последствий с точки зрения формальной оценки их деятельности. Таким образом была получена «чистая» оценка системы, однако, важно учитывать возможность некоторого изменения отклика при «боевом» использовании системы, когда успехи и неудачи там тем или иным образом влияют на формальные аспекты занятий. При дальнейшем внедрении продукта в практические занятия по различным дисциплинам необходимо проведение более развернутых тестов с анкетированием, отражающим различные стороны взаимодействия пользователей с системой.

Заключение

Обзор существующих систем поддержки образовательного процесса и подходов к усилению образовательного аппарата посредством внедрения ИКТ, приведенный во **введении** к данной работе, показывает, что на сегодняшний день остается открытым вопрос об эффективном пополнении инструментов, которыми располагают преподаватели и студенты в рамках занятий в стенах образовательных учреждений и самостоятельного обучения. Концепция *умного образования*, сформировавшаяся не так давно и все еще находящаяся в стадии поиска строгих определений, указывает на новейшие требования, предъявляемые к образовательным продуктам, и на возможность их удовлетворить при текущем уровне технологического развития, ставя в основу персонификацию, адаптивность и общее повышение эффективности образования. История развития образовательного ПО позволяет проследить ошибки и проблемы первых таких систем и избежать их теперь. Обзор литературы по данной теме показывает, что одной из основных проблем на заре внедрения ИКТ в образование был фокус на технологиях, отсутствие ясного понимания места и роли программных продуктов в обучении. Теперь же становится понятно, что образовательное ПО должно быть в первую очередь посвящено организации наиболее удобного способа взаимодействия всех участников образовательного процесса друг с другом и с объектами изучения. В этом смысле математические дисциплины стоят на особом месте. Как уже неоднократно упоминалось в тексте работы, на ранних стадиях обучения таким дисциплинам чрезвычайно важную роль играют вычислительные тренировочные задачи в своей естественной форме. И эта форма не терпит упрощений или адаптаций под какую-то технологию. Напротив, необходимо предоставить техническое решение, способное обеспечить возможность работы с такими задачами в их естественном виде.

Настоящая работа базируется на общей идее порождения конкретных задач по некоторым общим описаниям, сформулированной В.Г. Даниловым, а также — на некоторых попытках реализации этой идеи, предпринятых ранее [46, 47]. В рамках данного исследования была предложена удобная и жизнеспособная модель шаблонов тренировочных задач, формализующая эту идею. Сама идея шаблонизации является весьма естественной для тренировочных задач. Она подсказана самой структурой таких специализированных примеров, а любой специалист, когда-либо занимавшийся самостоятельной разработкой задач для целей обучения, осознанно или подсознательно прибегал к этой идее. Необходимость автоматизации процессов генерации конкретных тренировочных заданий и их обработки обусловлена в первую очередь тем обстоятельством, что они отнимают значительное время, не принося при этом практической пользы ни преподавателям, ни студентам. Временны́е рамки, в которых вынуждены существовать учителя и обучающиеся вынуждают устанавливать компромисс между оттачиванием достаточно механических, но, в то же время, необходимых, навыков и изучением более тонких, интересных, творческих элементов. В настоящее время даже на математических факультетах в большинстве ВУЗов не удается уделить достаточно времени таким темам, как доказательство теорем, вывод свойств, исследование математических объектов и связей между ними. Практические занятия на первом курсе математического анализа оказываются практически полностью посвященными выработке и оттачиванию вычислительных навыков: время занятий тратится на разбор бесконечных примеров и проблем, возникающих у обучающихся при решении домашних заданий в силу недостатка информации или сложности ее поиска. Значительное время у преподавателей занимает подготовка различного рода заданий и их проверка. Подход, описанный в данной работе, имеет своей целью избавить преподавателей и студентов от этих рутинных составляющих процесса обучения, оставив их выполнение соответствующим программным решениям. Разработанная в рамках настоящего диссертационного исследования система EdLeTS представляет собой комплекс инструментов и средств, позволяющих преподавателям и студентам сконцентрироваться на действительно важных аспектах образовательного процесса, минимизируя затраты на механические составляющие.

