УДК 510.62:510.662:004.832.38

К.П. Вершинин
Университет Paris XII — Val de Marne, г. Париж, Франция
А.В. Лялецкий, А.Ю. Паскевич
Киевский национальный университет имени Тараса Шевченко, Украина

Применение Системы Автоматизированной Дедукции для верификации математических текстов

Статья посвящена одному подходу к обработке математического текста в стиле программы “Алгоритм Очевидности” (АО) и описывает процедуру верификации текста, реализованную в Системе Автоматизированной Дедукции (САД). Исследования в рамках АО нацелены на создание программных систем автоматизации математической деятельности, в частности, доказательства теорем. Система САД извлекает и накапливает формальные математические знания и использует их для доказательства утверждений в контексте замкнутого математического текста. В основу системы положены: формальный язык представления математических текстов и секвенциальный формализм, определяющий шаг автоматического доказательства. Математический текст, записанный в этом языке, преобразуется в структурированную последовательность формул первого порядка, которая затем верифицируется посредством корректного и полного секвенциального исчисления. Процедура верификации в САД объяснена на примере из формальной арифметики.

The paper is devoted to an approach to mathematical text processing in the style of Evidence Algorithm programme and describes the verification procedure implemented in the System for Automated Deduction (SAD). The Evidence Algorithm programme are aimed at construction of software systems for computer-aided mathematical activity, in particular, for automated theorem proving. The system SAD extracts and accumulates formalized mathematical knowledge and uses it to prove statements in the frame of a self-contained mathematical text. By now, the system SAD exploits a formal language for presenting mathematical texts and a sequent formalism that defines a notion of computer-made proof step. A mathematical text written is that language is translated into a structured sequence of first-order formulas which is verified then by means of a complete and sound sequent calculus. This verification scheme of SAD is explained by an example text from formal arithmetic.

Стаття присвячена одному підходу до обробки математичного тексту у стилі програми “Алгоритм Очевидності” (АО) та змальовує процедуру веріфікації тексту, імплементовану в Системі Автоматизованої Дедукції (САД). Дослідження в рамках АО націлені на створення програмних систем автоматизації математичної діяльності, зокрема, доведення теорем. Система САД видобуває та накопичує формальні математичні знання та використовує їх для доведення тверджень у контексті замкненого математичного тексту. В основу системи покладені: формальна мова поданння математичних текстів та секвенційний формалізм, що визначає шаг автоматичного доведення. Математичний текст, записаний цією мовою, перетворюється на структуровану послідовність формул першого порядку, яка потім веріфікується за допомогою коректного та повного секвенційного числення. Процедура веріфікації в САД пояснена на прикладі з формальної арифметики.

УДК 510.62:510.662:004.832.38

К.П. Вершинин
Университет Paris XII — Val de Marne, г. Париж, Франция
А.В. Лялецкий, А.Ю. Паскевич
Киевский национальный университет имени Тараса Шевченко, Украина

Применение Системы Автоматизированной Дедукции для верификации математических текстов[1]

Статья посвящена одному подходу к обработке математического текста в стиле программы “Алгоритм Очевидности” (АО) и описывает процедуру верификации текста, реализованную в Системе Автоматизированной Дедукции (САД). Исследования в рамках АО нацелены на создание программных систем автоматизации математической деятельности, в частности, доказательства теорем. Система САД извлекает и накапливает формальные математические знания и использует их для доказательства утверждений в контексте замкнутого математического текста. В основу системы положены: формальный язык представления математических текстов и секвенциальный формализм, определяющий шаг автоматического доказательства. Математический текст, записанный в этом языке, преобразуется в структурированную последовательность формул первого порядка, которая затем верифицируется посредством корректного и полного секвенциального исчисления. Процедура верификации в САД объяснена на примере из формальной арифметики.

Введение

В настоящей статье результаты последних исследований в рамках программы Алгоритм Очевидности (АО) рассматриваются с точки зрения задачи верификации математических текстов. Алгоритм Очевидности был основан академиком В.М. Глушковым в конце 1960-ых годов как программа исследований по автоматизации математической деятельности [1]. Основная задача АО состоит в обработке математических текстов, в том числе, в построении длинных, но, в некотором смысле, "очевидных" доказательств, и в их верификации. В.М. Глушков предложил вести параллельные исследования в нескольких направлениях: формальные языки для представления математических текстов в наиболее удобной для пользователя форме; понятие шага автоматического доказательства и его эволюция; информационная среда АО, влияющая на “очевидность” отдельного шага доказательства; интерактивный режим поиска доказательства.

