Санкт-Петербургский государственный университет
информационных технологий, механики и оптики
Кафедра Компьютерных Технологий

Илья Колыхматов, Михаил Смачных

Формальное доказательство утверждений с использованием HOL

Содержание

Санкт-Петербург, 2005

1. Введение

В данной работе приводится формальное доказательство утверждения о том, что если не существует треугольника с длинами сторон a, b, c, то и не существует треугольника с длинами сторон вида a^n, b^n, c^n $\forall n \in \mathbb{N}$, с использованием интерактивной системы HOL. Сначала более формально формулируется требуемое утверждение и приводится неформальное доказательство, а затем приводится полное доказательство на HOL как для натуральных, так и для вещественных чисел.

2. Описание задачи

Требуется доказать, что если не существует треугольника с длинами сторон a, b, c, то нет и треугольника со сторонами a^n, b^n, c^n, где $n \in \mathbb{N}$.

Сначала введем ограничения на область определения. Здравый смысл подсказывает, что утверждение нужно доказывать для треугольника с неотрицательными длинами сторон. В противном случае, например треугольник, для длин сторон которого выполнено равенство $a = b = c = -1$ не существует, но при $n = 2 \in \mathbb{N}$ имеем: $a^2 = b^2 = c^2 = 1$ — равносторонний треугольник.

Термы, теоремы и стратегии доказательства в HOL записываются на языке ML. В формальном доказательстве утверждения для натуральных чисел на HOL используются теоремы *arithmeticTheory*, а в случае вещественных чисел — теоремы *realTheory*.

3. Приступаем к решению задачи

Приведем "каркас" доказательства.

Сформулируем определение треугольника в виде:

$$triangle\ a\ b\ c = (a + b > c) \wedge (b + c > a) \wedge (c + a > b) \qquad (triangle)$$

Следующие теоремы: (1), (2), (3), (4) — будут использованы в дальнейшем для доказательства *"stumbling-block"* — утверждения, которое в свою очередь будет использовано для установления справедливости индукционного шага в *"kernel of the proof"*.

В (1) обе части неравенства $(a + b)^n \leq (a + b)^n$ домножаются на $(a + b)$, причем левая и правая части получившегося неравенства преобразованы по-разному. Утверждение доказывается с использованием коммутативности умножения, рефлексивности неравенства $a \leq a$, дистрибутивности, определения степени:

$$\forall a,\ b,\ n\colon (0 \leq a) \wedge (0 \leq b) \Rightarrow a \cdot (a + b)^n + b \cdot (a + b)^n \leq (a + b)^{n+1} \qquad (1)$$

От уменьшения левой части неравенства вида $a \leq a$ оно не перестанет быть верным:

$$\forall a,\ b,\ n\colon (0 \leq a) \wedge (0 \leq b) \Rightarrow a \cdot a^n \leq a \cdot (a+b)^n \tag{2}$$

Симметричный случай:

$$\forall a,\ b,\ n\colon (0 \leq a) \wedge (0 \leq b) \Rightarrow b \cdot b^n \leq b \cdot (a+b)^n \tag{3}$$

Если почленно сложить (2) и (3), то при тех же условиях, при которых выполнены (2) и (3), будет верно:

$$\forall a,\ b,\ n\colon (0 \leq a) \wedge (0 \leq b) \Rightarrow a^{n+1} + b^{n+1} \leq a \cdot (a+b)^n + b \cdot (a+b)^n \tag{4}$$

Важное утверждение (*"stumbling-block"*, *"камень преткновения"*) используется для доказательства индукционного шага в более общем утверждении *"kernel of the proof"*. Для доказательства этого утверждения достаточно последовательно посмотреть на содержание теорем (4) и (1) и применить свойство транзитивности для неравенств:

$$\forall a,\ b,\ n\colon (0 \leq a) \wedge (0 \leq b) \Rightarrow a^{n+1} + b^{n+1} \leq (a+b)^{n+1} \quad (stumbling - block)$$

Вспомогательное утверждение — если не выполняется одно неравенство, то выполняется другое:

$$\forall a,\ b,\ c\colon \sim(a+b > c) = (a+b \leq c) \qquad (inverting\ of\ inequality)$$