Основным отличием разработанной системы EdLeTS от прочих образовательных продуктов является фокусировка на понятии тренировочных задач в математических дисциплинах (как и в других дисциплинах, в области точных наук) и понимание роли тренировочных задач в процессе обучения. Как уже указывалось выше, для эффективности образовательного процесса чрезвычайно важно, чтобы при решении задачи обучающийся прошел весь путь целиком — от постановки задачи до окончательного ответа. При этом он проходит такие важные этапы, как интерпретация условия задачи, построение стратегии (или нескольких стратегий) решения, проведение всех необходимых вычислительных манипуляций, получение окончательного ответа. При этом ошибка на любом из этих этапов является существенной. Важно, что задача не может считаться полностью правильно решенной, если на каком-то этапе допущена ошибка — пусть и незначительная. Основной подход в EdLeTS предписывает предоставлять студенту задачи до тех пор, пока он не научится **все** делать правильно. Кому-то для достижения успеха понадобится совсем немного задач, другому может понадобиться значительно больше. В любом случае, каждый должен иметь возможность отточить тот или иной навык до надлежащего уровня. Опыт многих преподавателей показывает, что даже, казалось бы, хороший уровень усвоения теоретического материала не всегда приводит к столь же хорошим показателям на практике. Практический навык нарабатывается на конкретных задачах и укрепляет теоретические знания. Лишь посредством тренировки на множестве специально подобранных примеров можно добиться появления устойчивого практического навыка. При этом у обучающегося появляются специфические способности распознавать типичные конструкции в сложных задачах, дробить, упрощать, производить ветвление решения и многие другие.

Предложенная и реализованная в EdLeTS модель тренировочных задач допускает генерацию множества конкретных задач с помощью определения конкретных значений динамических параметров. Количество порождаемых таким образом конкретных задач зависит от мощностей множеств значений динамических параметров данной задачи и от количества этих параметров. Для некоторой задачи можно оценить это количество через c^n, где c — мощность наименьшего из множеств значений динамических параметров задачи, а n — количество динамических параметров. Таким образом, в реальных примерах количество порождаемых задач оказывается достаточно велико. При этом, на самом деле, не так уж важно, чтобы каждый задачи, решаемые студентами в рамках домашних или контрольных работ никак не пересекались. В случае самостоятельной работы, каждый обучающийся с чрезвычайно высокой вероятностью прорешает свой уникальный **набор** задач. В подходе, реализуемым в EdLeTS, попросту пропадает проблема списывания или подмены работ. С другой стороны, остается проблема недобросовестности при решении задач в домашних условия. Так, потенциальный студент-злоумышленник может тем или иным способом убедить другого человека решить задачи за него, через его аккаунт (например, чтобы набрать необходимый установленный уровень по количеству правильно решенных задач, отнесенных к общему числу попыток). Также в ряде задач особо осведомленные студенты могут воспользоваться средствами символьных вычислений (например, если задание касается вычисления производной, то с этим легко справится какая-либо СКА). Такие проблемы есть, и их решение не входит в задачи настоящего исследования. EdLeTS не ставит своей целью предложить форму строгого контроля и надзора за действиями студентов. Можно сказать, что в основе лежит предположение о том, что преподаватель и студент вместе работают для достижения одной и той же цели — в противном случае весь образовательный процесс становится бессмысленным. Таким образом, предусмотренные в рамках EdLeTS разграничения доступа и всевозможные методы защиты реализованы как просто соответствующие здравому смыслу и естественные (например, сокрытие посчитанного ответа до получения ответа от пользователя). Отметим, что, в случае необходимости проведения дополнительного контроля, возможно применение дополнительных мер. К примеру, использование функциональности контрольных работ EdLeTS на практике возможно при проведении работы в дисплейном классе под контролем преподавателя. В зависимости от желаемого уровня строгости контроля можно применять различные меры ограничения неподобающих действий.

Разработанные средства составления описаний тренировочных задач позволяют пользователям системы создавать шаблоны задач, не прибегая к каким-либо специфическим конструкциям, и не требует от них дополнительной квалификации. Представленный дизайнер задач позволяет специалистам (преподавателям, менторам, ассистентам) создавать описания практически так же, как они делали бы это вручную, с помощью бумажного листа и ручки. При этом широкие возможности рандомизации по различным множествам значений позволяют избежать излишних сложностей и ухищрений при определении динамических параметров. Данный набор неминуемо будет пополняться дополнительными функциями при выявлении новых требований по мере расширения области применения системы. В случае необходимости пользователи имеют возможность прибегнуть непосредственно к языку описания динамических параметров задач — SmallTask. Данный язык разработан с учетом специфических требований области своего применения при поддержании максимальной простоты синтаксиса. Его использование предъявляет минимальные требования к знаниям пользователей или обращению к дополнительной документации.