Результатом явилась Система Автоматизированной Дедукции, САД [5,6]. Последняя программная реализация САД, описанная в настоящей статье, работает по следующей схеме.

Математический текст формализуется посредством языка ForTheL, который, с одной стороны, имеет формально заданные синтаксис и семантику, а, с другой стороны, приближен к естественному языку математических публикаций. Текст, записанный в ForTheL, автоматически преобразовывается, с сохранением структуры и сигнатуры, в совокупность формул первого порядка.

Преобразованный текст образует информационную среду, в которой решаются дедуктивные задачи трех различных видов: установление

[1] Эта работа частично поддержана проектом INTAS 2000-447

выводимости заданной секвенции, доказательство некоторого утверждения в контексте исходного ForTheL-текста, верификация исходного ForTheL-текста.

Краткое описание двух первых видов дедуктивной обработки текста приводится в [5,6,7]. Данная статья посвящена проблеме верификации текста и ее решению в САД. Несмотря на то, что в основу всех программных реализаций САД положены одни и те же принципы — язык ForTheL как средство представления текста и секвенциальное исчисление GD [5] как базовая дедуктивная техника — переориентация системы на задачи верификации значительно повлияла на её архитектуру, её языковой и дедуктивный аппарат. Эти изменения и являются предметом настоящей статьи.

Web-интерфейс САД доступен по адресу http://ea.unicyb.kiev.ua.

Архитектура САД

Текущая реализация САД может действовать как верификатор формальных текстов. В течение одной сессии верификации САД выполняет три следующих задания: (а) преобразование входного ForTheL-текста во внутреннее представление; (б) установление последовательности целевых утверждений, подлежащих верификации (доказательству); (в) поиск логического вывода каждой цели из её логических предшественников в обрабатываемом тексте.

Система САД состоит из четырех основных модулей: [ForTheL], [FOL], [Reason] и [Moses]. Ниже мы приводим краткое описание каждого модуля.

Модули [ForTheL] *и* [FOL] осуществляют синтаксический анализ входных ForTheL-текстов и текстов первого порядка, соответственно. Каждый из этих модулей переводит текст в соответствующее внутреннее представление. Это преобразование сохраняет структуру исходного текста и переводит фразы как формулы первого порядка. Формулы, полученные переводом ForTheL-фраз, могут содержать специальные метки (*аннотации*), которые используются при поиске доказательства. Так, модуль [ForTheL] отмечает “главный” атом в определениях. В текстах первого порядка фразы уже являются формулами первого порядка, поэтому модуль [FOL] лишь устраняет некоторый “синтаксический сахар”.

Внутреннее представление входного текста, которое было построено модулями [ForTheL] или [FOL] служит информационной средой для дальнейших действий системы.

Модуль [Reason] запускает цикл верификации и назначает задания пруверу. Цикл верификации задается следующим алгоритмом (далее в тексте, внутреннее представление входного ForTheL-текста называется просто “текст”; специальный указатель отмечает обрабатываемый блок текста):

1. Установить указатель на начало текста; положить список логических предшественников (*контекст*) равным пустому списку.
2. Передвинуть указатель на следующий блок U в тексте. В зависимости от типа U (утверждение, предположение, теорема, аксиома, доказательство, и т.д.), определить, является ли U целью, т.е. подлежит ли U верификации. Если да, перейти к следующему шагу, иначе перейти к шагу 6.