"Ядро" докательства исходного утверждения доказывается индукцией по n с использованием *stumbling-block* и вспомогательного утверждения *inverting of inequality*:

$$\forall a,\ b\colon (0 \leq a) \wedge (0 \leq b) \wedge \sim(a+b > c) \Rightarrow \\ \forall n \in \mathbb{N}\colon (n > 0) \Rightarrow \sim(a^n + b^n > c^n) \tag{5}$$

Исходное утверждение докажем от противного с использованием "ядра" доказательства (5):

$$\forall a,\ b,\ c\colon (0 \leq a) \wedge (0 \leq b) \wedge (0 \leq c) \wedge \sim(triangle\ a\ b\ c) \\ \Rightarrow \forall n \in \mathbb{N}\colon (n > 0 \Rightarrow \sim(triangle\ a^n\ b^n\ c^n)) \tag{6}$$

4. Система HOL

По адресу http://www.cl.cam.ac.uk/Research/HVG/FTP доступен дистрибутив системы HOL. Кроме того, для работы необходим Moscow ML[1] – это реализация функционального языка ML.

[1] http://www.dina.kvl.dk/~sestoft/mosml.html

4.1. Используемые тактики

В дальнейшем при доказательстве теорем будут использоваться различные тактики. Тактика **ARW_TAC** часто применяется с указанием списка теорем, которые в этом случае будут добавлены в качестве правил преобразования к стандартному множеству **arith_ss** правил упрощения и преобразования арифметических выражений. Отметим, что **ARW_TAC** – это введенное для удобства сокращение для тактики условного и контекстного переписывания **RW_TAC**:

val ARW_TAC = RW_TAC arith_ss;

Итак, при вызове тактики **ARW_TAC** система HOL пытается упростить целевое утверждение с использованием набора стандартных правил, а также некоторых дополнительных теорем и определений (если таковые были указаны пользователем). Очень часто данная тактика используется, когда какое-нибудь выражение нужно подробно расписать по определению, а затем упростить. Так, например, в ходе доказательства центральной теоремы *TRIANGLE_NOT_EXIST* на начальном этапе выполняется подстановка определения треугольника *triangle*.

В процессе доказательства справедливости утверждения для вещественных чисел будет активно использоваться аналогичная **ARW_TAC** тактика **S_TAC**, которая определена как сокращение для **RW_TAC** следующим образом:

val S_TAC = RW_TAC std_ss;

При вызове тактики **PROVE_TAC** система HOL пытается доказать целевое утверждение с использованием техники логики первого порядка. Поскольку система HOL содержит много различных теорем, а попытки их применения и вывода следствий из них приводят к огромному числу новых утверждений, очень важно точно указать вспомогательные леммы. Иначе процедура поиска доказательства будет работать очень долго и завершится переполнением стека. В отличие от тактик упрощения и переписывания, вызов тактики **PROVE_TAC** либо завершается успешным доказательством целевого утверждения, либо не дает вообще ничего.

Вызов **Cases_on** ‘a‘ заменяет текущее целевое утверждение на несколько новых утверждений, каждое из которых рассматривается отдельно (при этом все новые утверждения в совокупности эквивалентны исходному целевому утверждению). К примеру, если для a выполнено неравенство $a \geq 0$, то вызов **Cases_on** ‘a‘ приведет к двум утверждениям: в первом вместо a будет подставлен нуль, а во втором a будет строго больше нуля ($a > 0$). Такой разбор случаев имеет смысл, когда имеются теоремы для доказательства цели в предположении $a > 0$. При этом тривиальный случай $a = 0$ удобно рассмотреть отдельно.

Induct_on ‘n‘ применяется для установления справедливости теоремы по индукции. При этом вместо исходной цели будут образованы две новые подцели: база индукции (как правило, тривиальный случай) и индукцион-

ный переход.

