УДК 51965:519.95

ОБЕСПЕЧЕНИЕ МОНОТОННОСТИ ВЫВОДА И ВЕРИФИКАЦИЯ БАЗ ЗНАНИЙ В ИНСТРУМЕНТАЛЬНОЙ ОБОЛОЧКЕ ДЛЯ СОЗДАНИЯ ИНТЕЛЛЕКТУАЛЬНЫХ НАДСТРОЕК НАД САПР

А.В. Григорьев, О.В. Малявка
Донецкий национальный технический университет

В работе определена структура и функции новой версии системы вывода в инструментальной оболочке для создания интеллектуальных надстроек над САПР, обеспечивающей полное использование всех типов продукций, определенных в рамках подхода. Разработаны методы обеспечения монотонности вывода продукций и верификация модуля знаний, позволяющие выполнить разработку соответствующих программных средств поддержки в новой версии системы вывода.

Введение. В настоящие время имеет место явная тенденция создания гибридных САПР, имеющих как «обычную», так и «интеллектуальную» компоненту. Т.о., можно говорить об актуальности работ по созданию интеллектуальных компонент гибридных САПР. Одним из путей построения гибридных САПР есть путь создания интеллектуальных надстроек над проблемно-ориентированной САПР. В данной работе рассматривается задача создания интеллектуальной надстройки над проблемно-ориентированной системой автоматизированного проектирования (САПР) для автоматизации построения интеллектуальных систем структурного проектирования сложных объектов.

Такого рода интеллектуальная надстройка над САПР проектирования парогазовых установок «Спрут» ранее была разработана авторами [1,2]. Данная система обеспечивает накопление и классификацию поступающих знаний о различных предметных областях для САПР. Система позволяет модифицировать существующие знания и создавать методики проектирования новых элементов и образцов. Спецификой САПР СПРУТ является возможность по имеющемуся внутреннему текстовому описанию объекта, включающему трубы, арматуру, оборудование и т.д. автоматически формировать графическую модель объекта, как трехмерную, так и в форме обычного чертежа. Это позволило создать интеллектуальную надстройку, способную либо выполнить реконструкцию заданного объекта, представленного текстом, либо – синтезировать требуемую текстовую модель объекта по ряду параметров, задаваемых проектировщиком с последующим автоматическим построением графической модели. Внутренним

языком СПРУТ являлось подмножество языка АВТОЛИСП, языком надстройки являлся универсальный язык типа HDL. Такое положение требовало построение интерфейса «АВТОЛИСП – HDL». Данный подход был ориентирован на высококвалифицированного эксперта по знаниям, способного описать методику реконструкции или синтеза как специализированный модуль знаний.

Однако данная надстройка имела в себе ряд недоработок:

- неполное использование всех типов продукций, определенных в рамках подхода [3];
- отсутствие средств обеспечения монотонности вывода продукций и верификации созданной базы знаний.

В данной работе ставятся задачи развития данного подхода. Цель предлагаемой работы является построение новой версии системы вывода, обеспечивающей полное использование всех типов продукций, определенных в рамках подхода, а также методов и средств обеспечения монотонности вывода продукций и верификация модуля знаний.

1. Обеспечение монотонности вывода продукций.

Данная система обеспечивает накопление данных, то есть при ее работе формируется база знаний предметной области. Система ориентирована на решение задач конструирования сложных объектов. База знаний обладает механизмом отслеживания зависимостей между предположениями и выявления их несовместимости. Механизм работы надстройки представлен на рис. 1.

При стандартном подходе процедура вывода выбирает из ряда продукций, готовых сработать, те, выбор которых позволит наиболее быстро закончить вывод. Для этого применяются различные критерии оптимизации вывода.

В нашем случае требование монотонности, т.е. сужение числа продукций, которые могут сработать при данном пространстве поиска решений, обеспечивается правильным порядком следования продукций.