3. Если U является разделом, запустить “внутренний” цикл верификации для этого раздела. После завершения этого цикла перейти к шагу 6. Иначе, если U есть отдельная фраза (формула), перейти к следующему шагу.
4. Попытаться упростить рассматриваемую цель U . На данном этапе [Reason] применяет различные эвристики и тактики. К примеру, если цель U является конъюнкцией двух формул, не имеющих общих свободных переменных, то U может быть разбито на две независимо доказываемые подцели.
5. Для всех подцелей, которые не могут быть упрощены далее, запустить [Moses], прувер системы САД. Если [Moses] не находит доказательство и нет альтернативных подцелей, цикл верификации завершается неудачей (диагностика “verification failed”). Иначе, рассматривается следующая подцель. Когда все подцели U доказаны, U считается верифицированной.
6. Если U не является последним блоком в разделе (или во всем тексте), добавить U в контекст и перейти к шагу 2. Если U завершает раздел, текущий цикл верификации оканчивается успешно, и [Reason] продолжает “внешний” цикл верификации. Если же U является последним блоком в тексте, то текст считается верифицированным (диагностика “verification successful”).

Модуль [Moses] предназначен для поиска доказательства. До начала процедуры поиска вывода, [Moses] подготавливает дедуктивную среду для работы процедуры: находит дизъюнктивно связанные литеры, устанавливает зависимости между переменными, фиксирует пары дополнительных литер и так далее.

Напомним, что прувер реализован на основе секвенциального исчисления GD, описанного в [5]. Прувер [Moses] запускает процедуру поиска логического вывода, которая осуществляет исчерпывающий поиск в глубину с инкрементным увеличением порога глубины и бэктрекингом. Для повышения эффективности поиска вывода, [Moses] использует специальные ограничения (*констрэйнты*) и технику “folding-up”. Недавно была разработана и реализована специальная техника применения определений.

Для обработки равенства, в [Moses] реализована некоторая разновидность метода модификации Бранда [8].

Заметим, что исчисление GD не изменяет исходных посылок, поэтому прувер может оперировать деревом литерных целей вместо дерева секвенций. Это значительно сокращает потребность в вычислительных ресурсах.

Благодаря отсутствию предварительной сколемизации, прувер САД может использовать для решения систем уравнений любую подходящую систему (*солвер*). Подмодуль [Moses], отвечающий за решение уравнений, действует как посредник между прувером и внешними солверами. Этот подмодуль проверяет найденную солвером подстановку на допустимость и в случае необходимости формулирует дополнительные уравнения. В качестве основного солвера используется процедура вычисления наиболее общего унификатора.

Лингвистические особенности системы САД

Любой язык, предназначенный для представления формальных текстов в стиле АО, должен удовлетворять следующим требованиям. Синтаксис и семантика такого языка должны быть формально заданы. Язык должен позволять записывать аксиомы теории, теоремы, доказательства и определения, чтобы

обеспечить самодостаточность (*замкнутость*) текста. Вместе с тем, язык должен быть приближен к естественному языку математических публикаций. Это облегчает пользователю написание и обработку текста, позволяет создать удобный интерактивный режим обработки.

Последняя версия языка ForTheL [2] вполне удовлетворяет вышеперечисленным требованиям. Перечислим его основные особенности.

ForTheL-текст представляет собой последовательность разделов, фраз и специальных конструкций, таких как описания шаблонов. Фразы являются либо предположениями (в этом случае они начинаются словами "`let`" или "`assume`"), либо утверждениями. Разделы состоят из фраз и/или разделов нижнего уровня. Типичные разделы верхнего уровня: аксиома, определение, теорема. Типичные разделы нижнего уровня: доказательство и случай в доказательстве.

Каждый раздел верхнего уровня есть последовательность предположений, за которой следует утверждение. Утверждение теоремы или аксиомы может быть произвольным ForTheL-утверждением, тогда как утверждение определения принимает одну из нескольких форм, в зависимости от типа определяемого объекта.

Грамматика ForTheL-фраз имитирует грамматику английских предложений. Фразы строятся из существительных, обозначающих функции или понятия (классы), из глаголов и прилагательных, обозначающих предикаты, из предлогов, союзов и служебных слов, определяющих логический смысл сложного предложения. Вот пример простого ForTheL-утверждения: "`Every closed subset of every compact set is compact`".

Текущий тезаурус может быть расширен при помощи специальных конструкций верхнего уровня: описаний шаблона. Пример описания шаблона унарного понятия: "`[an element/elements of x]`". Если слово, использованное в шаблоне, имеет несколько форм, они могут быть перечислены через косую черту. Шаблон может быть введен и как синоним некоторого выражения того же типа, при этом описание шаблона имеет вид "`[шаблон @ значение]`". Единственный предопределенный ForTheL-шаблон — это шаблон предиката "`[x is equal to y]`".

