

Система обработки формального языка ForTheL. Её использование в математике и обработке требований к программному обеспечению.

Процик Петр Павлович

Киевский Национальный Университет имени Тараса Шевченко, Киев 03127,
ул. Сеченова 6 общ. 16 ком. 132, tel. 80668731450.

1. Введение: постановка проблемы

Существуют различные подходы к формализации и хранению знаний. Наибольшим недостатком многих существующих подходов является сложность языков представления, что приводит к их ограниченному применению и сужает круг пользователей.

Язык ForTheL (Formal Theory Language) представляет собой средство формализации знаний на языке близком к естественному английскому языку технических публикаций. Это достигается путём использования специальной грамматики английского языка и возможностью непосредственного её расширения в ForTheL тексте. Мощность языка ограничена выразительной силой формул логики первого порядка. Кроме того, существует алгоритм, позволяющий эффективно транслировать высказывания языка в логические формулы и далее обрабатывать их.

Первоначально язык применялся для записи и хранения математических знаний, и кроме того автоматического доказательства теорем. Для решения этих задач на факультете кибернетики была разработана система автоматизации дедукции (САД). Она доступна через веб узел в on line режиме (<http://ea.unicyb.kiev.ua>).

Изначально, целью моей работы было дальнейшее развитие системы – разработка персональной версии для ОС Windows, разработка графической оболочки пользователя для работы в системе, поиск новых путей применения системы.

Другой не менее важной задачей было использование языка ForTheL для формализации различных текстов, чтобы показать возможность его реального практического применения.

В процессе исследовательских работ посвященных языку ForTheL была поставлена задача о возможности использования языка для хранения и обработки требований к программному обеспечению. Такой обработкой может быть, например, проверка на непротиворечивость, полноту требований, доказательство различных свойств системы. Преимущество такого подхода над существующими подходами в том, что он не только позволяет хранить и обрабатывать *формальные требования* к программному обеспечению в удобном формате, но и допускает использовать его широким кругом пользователей, что, несомненно, является важным качеством для систем такого рода.

Таким образом, работа состоит из таких основных частей – разработка персональной версии и других средств поддержки системы САД, использование языка ForTheL для записи математических текстов, обработки их в системе и разработка многопользовательской ориентированной на веб системы поддержки процесса хранения и обработки требований к программному обеспечению.

Первые две задачи решались в моей бакалаврской работе в прошлом году, исследования и разработка системы обработки требований к программному обеспечению проводятся в дипломной работе в этом учебном году.

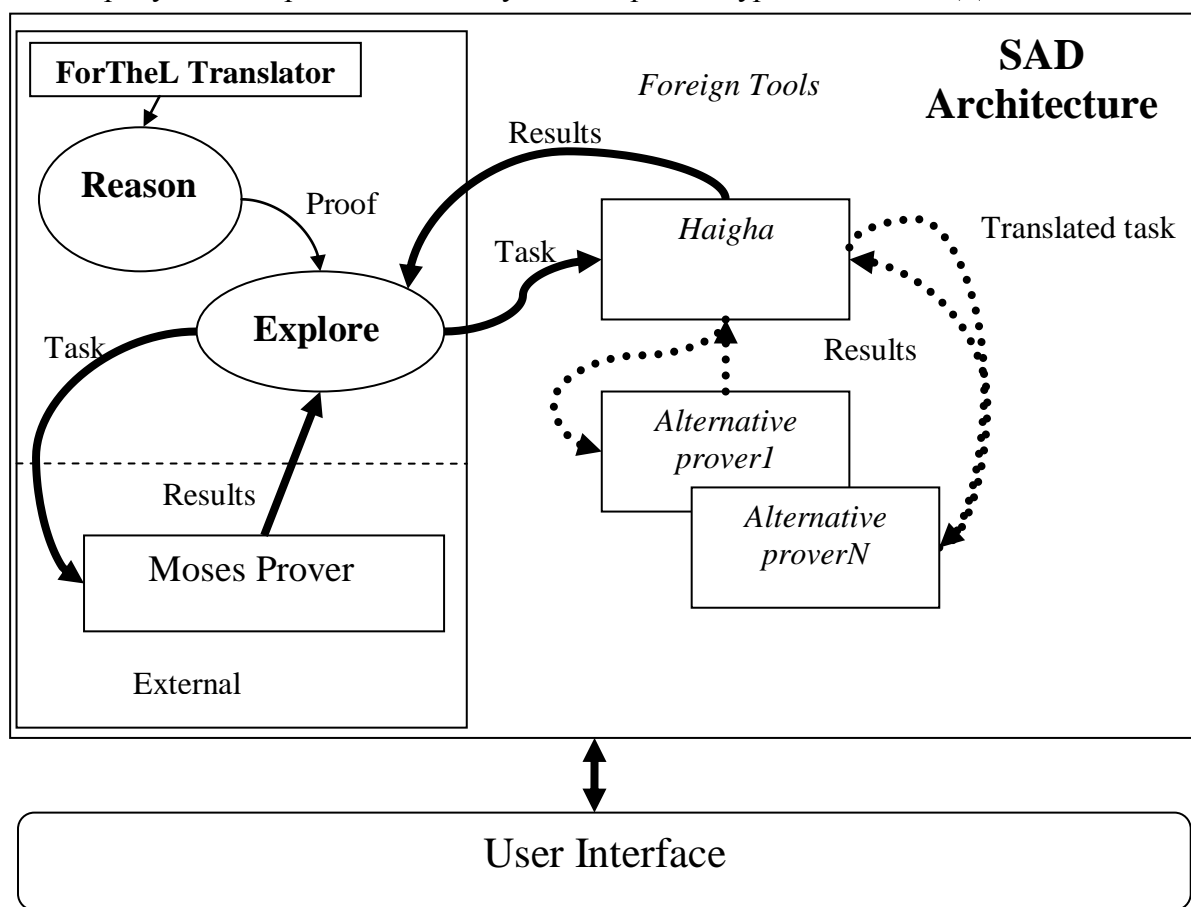
Следует заметить, что обе работы тесно связаны, как с точки зрения реализации, так и теоретических исследований. Поскольку текущая работа непосредственно опирается на результаты и программные разработки, полученные в прошлом году.