Монотонный логический вывод - это вывод, при котором БЗ, принимая информацию о новых фактах на данном этапе вывода и определяя, как эти факты могут повлиять на имеющийся результат, обеспечивает требование неизменности результатов предыдущих этапов вывода, при том что итоговый результат должен удовлетворять всем условиям, заданным в фактах [4]. При этом предполагается учитывать, что факты могут иметь необратимые последствия в базе знаний.

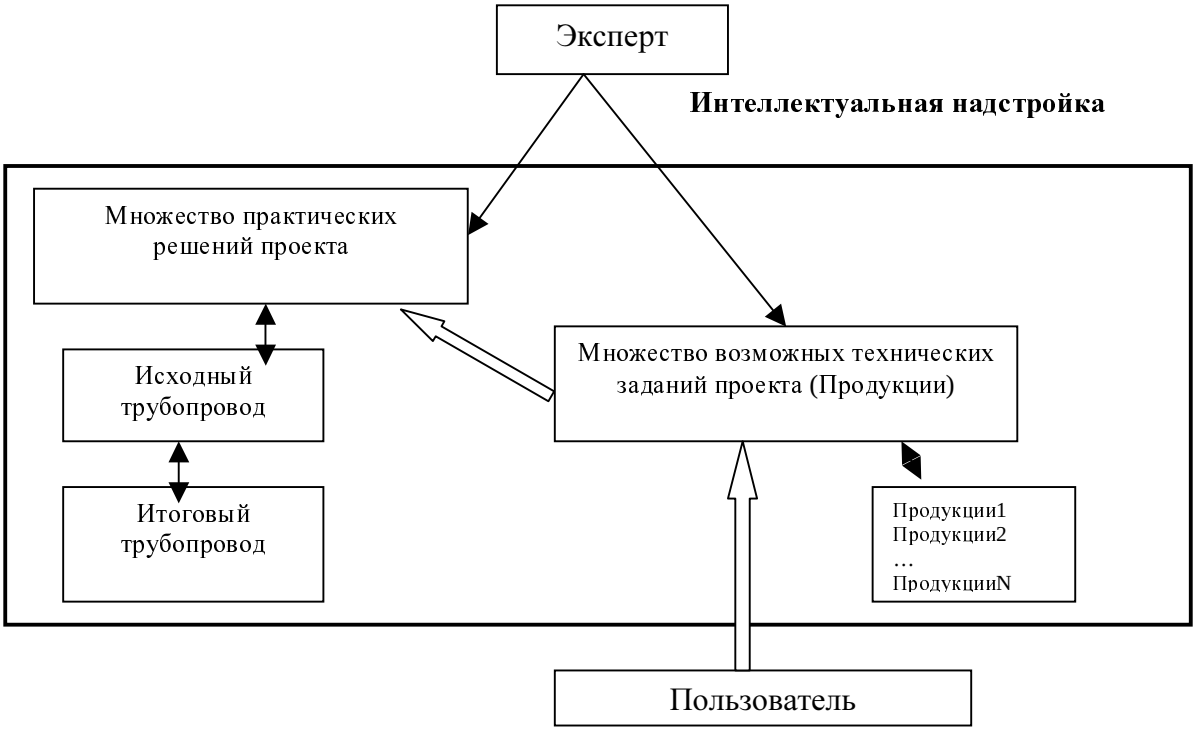

Рис. 1. Механизм работы надстройки

Для обеспечения монотонности в нашем случае предлагается использовать линейный, последовательный порядок выбора продукций для отработки. Исходя из этого, рассмотрим имеющиеся типы продукций и их правильный порядок выполнения, который должен обеспечить инженер по знаниям (эксперт), формирующий базу знаний в предлагаемом подходе.