Любое утверждение в тексте, например, утверждение теоремы, может сопровождаться доказательством. Раздел "доказательство" представляет собой последовательность предположений, утверждений (возможно, имеющих собственные доказательства) и подразделов "случай в доказательстве".

Когда верификатор обрабатывает доказательство или случай в доказательстве, он запускает отдельный, внутренний цикл верификации. Если раздел верифицирован успешно, то его *формульный образ* присоединяется к контексту внешнего процесса верификации. Образ раздела "доказательство" используется только для верификации утверждения, обосновываемого этим доказательством, а затем удаляется из контекста. Образ раздела "случай в доказательстве" используется до конца внешнего раздела.

Формульный образ раздела — это формула, построенная на основе содержимого этого раздела. В этой формуле предположения становятся антецедентами импликаций, а утверждения и подразделы "случай" становятся конъюнктами. Каждая переменная, объявленная внутри раздела, связывается универсальным квантором в формульном образе. Так, образом раздела вида

“`case. let X be N. then P. assume H. then Q. end.`” будет формула $\forall X(N(X) \supset (P \wedge (H \supset Q)))$.

Кроме доказательства, отдельное ForTheL-утверждение может быть снабжено *ссылками*: перечислением имен предыдущих разделов. Эти разделы получают более высокий приоритет при верификации этого утверждения.

Логические особенности системы САД

Как было сказано выше, для построения эффективных методов поиска доказательства был предложен секвенциальный формализм [3,5]. Такой выбор обусловлен тем, что секвенциальный подход более близок к естественному способу доказательства, чем резолюционный. В частности, нет необходимости преобразовывать текст во множество дизъюнктов, содержащих сколемовские функциональные символы. Это облегчает диалог с пользователем или с внешним математическим сервисом (например, специальным солвером).

Дедуктивная техника, используемая в САД, обладает следующими свойствами: доказательство выбранной цели сводится к доказательству некоторой совокупности новых подцелей (целецправляемость); оригинальная техника работы с кванторами, не использующая предварительную сколемизацию, позволяет работать в сигнатуре исходной теории.

Верификация в САД

В этом разделе мы рассмотрим простой замкнутый ForTheL-текст, описывающий некоторые свойства порядка на натуральных числах в определенном подмножестве арифметики Пеано. Этот текст полностью верифицируется системой САД за пятнадцать секунд на рабочей станции Pentium IV 1.8 GHz station.

Мы рассматриваем текст по разделам, сопровождая каждый фрагмент ForTheL комментариями и диагностикой САД.

Аксиомы нуля и последователя.

```
[a number] [the zero] [the successor of x]
[x is nonzero @ x is not equal to zero]

Axiom _NatZero. Zero is a number.

Axiom _NatSucc. The successor of every number is a number.

Axiom SuccNotZero. The successor of no number is zero.

Axiom SuccEquSucc. Let A be a number and B be a number.
  If the successor of A is the successor of B then A is equal to B.
```

Обратите внимание на понятие “`number`” и на аксиомы “`_NatZero`” и “`_NatSucc`”. Язык ForTheL (в его текущей реализации) требует, чтобы каждая переменная была объявлена как представитель некоторого понятия. Поэтому необходимо ввести понятие натурального числа и постулировать, что значения функций “`zero`” и “`successor of`” являются натуральными числами.

Эти две аксиомы используются практически в каждом доказательстве, поэтому желательно, чтобы прувер никогда не "терял их из виду", даже в

присутствии ссылок на другие аксиомы или теоремы. Чтобы указать на это, имена этих аксиом начинаются с подчерка.