2. Изложение решения проблемы

Исходная версия системы САД была разработана для ОС Linux и была доступна пользователям через Интернет. Такой подход существенно ограничивал круг пользователей системы. Поэтому было принято решение о разработке персональной

системы для ОС Windows. Она должна быть полностью совместима с предыдущей версией и сохранить её внутреннюю архитектуру.

На рисунке изображена концептуальная архитектура системы САД:



Из схемы видно что, она состоит из нескольких модулей – ForTheL translator, Reason, Explorer, Haigha, внешних и внутреннего пруверов (программ доказывающих утверждения), пользовательского интерфейса.

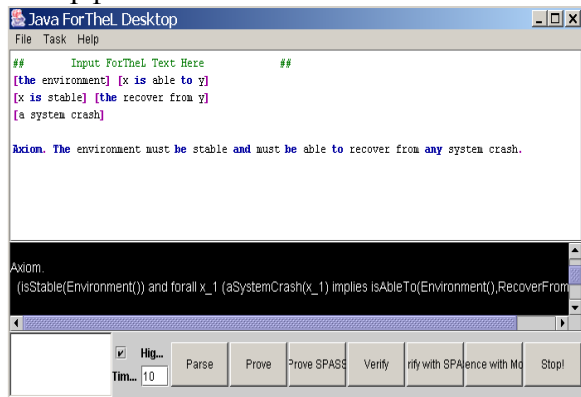
Поскольку первоначальная система разрабатывалась для платформы Linux, прямой перенос существующих модулей под Windows был невозможен из-за особенностей технической реализации, которые характерны для этой платформы. Эти особенности касаются аппарата ввода/вывода, потоков и процессов, а также способа работы с внешними модулями (программами). Для работы с этими объектами, операционные системы на ядре Linux используют стандарт Posix. Он описывает низко уровневые функции работы с этими классами объектов. Платформой MS Windows стандарт Posix полностью не поддерживается, вместо него используются объекты ядра: процессы (process), потоки (thread) и каналы ввода/вывода (pipe).

Ядро системы (транслятор, постановщик задач, основная логика) написано на языке функционального программирования Haskell, остальные модули разработаны на языке C. При переработке системы для Windows в ядре необходимо было переписать все системно зависимые процедуры, на эквивалентные с использованием аппарата объектов ядра. Модули, написанные на языке C (Haigha, Moses), подверглись значительной переработке.

Кроме того, жирными стрелочками на схеме изображены обмены данными между модулями, для которых возникает задача синхронизации (согласования). С теоретической точки зрения она сводится задаче „производитель - потребитель”. Напомним ее формулирование: производитель вырабатывает объекты и записывает их в буфер обмена. Когда буфер, заполненный он не должен больше производить. Потребитель не может потреблять из пустого буфера и на обработку одного объекта необходим определенный

промежуток времени. Решение такой задачи основано на использовании техники семафоров. В нашем случае взаимодействие более сложное. Поскольку каждый модуль является как производителем, так и потребителем. Эта задача также была успешно решена в рамках работ по разработке персональной версии системы.

