Компьютерные доказательства

Л.Д. Беклемишев

Математический институт им. В.А. Стеклова

20 марта 2009 г.

Формальные аксиоматические теории

Аксиоматический метод Гильберта предполагает явную формулировку всех предположений теории и допускает лишь чисто логические выводы из этих посылок (в частности, запрещены опора на зрительную интуицию, рассуждения по аналогии и т.д.)

Логический вывод может быть записан в символьном виде,[1] что превращает его в вычислительный процесс: «игру» в переписывание логических выражений по определенным правилам. Это привело к созданию *формальных аксиоматических теорий* в начале 20-го века (Frege, Peano, Russell, Whitehead).

[1] используя логические связки $\rightarrow$ (влечет), $\&$ (и), $\neg$ (не) и кванторы $\exists$ (существует), $\forall$ (для всех).

Стандартные «универсальные» теории в математической логике

Арифметика Пеано PA: формализует «математику конечного»; основана на аксиомах для натуральных чисел с операциями $+$ и $\cdot$.

Теория множеств Цермело–Френкеля ZFC: формализует *всю* обычную математику; основана на аксиомах для множеств и отношения принадлежности.

Формализм *ZFC* и *PA* (в рамках гильбертовского исчисления предикатов) является простой и удобной теоретической моделью, однако плохо приспособлен к нуждам *реальной* формализации математики.

Подсчитано, что развёрнутая запись числа 1 в формальной теории множеств из книги Бурбаки (вариант *ZFC*) содержит более $4 \cdot 10^9$ символов[2].

- Доказательство существования формального объекта (терма или вывода) — не то же самое, что его реальное построение.

[2] Точнее, 4 523 659 424 929 символов (A. Mathias)

Формализм *ZFC* и *PA* (в рамках гильбертовского исчисления предикатов) является простой и удобной теоретической моделью, однако плохо приспособлен к нуждам *реальной* формализации математики.

Подсчитано, что развёрнутая запись числа 1 в формальной теории множеств из книги Бурбаки (вариант *ZFC*) содержит более $4 \cdot 10^9$ символов[2].

- Доказательство существования формального объекта (терма или вывода) — не то же самое, что его реальное построение.

[2] Точнее, 4 523 659 424 929 символов (A. Mathias)

Формализм *ZFC* и *PA* (в рамках гильбертовского исчисления предикатов) является простой и удобной теоретической моделью, однако плохо приспособлен к нуждам *реальной* формализации математики.

Подсчитано, что развёрнутая запись числа 1 в формальной теории множеств из книги Бурбаки (вариант *ZFC*) содержит более $4 \cdot 10^9$ символов[2].

- Доказательство существования формального объекта (терма или вывода) — не то же самое, что его реальное построение.

[2] Точнее, 4 523 659 424 929 символов (A. Mathias)

Зачем нужна формализация доказательств?

Утопия:

Манифест «QED» (Springer LNAI, 814(1994), 238-251):
Призыв к созданию всеобъемлющей компьютерной базы знаний по математике, вместе с механизмом проверки (верификации) формальных доказательств и инструментами, облегчающими поиск доказательств и манипуляции с ними.

http://www.cs.ru.nl/~freek/qed/qed.html

Цели:

- Повышение надёжности сложных математических доказательств.
- Постановка процесса журнального рецензирования (peer review) на объективную основу.
- Облегчение поиска доказательств, избавление математиков от рутины.
- Распостранение надёжных математических доказательств на доказательства корректности работы программ.

Цели:

- Повышение надёжности сложных математических доказательств.
- Постановка процесса журнального рецензирования (peer review) на объективную основу.
- Облегчение поиска доказательств, избавление математиков от рутины.
- Распостранение надёжных математических доказательств на доказательства корректности работы программ.

Цели:

- Повышение надёжности сложных математических доказательств.
- Постановка процесса журнального рецензирования (peer review) на объективную основу.
- Облегчение поиска доказательств, избавление математиков от рутины.
- Распостранение надёжных математических доказательств на доказательства корректности работы программ.

Цели:

- Повышение надёжности сложных математических доказательств.
- Постановка процесса журнального рецензирования (peer review) на объективную основу.
- Облегчение поиска доказательств, избавление математиков от рутины.
- Распостранение надёжных математических доказательств на доказательства корректности работы программ.

Для этого должна развиваться

технология формальных/компьютерных доказательств
(formal proof technology)

Нужно уметь строить формальные доказательства и манипулировать с ними: переводить из одного формата в другой, упрощать, извлекать полезную информацию, и т.д.