Приведенный фрагмент преобразуется транслятором в следующие формулы первого порядка:

_NatZero: $Number(Zero)$

_NatSucc: $\forall x_1 \left(Number(x_1) \supset Number(SuccessorOf(x_1)) \right)$

SuccNotZero: $\forall x_1 \left(Number(x_1) \supset SuccessorOf(x_1) \neq Zero \right)$

SuccEquSucc: $\forall A, B \left(\left(Number(A) \wedge Number(B) \right) \supset \supset \left(SuccessorOf(A) = SuccessorOf(B) \supset A = B \right) \right)$

Введение сложения.

```
[the sum of x and y]

Axiom AddZero. Let A be a number.
  The sum of A and zero is equal to A.

Axiom AddSucc. Let A be a number and B be a number.
  The sum of A and the successor of B is equal
    to the successor of the sum of A and B.
```

Эти две аксиомы являются примером примитивно рекурсивного определения. В настоящее время мы работаем над дальнейшим расширением языка и дедуктивного инструментария САД, чтобы позволить формулировать и обрабатывать такие определения естественным и эффективным способом.

Вспомогательные аксиомы. На текущий момент, САД не позволяет вести доказательство по индукции. Поэтому следующие факты приходится постулировать как аксиомы.

```
Axiom ZeroOrSucc.
  Every nonzero number is the successor of some number.

Axiom _NatAdd. The sum of every number and every number is a number.

Axiom AssoAdd.
  Let A be a number and B be a number and C be a number.
  The sum of A and the sum of B and C is equal
    to the sum of (the sum of A and B) and C.

Axiom InjAdd.
  Let A be a number and B be a number and C be a number.
  If the sum of A and B is equal to the sum of A and C
    then B is equal to C.

Axiom Diff. Let A be a number and B be a number.
  There exists a number C such that
    B is the sum of A and C or A is the sum of B and C.
```

Аксиома “ZeroOrSucc” переводится следующей формулой:

$$\forall x_1 \left(\left(Number(x_1) \wedge x_1 \neq Zero \right) \supset \exists x_2 \left(Number(x_2) \wedge x_1 = SuccessorOf(x_2) \right) \right)$$

Введение упорядочения. Порядок на множестве натуральных чисел определяется с помощью сложения:

```
[x is less than y] [x is greater than y @ y is less than x]

Definition DefLess.  Let A be a number and B be a number.
  A is less than B  iff  B is equal to
    the sum of A and the successor of some number.
```

Формульным образом этого определения будет:

$$\forall A,B\left(\left(Number(A)\wedge Number(B)\right)\supset\right.$$
$$\left.\supset\left(LessThan(A,B)\equiv\exists x_1\left(Number(x_1)\wedge B=SumOfAnd(A,SuccessorOf(x_1))\right)\right)\right)$$

Доказательство антирефлексивности. Первой целью, подлежащей верификации, становится антирефлексивность отношения "`less than`".

```
Theorem NReflLess. Let A be a number.
  A is not less than A.
Proof.
  Assume the contrary.
  Let C be a number such that
    A is equal to the sum of A and the successor of C.
  Then the successor of C is zero (by AddZero,InjAdd).
  We have a contradiction.
qed.
```

В разделах "доказательство" и "случай" специальные утверждения "`thesis`" и "`contrary`" обозначают ближайшее утверждение, которое обосновывается доказательством, и его отрицание, соответственно. Таким образом, первая фраза в вышеприведенном доказательстве переводится как "`assume not not LessThan(A,A)`".

Второе предположение в доказательстве вводит переменную `C`. На данном этапе, САД не пытается доказать существование натурального числа с такими свойствами.

Чтобы верифицировать теорему "`NReflLess`", верификатор должен доказать три цели: (а) последователь `C` равен нулю; (б) этот факт противоречит сделанным допущениям; (в) это противоречие влечет утверждение теоремы.

Первая цель не является очевидной: чтобы доказать ее, необходимо учесть, что прибавление нуля не изменяет числа (аксиома "`AddZero`"), и что никакое число, кроме нуля, не обладает этим свойством (аксиома "`InjAdd`"). Существенно также то, что последователь `C` является натуральным числом. Однако, мы не упоминаем соответствующую аксиому среди ссылок, поскольку она является приоритетной по умолчанию. Система САД дает следующую диагностику при доказательстве этой цели:

```
[Reason] line 85: encountered a goal
[Moses] launch (time limit: 10 sec, initial db: 1, final db: 100)
[Moses] proved in 0 msec - proof nodes: 25 - proof depth: 4
[Moses] steps: 450 - born nodes: 1087 - depth bound: 4
```

Вторая строка указывает параметры сессии поиска доказательства. Пруверу отведено десять секунд, поиск начинается с порога глубины (максимальная высота дерева вывода) равного единице, и этот порог может быть увеличен до глубины 100.