Вся деятельность пользователей-студентов, связанная с решением тренировочных задач, отслеживается и записывается с целью обеспечения возможности прослеживания пути каждого студента и предложению различной помощи. Возможность установления отношения предшествования на множестве тренировочных задач позволяет при имеющемся портрете студента в автоматическом режиме генерировать подсказки — предположения о возможных способах устранения пробелов в знаниях и предложения по дальнейшему развитию. Связь таких отношений с объектами теории образовательных пространств позволяет эксплуатировать результаты этой теории. В частности, это обстоятельство позволяет использовать в рамках системы EdLeTS марковскую процедуру оценивания, позволяющую с хорошей точностью определить текущее состояние знаний студента в заданной области. Эта возможность находит применения в задачах промежуточного контроля, определения наличия достаточного набора компетенций при поступлении на какой-либо курс или в команду образовательного проекта, в задачах определения необходимых дополнительных курсов и других. Благодаря наличию подробной информации по прогрессу каждого студента в отдельности, возможно построить достаточно подробный портрет каждого обучающегося, что позволяет преподавателю надлежащим образом модернизировать программу, оказать своевременную помощь, выявить общие, средние и персональные показатели студентов. Все эти средства предоставляют совершенно новый, по сравнению с традиционным подходом, уровень персонификации, что признано важным направлением развития образовательных систем.

Проведенные исследования аспектов внедрения разработанной системы EdLeTS в образовательный процесс позволяют рассчитывать на успех при построении курса с использованием данного продукта. Показатели, выявленные при тестировании основной модели, демонстрируют состоятельность предложенной идеи: решая раз за разом все новые примеры на одну и ту же тему, даже наименее успешные испытуемые относительно быстро добиваются успеха. Сам по себе этот результат не является удивительным. Методика обучения механическим вычислительным навыкам с помощью большого числа примеров — давно известная, выдержавшая проверку временем практика. Система EdLeTS позволяет вернуть этой методике надлежащий размах и объемы, автоматизируя сопровождающие процессы и оптимизируя время, затрачиваемое на весь процесс тренировки в целом. Опрос студентов показал положительный отклики среди обучающихся. Привычный способ взаимодействия с системой только укрепляет их одобрение, а очевидные преимущества (по сравнению с традиционной формой), которые сулит использование EdLeTS в образовательном процессе повышают также и мотивированность. Внедрение системы в существующие образовательные программы происходит при тесном сотрудничестве с преподавателем. Личные беседы показывают, что среди преподавателей наибольший энтузиазм по отношению к подобной инициативе проявляют молодые преподаватели, тогда как более опытные старшие преподаватели, успевшие выработать собственный, уже доказавший определенную эффективность подход, нередко проявляют скептическое отношение. Однако положительных и заинтересованных откликов оказывается достаточно много. Учитывая тот факт, что разработка и внедрение образовательного ПО на сегодняшний день обретает новую популярность, есть все основания считать, что спрос на разработанный продукт окажется достаточно высоким.

Список сокращений и условных обозначений

АСД - Абстрактное синтаксическое дерево
БД - База данных
ИКТ - Информационно-коммуникационные технологии
УК - Учебная комната, способ организации пользователей в системе EdLeTS
ПО - Программное обеспечение
РБНФ - Расширенная форма Бэкуса-Наура
с.в. - случайная величина
СКА - Система компьютерной алгебры
СКМ - Система компьютерной математики
DOM - англ. Document Object Model, объектная модель документа
LMS - англ. Learning Mamangement System, система управления обучением
MMVP - Шаблон проектирования Model-View-ViewModel
MTV - Шаблон проектирования Model-Template-View
MVC - Шаблон проектирования Model-View-Controller
ORM - Object-relational mapping, объектно-реляционное отображение

Литература

[1] *Carson S., Schmidt J.P.* The massive open online professor // *Academic Matters*. — 2012. — Pp. 20–23.

[2] Analysis of the effectiveness of online learning in a graduate engineering math course / C.L. Karr, B. Weck, D.W. Sunal, T.M. Cook // *The Journal of Interactive Online Learning*. — 2003. — Vol. 1, no. 3. — Pp. 1–8.