Словарь

- prover = theorem prover (система поиска выводов, прувер)
- automated theorem prover (система автоматического поиска выводов)
- proof-assistant = interactive theorem prover (система интерактивного поиска выводов)
- proof-checker (верификатор доказательств)
- proof-object = proof-term (явный формальный вывод)

Словарь

- prover = theorem prover (система поиска выводов, прувер)
- automated theorem prover (система автоматического поиска выводов)
- proof-assistant = interactive theorem prover (система интерактивного поиска выводов)
- proof-checker (верификатор доказательств)
- proof-object = proof-term (явный формальный вывод)

Словарь

- prover = theorem prover (система поиска выводов, прувер)
- automated theorem prover (система автоматического поиска выводов)
- proof-assistant = interactive theorem prover (система интерактивного поиска выводов)
- proof-checker (верификатор доказательств)
- proof-object = proof-term (явный формальный вывод)

Словарь

- prover = theorem prover (система поиска выводов, прувер)
- automated theorem prover (система автоматического поиска выводов)
- proof-assistant = interactive theorem prover (система интерактивного поиска выводов)
- proof-checker (верификатор доказательств)
- proof-object = proof-term (явный формальный вывод)

Словарь

- prover = theorem prover (система поиска выводов, прувер)
- automated theorem prover (система автоматического поиска выводов)
- proof-assistant = interactive theorem prover (система интерактивного поиска выводов)
- proof-checker (верификатор доказательств)
- proof-object = proof-term (явный формальный вывод)

Словарь

- prover = theorem prover (система поиска выводов, прувер)
- automated theorem prover (система автоматического поиска выводов)
- proof-assistant = interactive theorem prover (система интерактивного поиска выводов)
- proof-checker (верификатор доказательств)
- proof-object = proof-term (явный формальный вывод)

Автоматический поиск выводов

- Действующие системы, как правило, основаны на вариантах исчисления предикатов первого порядка (метод резолюций, обратный метод и др.)
- Проводятся соревнования таких систем. Чемпион в последние годы: Vampire (А. Воронков, Манчестер)
- Современные системы способны найти доказательства, как правило, лишь самых простых результатов. Или же результатов в специализированных областях (теория групп, универсальная алгебра).

Автоматический поиск выводов

- Действующие системы, как правило, основаны на вариантах исчисления предикатов первого порядка (метод резолюций, обратный метод и др.)
- Проводятся соревнования таких систем. Чемпион в последние годы: Vampire (А. Воронков, Манчестер)
- Современные системы способны найти доказательства, как правило, лишь самых простых результатов. Или же результатов в специализированных областях (теория групп, универсальная алгебра).

Автоматический поиск выводов

- Действующие системы, как правило, основаны на вариантах исчисления предикатов первого порядка (метод резолюций, обратный метод и др.)
- Проводятся соревнования таких систем. Чемпион в последние годы: Vampire (А. Воронков, Манчестер)
- Современные системы способны найти доказательства, как правило, лишь самых простых результатов. Или же результатов в специализированных областях (теория групп, универсальная алгебра).

Настоящий успех 1996 г.

McCune, EQN: Доказательство открытой *гипотезы Роббинса* в универсальной алгебре.

Пусть на множестве A заданы ассоциативная бинарная операция $a, b \mapsto ab$ и одноместная операция $a \mapsto [a]$, тогда

$$(\forall a, b \in A\, [[ab][a[b]] = a) \;\Rightarrow\; (\exists c, d \in A\, [cd] = [c]).$$

Ответ: $c = x^3[x[x]]$ и $d = x[x[x]]$, где x любое.
Выкладка, устанавливающая $[cd] = [c]$: около 1/2 страницы.

Настоящий успех 1996 г.

McCune, EQN: Доказательство открытой *гипотезы Роббинса* в универсальной алгебре.

Пусть на множестве A *заданы ассоциативная бинарная операция* $a, b \mapsto ab$ *и одноместная операция* $a \mapsto [a]$, *тогда*

$$(\forall a, b \in A\, [[ab][a[b]] = a) \;\Rightarrow\; (\exists c, d \in A\, [cd] = [c]).$$

Ответ: $c = x^3[x[x]]$ и $d = x[x[x]]$, где x любое.
Выкладка, устанавливающая $[cd] = [c]$: около 1/2 страницы.

Интерактивные пруверы: «Семнадцать пруверов»

Freek Wiedijk (ed.) *The Seventeen Provers of the World*, Springer LNCS 3600, 2006.