Продукции могут иметь тип, отличающийся различными значениями четырех параметров, определяющим механизм управления их срабатыванием. Эти параметры задают отношения между различными составляющими продукций и системы управления выводом, т.е. - посылки, вывода и кучи [3], как аналога классной доски. Возможные отношения:

1) посылки с "кучей" - проверяющие и не проверяющие состояние кучи («опускающие голову» и «не опускающие»);

2) посылки с пользователем - спрашивающие и неспрашивающие;

3) вывода с "кучей" - удаляющие и прибавляющие;

4) посылки и вывода - однонаправленные и двунаправленные.

Допустимы любые формы продукций, которые можно получить, исходя из декартового произведения значений атрибутов:

- (оп, неоп)*(спр, неспр)*(удал, приб)*(одно, дву).

Т.е. возможно 16 типов продукций.

Существует два возможных пути обеспечения монотонности вывода в данных условиях:

- создать вначале общий предикат (грамматику), содержащий все возможные решения, а затем отсекать из него лишнее;
- строить не грамматику, а текст готового решения путем последовательного дополнения, достройки решения.

Оба пути обеспечивают монотонность вывода, но требуют использования различных типов продукции на различных этапах построения модуля знаний. Например:

- сначала следуют вопросы к пользователю с формированием грамматики;
- затем выполняется сужение грамматики до одного (нескольких) текста в грамматике.

Второй пример:

- любой вопрос к пользователю только модифицирует данных текст, задающий цель вывода.

Для примера, рассмотрим продукцию с семантикой «если элемент существует» «то должен существовать следующий элемент».

Труба.прямая.Мас[0:2] ->Арматура.вентиль.Мас[0]

Данная продукция может быть охарактеризована, как неспрашивающая, проверяющая, то есть направленная на проверку текущего содержания «кучи» (классной доски, результата вывода). Посылка, как и вывод, имеет модальность «существует». Здесь «Труба» - имя библиотеки, «Прямая» - имя типа, «Мас» - имя массива труб размерностью 3 (нумерация от 0 до 2). Семантика продукции «если существует массив «Мас» из 3 труб, то существует и арматура вентиль».

То есть, при выполнении условия, заданного в посылке (образце) некоторой продукции, как существование некоторого элемента в результате предыдущего вывода вывода, по следствию («то») обязательно должен существовать некоторый другой требуемый элемент. Если данный элемент не существует, то он должен быть добавлен в результат вывода.

Существует три типа библиотек образцов: трубы, арматуры и оборудование. В каждой библиотеке существуют свои типы элементов. Так, например, в библиотеке «Трубы» имеются типы – прямые, уголки и др. В библиотек «Оборудование» – насос и т.д.

Возможна запись данного образца не как наличие блоков, а как наличие структурной связи труб и арматуры через характерные точки.

Характерная точка - это точка какого-либо объекта, которая имеет свойства, принадлежащие только этой точке (такие как координаты, цвет и т.д.), но не принадлежащие всему объекту. Стык

(связь) по характерной точке предполагает некоторую зависимость между значениями свойств, используемых в стыкуемых характерных точках.

Каждый структурный элемент, упомянутый в образце (посылке продукции), в зависимости от типа структурных элементов, к которым он относится, имеет свои определенные общие свойства. Такого рода свойства есть как бы характерная точка всего объекта в целом. Так, например, тип «Труба – прямая», имеет свойства: вес, радиус, длину, материал и др. Возможная продукция:

Труба.прямая.Mac[3]:XT.Mac[3]->

Арматура.вентиль.Mac[0]: Материал[4]=«медь»

Здесь «двоеточие» отделяет в тексте тип блока его свойства, заданные как характерная точка.

Схема расположения труб и арматур при таком задании входных данных изображено на рис. 2.