4.2. Работа с интерактивной системой

Во время работы с интерактивной системой HOL для задания новой цели можно выполнить команду **g** с указанием утверждения:

```
g ‘!a c. (a <= c) ==> ((a * (c ** n)) <= (c ** SUC n))‘;
```

Затем может следовать последовательность команд для доказательства. К примеру, для упрощения выражения с использованием стандартной теоремы SUC_ONE_ADD можно выполнить команду вида

```
e (ARW_TAC[SUC_ONE_ADD]);
```

Когда процесс доказательства теоремы завершен, необходимо присвоить ей уникальное имя, чтобы впоследствии иметь возможность на нее ссылаться. Это можно сделать следующей командой:

```
val w41 = top_thm();
```

Теперь для удаления теоремы из вершины стека используется команда

```
drop();
```

При этом по мере надобности можно обращаться к доказанной теореме с помощью ассоциированного с ней уникального имени (в приведенном примере – это w41).

Ясно, что при наличии множества теорем будет полезен способ компактного хранения утверждений и последовательности действий для доказательства этих утверждений. В приводимом ниже формальном доказательстве для HOL употребляется краткая форма записи утверждения теоремы, а также ее доказательства, при помощи процедуры **store_thm**.

Отметим, что при работе с интерактивной системой HOL очень полезна функция **DB.match**, позволяющая осуществить поиск в огромном числе имеющихся теорем и найти нужную в процессе доказательства конкретной теоремы. Приведем пример использования этой функции:

```
DB.match ["arithmetic"] (Term‘SUC n * x‘);
```