Наиболее успешные:

HOL Light	UK	69 теорем из 100
Mizar	PL	45
ProofPower	UK	42
Isabelle	UK,DE	40
Coq	FR	39

Наиболее значительные формализации:

- Первая теорема Гёделя о неполноте (Shankar 1986, nqthm; O'Connor 2003, Coq)
- Теорема Жордана о кривых (Hales 2005, HOL Light; Korniłowicz 2005, Mizar)
- Закон распределения простых чисел: $\pi(x) \sim x/\ln(x)$ (Avigad et al. 2004, Isabelle; Harrison 2008, HOL Light).
- Доказательство гипотезы о четырёх красках (Gonthier et al. 2004, Coq)

Гипотеза о четырёх красках

Francis Guthrie, 1852: *Всякая простая карта на плоскости имеет правильную раскраску в четыре цвета.*

- Alfred Kempe, 1879; Peter Thomas, 1880: «решение»
- Обнаружена ошибка у Кемпе в 1890; у Томаса в 1891.
- K. Appel, W. Haken, 1976: решение с использованием компьютера.

  Доказательство сведено к перебору более 1476 различных графов и проверки для них некоторого условия на компьютере. 400 страниц + 1000 часов компьютерного времени.

- Сомнения в правильности решения, неточности в процедуре сведения (Ulrich Schmidt). Подробное изложение исправленного решения в книге: Appel/Haken, 1989.
- N. Robertson, D. Sanders, P. Seymour, R. Thomas, 1997: оптимизация решения Аппеля и Хакена. 633 случая, более эффективный (квадратичный) алгоритм проверки условия.
- Georges Gonthier, Benjamin Werner, 2005: формализация в Coq на основе доказательства Робертсона и др.

  Это включает как верификацию содержательного сведения, так и компьютерного перебора. Фактически, была написана верифицированная в Coq программа перебора. (И не нужно было вводить 633 случая от руки.)

Насколько надёжны компьютерные доказательства?

- Степень надёжности (к сожалению) зависит от прувера, его дизайна и внутренней архитектуры.
- Абсолютной надёжности (по целому ряду не зависящих друг от друга причин) не гарантирует ни один прувер.
- Несмотря на это, в целом компьютерные доказательства намного надёжнее всего остального.

Насколько надёжны компьютерные доказательства?

- Степень надёжности (к сожалению) зависит от прувера, его дизайна и внутренней архитектуры.
- Абсолютной надёжности (по целому ряду не зависящих друг от друга причин) не гарантирует ни один прувер.
- Несмотря на это, в целом компьютерные доказательства намного надёжнее всего остального.

Насколько надёжны компьютерные доказательства?

- Степень надёжности (к сожалению) зависит от прувера, его дизайна и внутренней архитектуры.
- Абсолютной надёжности (по целому ряду не зависящих друг от друга причин) не гарантирует ни один прувер.
- Несмотря на это, в целом компьютерные доказательства намного надёжнее всего остального.

«Принцип де Брёйна» (de Bruijn)

Идеальное техническое решение проблемы:

- В основе прувера лежит логическое ядро — формальная аксиоматическая система, в которой записываются выводы.
- логическое ядро должно быть обозримым (достаточно маленьким и простым).

  Например, PA и ZFC удовлетворяют этому условию. На практике в качестве ядер часто используются и более сложные теории.

- Прувер – который в принципе может быть сколь угодно сложной системой – в результате работы конструирует явный формальный вывод (proof-term) в языке своего ядра.
- Верификатор (независимо от прувера) проверяет корректность данного вывода на соответствие правилам ядра.
- Простота ядра гарантирует простоту верификатора. Более того, каждый желающий может сам написать свой собственный верификатор и убедиться в корректности каждого конкретного формального доказательства.

- Прувер – который в принципе может быть сколь угодно сложной системой – в результате работы конструирует явный формальный вывод (proof-term) в языке своего ядра.
- Верификатор (независимо от прувера) проверяет корректность данного вывода на соответствие правилам ядра.
- Простота ядра гарантирует простоту верификатора. Более того, каждый желающий может сам написать свой собственный верификатор и убедиться в корректности каждого конкретного формального доказательства.

- Прувер – который в принципе может быть сколь угодно сложной системой – в результате работы конструирует явный формальный вывод (proof-term) в языке своего ядра.
- Верификатор (независимо от прувера) проверяет корректность данного вывода на соответствие правилам ядра.
- Простота ядра гарантирует простоту верификатора. Более того, каждый желающий может сам написать свой собственный верификатор и убедиться в корректности каждого конкретного формального доказательства.