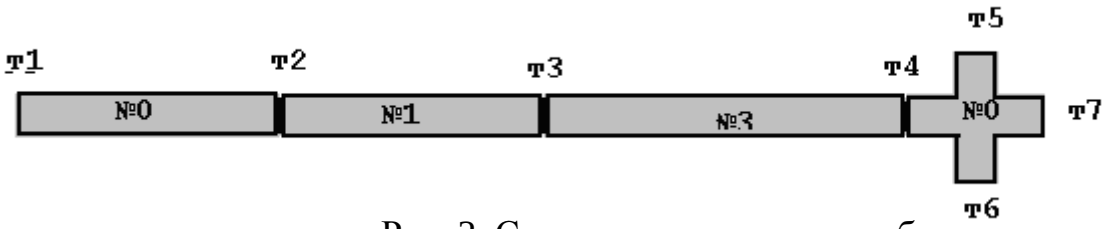

Рис. 2. Схема расположения труб

При этом:

- результаты диалога с пользователем «спрашивающих» продукций рассматривались как исходное множество фактов;
- исходный модифицируемый текст (или грамматика) рассматривался как цель вывода;
- зависимости по существованию элементов в связке «если – то», заданные в совокупности «не спрашивающих, проверяющих» продукций – как совокупность аксиом вывода.

Собственно продукции рассматривались как совокупность правил вывода.

2. Верификация модуля знаний. Верификация - это процесс определения, выполняют ли программные средства и их компоненты требования, наложенные на них в последовательных этапах жизненного цикла разрабатываемой программной системы [5].

Основная цель верификации состоит в подтверждении того, что программное обеспечение соответствует заданным требованиям. Дополнительной целью является выявление и регистрация дефектов и

ошибок, которые внесены во время разработки или модификации программы.

Модель проверки включает в себя:

- идентификацию правильных компонентов, т.е. тип структурного элемента;
- композицию повторных компонентов по их спецификациям;
- формирование общей спецификации компонентной системы, составленной из правильных компонентов [5].

В общем случае верификация базы знаний предполагает проверку правильного порядка выполнения продукций, обеспечивающего требуемый результат вывода. Метод верификации композиции правильных компонентов, может рассматриваться как способ верификации предикатов:

- в посылках продукций;
- в выводах продукций;

В нашем случае можно верифицировать алгоритм функционирования базы знаний как методику проектирования или реконструкции, заданный совокупностью продукций. При этом проверять этот алгоритм можно через факты, задаваемые в посылках продукций. Это может быть либо особенность текста описания реконструированного трубопровода, либо - ответы проектировщика, задаваемые в «спрашивающих» продукциях.

Исходя из имеющихся принципов, можно сделать вывод, что таким образом можно верифицировать образцы, т. е. посылки продукций. Это косвенно позволит верифицировать пространство состояний в процессе вывода как грамматику – сложный предикат.

Рассмотрим верификацию образца на примере, изображенном на рис. 2 при альтернативном выборе решения.

Путь дана продукция типа (a1 ИЛИ a2)->(b1 ИЛИ b2), где:

a1 - Труба.прямая.Mac[3]:XT.Mac[3];

a2 - Труба.прямая.Mac[0]:XT.Mac[1];

b1 - Арматура.вентиль.Mac[0]: T.Mac[4];

b2 - Арматура.вентиль.Mac[0]: T.Mac[7].

При анализе образца, заданного в посылке, система вывода производит сравнение заданных альтернатив с имеющимся множеством текстов, полученным в результате предыдущего вывода («классной доске») и заданном как сложный предикат. В данном случае продукция выявит существования одной соответствующей компоненты решения (a1) и выдаст цепочку выбора решений вида:
Труба.прямая.Mac[3]:XT.Mac[3]->Арматура.вентиль.Mac[0]: T.Mac[4]

Соответственно, вариант «b1» рассматривается как обязательно существующий, для чего выполняется достройка результирующего предиката.

Если вариант «a2» не существует, то решение «b2» не добавляется в готовое решение, т.е. в результирующий предикат.