5. Формальное доказательство для натуральных чисел

```
open arithmeticTheory;

val ARW_TAC = RW_TAC arith_ss;

(* Неравенство можно умножить на число в степени *)
val w41 = store_thm ("w41
```

```
‘‘!a c. (a <= c) ==> ((a * (c ** n)) <= (c ** SUC n))‘‘,
ARW_TAC[SUC_ONE_ADD]
THEN ARW_TAC[EXP_ADD, EXP_1]
THEN Cases_on ‘c ** n‘
THENL [ARW_TAC[],
       PROVE_TAC [MULT_LESS_EQ_SUC,
                  MULT_COMM]]);

(* Теорема о том, что можно домножить левую и правую часть неравенства
на число, переписанная в виде, удобном для дальнейшего использования *)
val w21 = store_thm ("w21
‘‘!a c. a ** n <= c ** n ==> (a ** SUC n) <= a * (c ** n)‘‘,
ARW_TAC[SUC_ONE_ADD]
THEN ARW_TAC[EXP_ADD, EXP_1]
THEN Cases_on ‘a‘
THENL [ARW_TAC[],
       PROVE_TAC[MULT_COMM,
                 MULT_LESS_EQ_SUC]]);

(* Левую и правую часть неравенства можно возводить в степень *)
val m1 = store_thm ("m1
‘‘!a c. a <= c ==> (a ** n) <= (c ** n)‘‘,
Induct_on ‘n‘
THENL [ARW_TAC[EXP],
       Cases_on ‘SUC n‘
       THENL[ARW_TAC[],
             PROVE_TAC[w21, w41, LESS_EQ_TRANS]]]);

(* Одна из ключевых теорем *)
val m2 = store_thm ("m2
‘‘!a b c. (a + b) <= c ==> ((a + b) ** n) <= (c ** n)‘‘,
ARW_TAC[m1]);

(* Вспомогательное утверждение о том, что можно раскрывать скобки *)
val DISCLOSING_OF_BRACKETS = store_thm ("DISCLOSING_OF_BRACKETS
‘‘!a b c d. ((a + b) * (c + d)) = (a * c + a * d + b * c + b * d)‘‘,
ARW_TAC[MULT_COMM, LEFT_ADD_DISTRIB]);

(* Следствие DISCLOSING_OF_BRACKETS для более узкого и нужного нам
в дальнейшем случая *)
val h2 = store_thm ("h2
‘‘!a b n. (n > 0) ==> (((a + b) * (a ** n + b ** n)) =
(a ** SUC n + a * (b ** n) + b * (a ** n) + b ** SUC n))‘‘,
ARW_TAC[DISCLOSING_OF_BRACKETS, MULT_COMM, EXP]);

(* Утверждение заключается в том, что можно взять некое исходное утверждение,
```

```
домножить его на (a + b), и чуть-чуть преобразовать *)
val INEQUALITY_CAN_BE_MULTIPLIED_BY_NUMBER =
store_thm ("INEQUALITY_CAN_BE_MULTIPLIED_BY_NUMBER
``!a b. (n > 0) ∧ (((a ** n + b ** n) <= ((a + b) ** n))) ==>
((a ** SUC n + a * (b ** n) + b * (a ** n) + b ** SUC n) <=
(((a + b) ** SUC n)))``,
ARW_TAC[GSYM h2]
THEN ARW_TAC[SUC_ONE_ADD]
THEN Cases_on `(a + b)`
THENL [ARW_TAC[],
       ARW_TAC[EXP_ADD, EXP_1]
       THEN PROVE_TAC[MULT_LESS_EQ_SUC,
                      EXP, MULT_COMM]]);

(* Лемма об уменьшении левой части неравенства вида a <= b *)
val g1 = store_thm ("g1
``!a b d e. (b > 0) ∧ (a + b + d <= e) ==> a + d <= e``,
ARW_TAC[]);

(* Ключевая теорема для случая со степенями *)
val REDUCE_THE_LEFT_PART_OF_AN_INEQUALITY =
store_thm ("REDUCE_THE_LEFT_PART_OF_AN_INEQUALITY
``!a b n. (n > 0) ∧ (a ** SUC n + a * (b ** n)
+ b * (a ** n) + b ** SUC n <= ((a + b) ** SUC n)) ==>
a ** SUC n + b ** SUC n <= ((a + b) ** SUC n)``,
ARW_TAC[g1]);

(* Камень преткновения *)
val STUMBLING_BLOCK = store_thm ("STUMBLING_BLOCK
``!a b. (n > 0) ==> (((a ** n + b ** n) <= ((a + b) ** n)))``,
Induct_on `n`
THENL [ARW_TAC[],
       Cases_on `n`
       THENL [ARW_TAC[ONE, EXP_1, EXP],
       ARW_TAC[INEQUALITY_CAN_BE_MULTIPLIED_BY_NUMBER,
                  REDUCE_THE_LEFT_PART_OF_AN_INEQUALITY]]]);

(* Ядро докательства исходного утверждения *)
val KERNEL_OF_THE_PROOF = store_thm ("KERNEL_OF_THE_PROOF
``!a b c n. (n > 0) ∧ (a + b <= c) ==>
((a ** n) + (b ** n)) <= (c ** n)``,
PROVE_TAC[STUMBLING_BLOCK,
              m2, LESS_EQ_TRANS]);

(* Треугольник существует, если выполнены ``неравенства треугольника'' *)
val triangle = Define `triangle a b c =
```

```
((a + b) > c) ∧ ((b + c) > a) ∧ ((c + a) > b)‘;

(* Исходное утверждение *)
val TRIANGLE_NOT_EXIST = store_thm ("TRIANGLE_NOT_EXIST
‘‘!a b c. ∼(triangle a b c)
==> (!n. (n > 0) ==> ∼(triangle (a**n) (b**n) (c**n)))‘‘,
ARW_TAC[triangle]
THENL [SPOSE_NOT_THEN STRIP_ASSUME_TAC
       THEN PROVE_TAC[KERNEL_OF_THE_PROOF, NOT_GREATER],
       SPOSE_NOT_THEN STRIP_ASSUME_TAC
       THEN PROVE_TAC[KERNEL_OF_THE_PROOF, NOT_GREATER],
       SPOSE_NOT_THEN STRIP_ASSUME_TAC
       THEN PROVE_TAC[KERNEL_OF_THE_PROOF, NOT_GREATER]]);
```