Две следующие строки сообщает результат поиска. Доказательство было найдено “почти немедленно”, дерево доказательства содержит 25 литер и имеет высоту 4. В процессе поиска, [Moses] сделал 450 шагов вывода (расширений и закрытий ветки) и породил 1087 узлов. Последнее значение порога равно 4.

Мы не приводим здесь дерево доказательства. Пример доказательства, порожденного САД, будет дан ниже.

Заметим, что доказательство было найдено так быстро благодаря ссылкам на релевантные аксиомы. Без этих подсказок поиск занял бы намного больше времени.

Теперь, равенство некоторого последователя нулю немедленно противоречит аксиоме “SuccNotZero”. Доказательство должно быть простым и нет необходимости в подсказках пруверу. Однако, [Moses] совершает довольно много шагов, прежде чем найти это, в самом деле тривиальное, доказательство (6 литер в дереве доказательства):

```
[Reason] line 86: encountered a goal
[Moses] launch (time limit: 10 sec, initial db: 1, final db: 100)
[Moses] proved in 5280 msec - proof nodes: 6 - proof depth: 3
[Moses] steps: 649366 - born nodes: 1161172 - depth bound: 3
```

Такая «неэффективность» может быть объяснена двумя соображениями. Во-первых, доказывается не конкретная формула, а абстрактное “противоречие”. Поэтому [Moses] пытается доказать отрицание каждой формулы в разделе при текущем значении порога глубины, прежде чем увеличить порог. Во-вторых, в задачах, содержащих равенство, возможно гораздо большее число различных шагов вывода по сравнению с задачами без равенства.

Наконец, необходимо убедиться, что полученное противоречие влечет утверждение теоремы. Чтобы доказать это, прувер должен проверить, что все остальные предположения помимо отрицания тезиса, являются выполнимыми. В данном случае, необходимо доказать существование числа C. Заметим, что текущей целью является утверждение “A is not less than A”, которое расположено в ForTheL-тексте выше предыдущих целей.

```
[Reason] line 80: encountered a goal
[Moses] launch (time limit: 180 sec, initial db: 1, final db: 100)
[Moses] proved in 10 msec - proof nodes: 26 - proof depth: 3
[Moses] steps: 938 - born nodes: 1917 - depth bound: 3
```

Доказательство транзитивности. Ассоциативность сложения влечет транзитивность порядка:

```
Theorem TransLess.
  Let A be a number and B be a number and C be a number.
  Assume A is less than B and B is less than C.
  Then A is less than C.
Proof.
  Let M be a number and N be the successor of M.
  Let P be a number and Q be the successor of P.
```

```
  Assume the sum of A and N is equal to B.
  Assume the sum of B and Q is equal to C.
  Let S be the sum of N and Q.
  S is the successor of the sum of N and P (by AddSucc).
  The sum of A and S is equal to C (by AssoAdd).
  Hence the thesis (by DefLess).
qed.
```

Верификация этой теоремы состоит в доказательстве четырех целей. Диагностика САД для этого фрагмента текста следует ниже.

```
[Reason] line 101: encountered a goal
[Moses] launch (time limit: 10 sec, initial db: 1, final db: 100)
[Moses] proved in 50 msec - proof nodes: 26 - proof depth: 5
[Moses] steps: 6222 - born nodes: 11769 - depth bound: 5
[Reason] line 102: encountered a goal
[Moses] launch (time limit: 10 sec, initial db: 1, final db: 100)
[Moses] proved in 10 msec - proof nodes: 39 - proof depth: 4
[Moses] steps: 1605 - born nodes: 3206 - depth bound: 4
[Reason] line 103: encountered a goal
[Moses] launch (time limit: 10 sec, initial db: 1, final db: 100)
[Moses] proved in 280 msec - proof nodes: 35 - proof depth: 5
[Moses] steps: 35685 - born nodes: 85354 - depth bound: 5
[Reason] line 94: encountered a goal
[Moses] launch (time limit: 10 sec, initial db: 1, final db: 100)
[Moses] proved in 1560 msec - proof nodes: 50 - proof depth: 4
[Moses] steps: 233910 - born nodes: 529587 - depth bound: 4
```

Хотя последнее утверждение в разделе доказательства есть просто “`thesis`” (т.е. совпадает с утверждением теоремы), прувер строит достаточно большое дерево (50 литер), чтобы завершить доказательство теоремы. Причина в том, что “`thesis`” в строке 103 был доказан в предположении существования чисел `M`, `N`, `P`, `Q` с необходимыми свойствами. Выполнимость этого предположения доказывается в последнем запуске прувера для строки 94.