[3] *Abdelraheem A.Y.* Computerized Learning Environments: Problems, Design Challenges and Future Promises // *The Journal of Interactive Online Learning*. — 2003. — Vol. 2, no. 2. — Pp. 1–9.

[4] *Richardson D.* Some undecidable problems involving elementary functions of a real variable // *The Journal of Symbolic Logic*. — 1969. — Vol. 33, no. 4. — Pp. 514–520.

[5] *Khan B.H.* Developing eLearning strategy in Universities of Bangladesh // ICT. — 2014. — P. 78.

[6] *Tikhomirov V.* The Moscow State University of Economics, Statistics and Informatics (MESI) on the way to smart education // Proceedings of the 10th International Conference on Intellectual Capital, Knowledge Management and Organisational Learning: ICICKM. — 2013.

[7] *Zhu Z.-T., Yu M.-H., Riezebos P.* A research framework of smart education // *Smart learning environments*. — 2016. — Vol. 3, no. 1. — P. 4.

[8] Smarter universities: A vision for the fast changing digital era / M. Coccoli, A. Guercio, P. Maresca, L. Stanganelli // *Journal of Visual Languages & Computing.* — 2014. — Vol. 25, no. 6. — Pp. 1003–1011.

[9] Smart university taxonomy: features, components, systems / V.L. Uskov, J.P. Bakken, A. Pandey et al. // Smart Education and e-Learning 2016. — Springer, 2016. — Pp. 3–14.

[10] *Hwang G.-J.* Definition, framework and research issues of smart learning environments-a context-aware ubiquitous learning perspective // *Smart Learning Environments.* — 2014. — Vol. 1, no. 1. — P. 4.

[11] *Koper R.* Conditions for effective smart learning environments // *Smart Learning Environments.* — 2014. — Vol. 1, no. 1. — P. 5.

[12] *Spector J.M.* Conceptualizing the emerging field of smart learning environments // *Smart learning environments.* — 2014. — Vol. 1, no. 1. — P. 2.

[13] *Kim T., Cho J.Y., Lee B.G.* Evolution to smart learning in public education: a case study of Korean public education // Open and Social Technologies for Networked Learning. — Springer, 2013. — Pp. 170–178.

[14] *Jonassen D.H.* Computers as mindtools for schools: Engaging critical thinking. — Prentice Hall, 2000.

[15] *Rubin A.* Technology Meets Math Education: Envisioning a Practical Future Forum on the Future of Technology in Education // *Resources in Education.* — 1999.

[16] *Таранчук В.Б.* Основные функции систем компьютерной алгебры: пособие для студентов факультета прикладной математики и информатики. — 2013.

[17] *Akritas A.G.* Elements of computer algebra with applications. — Wiley New York, 1989. — Vol. 3.

[18] *Дьяконов В.П.* Энциклопедия компьютерной алгебры. Учебное пособие. — ДМК Пресс, 2009.

[19] *Lavicza Z.* A comparative analysis of academic mathematicians' conceptions and professional use of computer algebra systems in university mathematics // *Unpublished Doctoral Dissertation. Faculty of Education, University of Cambridge, Cambridge, UK.* — 2008.

[20] *Drijvers P.* Students encountering obstacles using a CAS // *International Journal of Computers for Mathematical Learning.* — 2000. — Vol. 5, no. 3. — Pp. 189–209.

[21] *Lavicza Z.* Factors influencing the integration of Computer Algebra Systems into university-level mathematics education // *International Journal for Technology in Mathematics Education.* — 2007. — Vol. 14, no. 3. — P. 121.

[22] A Method to Describe Student Learning Status for Personalized Computer Programming e-Learning Environment / Y. Yan, K. Hara, H. Nakano et al. // Advanced Information Networking and Applications (AINA), 2016 IEEE 30th International Conference on / IEEE. — 2016. — Pp. 231–238.

[23] The assessment of knowledge, in theory and in practice / J.-C. Falmagne, E. Cosyn, J.-P. Doignon, N. Thiéry // Formal concept analysis. — Springer, 2006. — Pp. 61–79.

[24] *Falmagne J.-C., Doignon J.-P.* Learning spaces: Interdisciplinary applied mathematics. — Springer Science & Business Media, 2010.