6. Формальное доказательство для вещественных чисел

```
load "realTheory";
open realTheory;

val S_TAC = RW_TAC std_ss;

(* Треугольник существует, если выполнены ‘‘неравенства треугольника’’ *)
val triangle = Define ‘triangle a b c = (((a:real) + (b:real)) > (c:real))
∧ (((b:real) + (c:real)) > (a:real)) ∧ (((c:real) + (a:real)) > (b:real))‘;

(* Обе части неравенства (a + b) pow n <= (a + b) pow n домножили на
(a + b) *)
val q1 = store_thm ("q1
‘‘!a b n. (0 <= a) ∧ (0 <= b) ==>
a * ((a + b) pow n) + b * ((a + b) pow n) <= (a + b) pow SUC n‘‘,
S_TAC[pow, REAL_MUL_SYM,
        REAL_LE_REFL, REAL_ADD_RDISTRIB]);

(* От уменьшения левой части неравенства вида a <= b оно не перестанет
быть верным *)
val q3 = store_thm ("q3
‘‘!a b n. (0 <= a) ∧ (0 <= b) ==> a * (a pow n) <= a * ((a + b) pow n)‘‘,
S_TAC[REAL_LE_ADD, POW_LE,
        REAL_LE_ADDR, REAL_LE_LMUL_IMP]);

(* От уменьшения левой части неравенства вида a <= b оно не перестанет
быть верным *)
```

```
val q4 = store_thm ("q4
``!a b n. (0 <= a) ∧ (0 <= b) ==> b * (b pow n) <= b * ((a + b) pow n)``,
S_TAC[REAL_LE_ADD, POW_LE,
        REAL_LE_ADDL, REAL_LE_LMUL_IMP]);

(* Общий случай *)
val q6 = store_thm ("q6
``!a b n. (0 <= a) ∧ (0 <= b) ==>
a pow SUC n + b pow SUC n <= a * ((a + b) pow n) + b * ((a + b) pow n)``,
S_TAC[pow, REAL_MUL_SYM]
THEN S_TAC[q3, q4, REAL_LE_ADD2]);

(* Камень преткновения: будет использован для установления справедливости
индукционного шага *)
val STUMBLING_BLOCK = store_thm ("STUMBLING_BLOCK
``!a b n. (0 <= a) ∧ (0 <= b) ==>
a pow SUC n + b pow SUC n <= (a + b) pow SUC n``,
PROVE_TAC[q6, q1, REAL_LE_TRANS]);

(* Если не выполняется одно неравенство, то выполняется другое *)
val INVERTING_OF_INEQUALITY = store_thm ("INVERTING_OF_INEQUALITY
``!a:real b:real c:real. ∼((a + b) > c) = ((a + b) <= c)``,
S_TAC[real_gt, REAL_NOT_LT]);

(* Ядро докательства исходного утверждения *)
val KERNEL_OF_THE_PROOF = store_thm ("KERNEL_OF_THE_PROOF
``!a b. (0 <= a) ∧ (0 <= b) ∧ ∼((a + b) > c)
==> !n:num. (n > 0) ==> ∼(a pow n + b pow n > c pow n)``,
SPOSE_NOT_THEN STRIP_ASSUME_TAC
THEN Induct_on `n`
THENL [S_TAC[REAL_LT_REFL],
       PROVE_TAC[STUMBLING_BLOCK, POW_LE,
                 REAL_LE_TRANS, REAL_LE_ADD,
                 INVERTING_OF_INEQUALITY]]);

(* Исходное утверждение *)
val TRIANGLE_NOT_EXIST = store_thm ("TRIANGLE_NOT_EXIST
``!a b c. (0 <= a) ∧ (0 <= b) ∧ (0 <= c) ∧ ∼(triangle a b c)
==> !n:num. (n > 0 ==> ∼(triangle (a pow n) (b pow n) (c pow n)))``,
S_TAC[triangle]
THEN SPOSE_NOT_THEN STRIP_ASSUME_TAC
THENL [PROVE_TAC[KERNEL_OF_THE_PROOF],
       PROVE_TAC[KERNEL_OF_THE_PROOF],
       PROVE_TAC[KERNEL_OF_THE_PROOF]]);
```

7. Заключение

Мощная интерактивная система HOL Kananaskis 1 может успешно использоваться для проведения формального доказательства утверждений из различных предметных областей. Система HOL содержит богатый инструментарий. В ходе построения рассмотренного доказательства применялись различные стратегии доказательства теорем.

Список литературы

[1] M. Gordon. HOL: A Machine Oriented Formulation of Higher-Order Logic. Technical Report No. 68, University of Cambridge Computer Laboratory, 1985.

[2] R. S. Boyer, J. S. Moore. Metafunctions: Proving Them Correct and Using Them Efficiently as New Proof Procedures. The Correctness Problem in Computer Science, Academic Press, New York, 1981.

[3] L. Paulson. A Higher-Order Implementation of Rewriting. Science of Computer Programming, volume 3, pages 119–149, 1983.