Доказательство антисимметричности. Антисимметричность немедленно следует из антирефлексивности и транзитивности. Лемма не сопровождается доказательством: [Moses] доказывает ее менее, чем за одну миллисекунду.

```
Lemma ASymmLess. Let A be a number and B be a number less than A.
  Then A is not less than B.
```

Доказательство тотальности. Для верифицирования следующей теоремы, САД запускает семь сессий поиска доказательства. В общей сложности, верификация завершается за 3730 миллисекунд, после 564302 шагов вывода, породивших 1022099 узлов.

```
Theorem TotalLess.
  Let A be a number and B be a number not equal to A.
  Then A is less than B or B is less than A.
Proof.
  Let C be a number such that
    A is the sum of B and C or B is the sum of A and C.
  If C is zero then B is equal to A.
  Hence C is the successor of some number.
```

```
  If B is the sum of A and C then A is less than B.
  Then A is the sum of B and C or A is less than B.
  If A is the sum of B and C then B is less than A.
  Hence the thesis.
qed.
```

Введение умножения. Теперь мы определяем умножение натуральных чисел. Утверждения "`_NatMul`" и "`Monot`" доказываются в арифметике Пеано по индукции. Здесь они вводятся как аксиомы.

```
[the product of x and y]

Axiom MulZero. Let A be a number.
  The product of A and zero is equal to zero.

Axiom MulSucc. Let A be a number and B be a number.
  The product of A and the successor of B is equal
    to the sum of A and the product of A and B.

Axiom _NatMul.
  The product of every number and every number is a number.

Axiom Monot. Let A be a number and B be a number and C be a number.
  Assume A is greater than B and C is nonzero.
  Then the product of A and C is greater
    than the product of B and C.
```

Доказательство разбором случаев. Следующая теорема переформулирует аксиому "`Monot`" для нестрогого упорядочения. Мы рассматриваем эту теорему, чтобы продемонстрировать более сложное ForTheL-доказательство.

```
Theorem MonotNSt.
  Let A be a number and B be a number and C be a number.
  Assume A is not less than B.
  Then the product of A and C is not less
    than the product of B and C.
Proof.
  Case 1.
    Assume A is equal to B or C is zero.
    Then the product of A and C is equal
      to the product of B and C (by MulZero).
    Hence the thesis (by NReflLess).
  end.
  Case 2.
    Assume A is not equal to B and C is nonzero.
    Hence the thesis (by Monot,TotalLess,ASymmLess). # <--
  end.
qed.
```

Ниже приводится дерево доказательства, построенное прувером для цели, помеченной вверху стрелкой. Дерево представлено в упрощенной форме: опущены поддоказательства принадлежности классу натуральных чисел и применения аксиомы рефлексивности $\forall x(x = x)$. Поэтому число узлов, сообщенное в диагностике, превосходит число узлов в показанном дереве.

В приведенном дереве, один узел занимает одну строку; высота узла определяется смещением строки от левого края; последовательно идущие узлы одной высоты обозначают последовательность дизъюнктивно связанных литер. Корневой узел занимает литера "`-LessThan(ProductOfAnd(A,C), ProductOfAnd(B,C))`" — утверждение теоремы.

```
[Moses] launch (time limit: 10 sec, initial db: 1, final db: 100)
[Moses] proof tree (58 nodes, 5 levels):
[Moses]  1: -LessThan(ProductOfAnd(A, C), ProductOfAnd(B, C))
[Moses]  2:    LessThan(ProductOfAnd(A, C), ProductOfAnd(B, C))
[Moses]  3:    LessThan(ProductOfAnd(B, C), ProductOfAnd(A, C))
[Moses]  4:      -LessThan(ProductOfAnd(B, C), ProductOfAnd(A, C))
[Moses]  5:      -EQU(C, Zero)
[Moses]  6:         EQU(C, Zero)
[Moses]  7:      LessThan(B, A)
[Moses]  8:         -LessThan(B, A)
[Moses]  9:         -LessThan(A, B)
[Moses] 10:            LessThan(A, B)
[Moses] 11:         -EQU(A, B)
[Moses] 12:            EQU(A, B)
[Moses] proved in 300 msec - proof nodes: 58 - proof depth: 5
[Moses] steps: 55456 - born nodes: 141743 - depth bound: 5
```