[25] *Danilov V.G., Turuntaev I.S.* Interactive educational system based on generative approach, and the problem of answer checking //

Smart Education and e-Learning 2016. — Springer, 2016. — Pp. 527–537.

[26] *Turuntaev I.S.* EdLeTS: Towards Smartness in Math Education // Smart Universities. SEEL 2017 / Ed. by Howlett R. Jain L. Uskov V., Bakken J. — Springer, Cham, 2017. — Vol. 70 of *Smart Innovation, Systems and Technologies*.

[27] Personalised learning: Ambiguities in theory and practice / R.J. Campbell, W. Robinson, J. Neelands et al. // *British Journal of Educational Studies*. — 2007. — Vol. 55, no. 2. — Pp. 135–154.

[28] *Ignatova N., Dagiene V., Kubilinskiene S.* ICT-based learning personalization affordance in the context of implementation of constructionist learning activities // *Informatics in Education*. — 2015. — Vol. 14, no. 1. — P. 51.

[29] *Сканави М., Зайцев В., Рыжков В.* Элементарная математика 2-е изд., перераб. и доп. — М.: Наука, 1974.

[30] *Демидович Б.П.* Сборник задач и упражнений по математическому анализу. — АСТ, 2006.

[31] *Knuth D.E.* The TeX-book (revised). — 1990.

[32] *Хопкрофт Д., Мотвани Р., Ульман Д.* Введение в теорию автоматов, языков и вычислений. — Вильямс М, 2002.

[33] *Parr T.J., Quong R.W.* ANTLR: A predicated-LL (k) parser generator // *Software: Practice and Experience*. — 1995. — Vol. 25, no. 7. — Pp. 789–810.

[34] *Новиков П.С.* Об алгоритмической неразрешимости проблемы тождества слов в теории групп // *Труды Математического института имени ВА Стеклова*. — 1955. — Vol. 44, no. 0. — Pp. 3–143.

[35] *Richardson D., Fitch J.* The identity problem for elementary functions and constants // Proceedings of the international symposium on Symbolic and algebraic computation / ACM. — 1994. — Pp. 285–290.

[36] *Laczkovich M.* The removal of π from some undecidable problems involving elementary functions // *Proceedings of the American Mathematical Society.* — 2003. — Vol. 131, no. 7. — Pp. 2235–2240.

[37] *Committee IS et al.* 754–2008 IEEE Standard for Floating-Point Arithmetic // *IEEE Computer Society Std.* — 2008. — Vol. 2008.

[38] *Goldberg D.* What every computer scientist should know about floating-point arithmetic // *ACM Computing Surveys (CSUR).* — 1991. — Vol. 23, no. 1. — Pp. 5–48.

[39] *Hellmann D.* The Python standard library by example. — Addison-Wesley Professional, 2011.

[40] *Falmagne J.-C., Doignon J.-P.* Knowledge Structures and Learning Spaces // Learning Spaces. — Springer, 2011. — Pp. 23–41.

[41] *Cosyn E., Uzun H.B.* Note on two necessary and sufficient axioms for a well-graded knowledge space // *Journal of Mathematical Psychology.* — 2009. — Vol. 53, no. 1. — Pp. 40–42.

[42] *Birkhoff G. et al.* Rings of sets // *Duke Mathematical Journal.* — 1937. — Vol. 3, no. 3. — Pp. 443–454.

[43] *Falmagne J-.C., Doignon J.-P.* A class of stochastic procedures for the assessment of knowledge // *British Journal of Mathematical and Statistical Psychology.* — 1988. — Vol. 41, no. 1. — Pp. 1–23.

[44] *Deacon J.* Model-view-controller (MVC) architecture // *Online][Citado em: 10 de março de 2006.] http://www.jdl.co.uk/briefings/MVC.pdf.* — 2009.

[45] *Gossman J.* Introduction to Model/View/ViewModel pattern for building WPF apps // *MSDN Blogs*. — 2005.

[46] *Кантор И.А.* Интернет-система генерации задач на основе порождающих грамматик // *Сборник науных трудов МИЭМ*. — 2007.

[47] *Данилов В.Г., Петров П.П.* Пополнение набора задач в электронных обучающих системах с использованием визуального формульного редактора // *Качество. Инновации. Образование*. — 2009. — no. 7. — Pp. 57–65.