Для удобного взаимодействия с системой был разработан пользовательский интерфейс к системе на языке Java. Он предоставляет доступ ко всем существующим



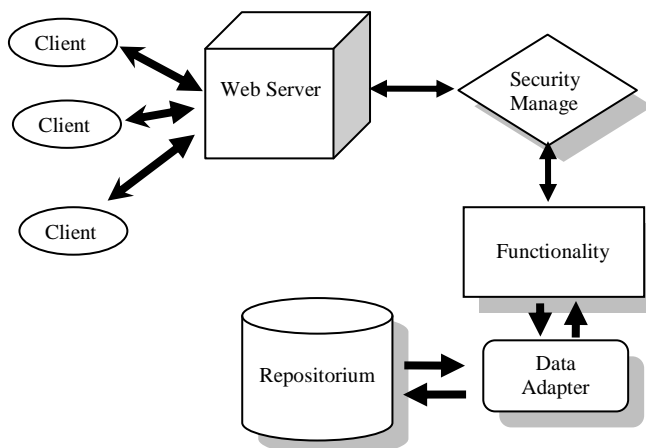
функциям системы. В нём реализована раскраска синтаксиса для языка ForTheL, что также способствует более удобной работе.

Для тестирования работоспособности системы использовались примеры исходной версии, а также мною был выполнен перевод части главы «Формальная арифметика» из книги Мендельсона [1] на язык ForTheL.

Кроме этого, было предложено использовать язык ForTheL для обработки требований к

программному обеспечению. Прежде всего, необходимо было определить классы требований, которые можно эффективно обрабатывать. В работе рассматриваются требования, которые представляют ограничения к системе. Например, требование «у самолёта должно быть два крыла», можно записать такой формулой: **plane(X) => CardinalityOf(wing(X)) = 2**. А на языке ForTheL: «the cardinality of wings of any plane is two». Далее такие требования можно проверять на непротиворечивость.

Система управления требованиями к программному обеспечению - многопользовательская система, содержащая механизм управление ролями пользователей, обеспечивающая управление доступом к документам. Для её реализации были использованы технологии J2EE, а именно – JSP (Java Server Pages), Java Servlets, Enterprise Java Beans. Веб сервер и контейнер Java сервлетов Tomcat.



На схеме изображена архитектура системы управления требованиями.

Она состоит из веб сервера, с которым пользователи работают при помощи браузера. Данные хранятся в репозитории. Получение данных из репозитория обеспечивает адаптер данных. Далее они подвергаются последующей необходимой обработке. Управлением прав пользователей занимается менеджер безопасности.

Требования в репозитории хранятся в структурированном виде. Они разбиты на классы. Каждый класс содержит атрибуты (поля) различного типа (текстовые, числовые, даты и времени, ForTheL тексты или документы/файлы любого типа). Требования, написанные на языке ForTheL, кроме того, обрабатываются на сервере системой САД и пользователи системы при наличии соответствующих прав могут видеть результаты этой обработки.

3. Выводы: преимущества и недостатки

Работа представляет собой дальнейшее развитие системы автоматизации дедукции (САД) и носит как прикладной, так и исследовательский характер.

Преимуществом реализованного подхода в первую очередь является используемый в системе язык ForTheL. Кроме того, модульность архитектуры системы САД позволяет легко модифицировать и расширять функциональность системы.

Система обработки требований позволяет не только хранить хорошо структурированные требования, но и выполнять дополнительную обработку. Она ориентирована на применение при разработке критических к ошибкам систем.

Всё программное обеспечение реализовано с использованием открытых технологий и современных методов.

К сожалению, предложенная методология пока не используется при разработке реальных коммерческих проектов. Это вызвано, прежде всего, новизной подхода и недостаточной его поддержкой программными средствами. Поэтому одним из направлений этой работы было решение указанных проблем. Исследования и разработки показали потенциальную возможность применения подхода, что является важным шагом в развитии системы автоматизации дедукции.

4. Список литературы

1. Мендельсон Э. Введение в математическую логику: Пер. с англ. / Под ред. С.И. Адяна. – 3-е изд. – М.: Наука. Главная редакция физико-математической литературы, 1984. – 320 с.
2. A.V. Lyaletski. Evidential paradigm: the logical aspect, - Cybernetics and System Analysis, Vol 39, No. 5, 2003
3. Alexander Lyaletski, Konstantin Verchinine, Anatoli Degtyarev, and Andrey Paskevich. System of Automated Deduction (SAD): Linguistic and Deductive Peculiarities. In M.A. Klopotek, S.T. Wierzchon, and M. Michaliwicz, editors, Advances in Soft Computing: Intelligent Information Systems 2002, Physica-Verlag, Springer, pages 413-422, 2002.
4. Сомервилл, Иан. Инженерия программного обеспечения, 6-е издание. : Пер. с англ. – М. : Издательский дом «Вильямс», 2002. – 624 с.: ил. ISBN 5-8459-0330-0 (rus), ISBN 0-201-39815-X (eng), Pearson Education Limited, 2001.
5. Julio Cesar Sampaio do Prado Leite, Peter A. Freeman, Requirements Validation Through Viewpoint Resolution, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 17, NO. 12, DECEMBER 1991.
6. Karl E. Wiegers, Writing Quality Requirements, Software Development magazine, May 1999