Это доказательство может быть прочитано снизу вверх, т.е. от посылок к целям, следующим образом (в скобках указано обоснование соответствующего утверждения и строка в вышеприведенном дереве доказательства, где находится это утверждение):

1. A не равно B (предположение, строка 11)
2. A не меньше B (условие теоремы, строка 9)
3. Поэтому B меньше A ("`TotalLess`", строка 7)
4. C не равно нулю (предположение, строка 5)
5. Произведение B и C меньше произведения A и C ("`Monot`", строка 3)
6. Следовательно, произведение A и C не меньше произведения B и C ("`ASymmLess`", строка 1), что и требовалось.

В настоящее время разрабатывается модуль САД, отвечающий за перевод деревьев, порождаемых прувером [Moses] в ForTheL-доказательства.

Теорема "`MonotNSt`" завершает рассматриваемый ForTheL-текст, и сессия верификации заканчивается успешно.

Заключение

Текущая версия САД предназначена для выполнения следующих заданий:

- установление выводимости в классической логике первого порядка;
- доказательство теорем, записанных в языке ForTheL и погруженных в замкнутый ForTheL-текст;
- верификация замкнутых ForTheL-текстов.

Таким образом, система САД может быть полезной при решении следующих задач: распределенное автоматическое доказательство теорем, верификация математических публикаций, извлечение знаний из математических текстов, компьютерное обучение математическим дисциплинам, построение баз знаний математических теорий, интеграция компьютерной алгебры в дедукцию.

Кроме того, система может быть применена для решения логических задач в теории принятия решений, для верификации программного и аппаратного обеспечения и так далее.

Список литературы

1. Глушков В.М. Некоторые проблемы теории автоматов и искусственного интеллекта // Кибернетика. – 1970. – Т. 2. – С. 3-13.
2. K. Vershinin, A. Paskevich. ForTheL — the Language of Formal Theories // IJ Information Theories and Applications. – 2000. – V. 7-3. – P. 121-127.
3. A. Degtyarev, A. Lyaletski, M. Morokhovets. Evidence Algorithm and Sequent Logical Inference Search // Lecture Notes in Artificial Intelligence. – 1999. – V. 1705. – P. 44-61.
4. A. Degtyarev, A. Lyaletski, M. Morokhovets. On the EA-Style Integrated Processing of Self-Contained Mathematical Texts // Symbolic Computation and Automated Reasoning (Proc. of CALCULEMUS-2000 Symposium). – A.K. Peters, Ltd., USA – 2001. – P.126-141.
5. K. Verchinine, A. Degtyarev, A. Lyaletski, A. Paskevich. SAD, a System for Automated Deduction: a Current State // Proceedings of the Workshop on 35 Years of Automating Mathematics. – Edinburgh, Great Britain. – 2002. – 12 p.
6. Z. Aselderov, K. Verchinine, A. Degtyarev, A. Lyaletski, A. Paskevich, A.Pavlov. Linguistic Tools and Deductive Technique of the System for Automated Deduction // Proceedings of of the 3rd International Workshop on the Implementation of Logics. – Tbilisi, Georgia. – 2002. – P. 21-24.
7. Асельдеров З.М., Вершинин К.П., Дегтярев А.И., Лялецкий А.И., Паскевич А.Ю. Особенности обработки математических текстов в Системе Автоматизированной Дедукции (САД) // Искусственный Интеллект. – 2002. – Т. 4 (труды Третьей Международной Конференции «Искусственный Интеллект»). – С. 164-171.
8. D. Brand. Proving Theorems with the Modification Method // SIAM Journal of Computing. – 1975. – v. 4. – P. 412-430.