Тестирование обеспечивает выявление (констатацию наличия) фактов расхождений с требованиями (ошибок). Т.к. ранее было показано, что верификация проводится только на образцах, то и тестирование определяется на входных данных – посылках продукций. В нашем случае верификации подвергается методика проектирования (реконструкции), заданная как совокупность последовательно выполняемых продукций, сведенных в модуль знаний. Для данного случая наиболее подходит метод тестирования базы знаний граничными значениями, что соответствует факторному анализу базы знаний [6]. Фактически это означает следующее. В рассматриваемой продукции для заданного образца в посылке могут выполняться условия:

- не существует ни «a1», ни «a2» (первый крайний случай);
- существует только «a1»;
- существует только «a2»;
- существует и «a1», и «a2» (второй крайний случай);

Соответственно, для проверки работоспособности данной продукции в базе знаний необходимо четыре опыта (4 фактора). Число альтернатив – «2», общее число факторов равно «4».

При проверке работоспособности всего модуля знаний в целом необходимо проверить К факторов:

$$K = \sum_{i=1}^{N} ((Ri)!+1)$$

Здесь:

N – общее число продукций;

Ri – число альтернатив в посылке i-й продукции.

При неполном факторном анализе можно проверить только два крайних случая:

- в требуемом предикате «ничего не существует», т.е. не одна альтернатива не удовлетворяется;
- в требуемом предикате «существует все», т.е. имеют место все альтернативы.

Исходя из этого, выбирается:

- как случай 1 - исходный текст, т.е. описание объекта, подвергаемого реконструкции, где не выполняется посылка не одной продукции;

- как случай 2 - рассматривается не исходный текст, а грамматика, т.е. описание совокупности всех возможных объектов, подвергаемых реконструкции, где выполняется посылка каждой продукции.

Если продукция относится к «спрашивающей», то есть образец есть вопрос к пользователю – проектировщику и включает ряд альтернатив, то, соответственно, предполагается ответ пользователя либо «ничего не подходит», либо – «все подходит».

Выводы. В данной работе получены следующие результаты:

- определена структура и функции новой версии системы вывода, обеспечивающей полное использование всех типов продукций, определенных в рамках подхода;
- разработаны методы обеспечения монотонности вывода продукций и верификация модуля знаний, позволяющие выполнить разработку соответствующих программных средств поддержки в новой версии системы вывода.

Как перспективные задачи работы можно назвать:

- разработка новой программной версии системы вывода, обеспечивающей полное использование всех типов продукций, определенных в рамках подхода;
- разработка соответствующих программных средств поддержки в новой версии системы, как реализацию предложенных методов обеспечения монотонности вывода и верификация модуля знаний.

Литература

1. Григорьев А.В. Методы и средства работы с графическими моделями в САПР парогазовых установок СПРУТ / Моделирование и компьютерная графика: Материалы 1-й международной научно-технической конференции, г Донецк, 04-07 октября 2005 г. — Донецк, ДонНТУ, Министерство образования и науки Украины, 2005. – С. 77-86.

2. Григорьев А.В. Пути создания интеллектуальных САПР при различных уровнях квалификации экспертов /Научно-теоретический журнал «Искусственный интеллект», №3, 2005. – Донецк: ИПИИ МОН и НАН Украины «Наука и образование», 2005. – С. 758–763.

3. Григорьев А.В. Классификация типов продукций в интеллектуальных САПР /Материалы IX-й международной конференции «Интеллектуальные САПР-2004 (CAD 2004)». Том второй. – Москва: Физматлитиздат, 2004 – С. 26–32.

4. А.Н. Аверкин, М.Г. Гаазе-Рапопорт, Д.А. Поспелов. Толковый словарь по искусственному интеллекту.- М, Радио и связь, 1992.

5. С.В. Синицын, Н.Ю. Налютин. Верификация программного обеспечения - http://www.intuit.ru/department/se/verify/

6. В. Томашевский, Е. Жданова. Имитационное моделирование в среде GPSS. – М.: Бестселлер, 2003. – 416 с.

Получено 29.05.09