ИНФОРМАЦИОННОЕ ПРОСТРАНСТВО

Dan Christensen

КРАТКОЕ ВВЕДЕНИЕ В ПРОГРАММУ DC PROOF

ЧТО ТАКОЕ DC PROOF?

DC Proof – это математическая образовательная программа, обучающая логике и методам доказательства учеников средней школы разных уровней[1]. В программе реализована новая технология проверки доказательства, позволяющая ученикам использовать правила и аксиомы логики, теории множеств и теории чисел с помощью удобной системы всплывающих меню. Обратная связь с учеником действует постоянно. Невозможно ввести в доказательство даже одну неверную строку.

ЛОГИЧЕСКАЯ ЗАДАЧА

Для иллюстрации некоторых уникальных возможностей DC Proof, начнем с логической задачи[2]:

В некотором (выдуманном) городе Парадокс закон требует, чтобы все мужчины были чисто выбриты, для чего один из жителей города назначен парикмахером. В законе также утверждается, что каждый житель города будет выбрит парикмахером тогда и только тогда, когда он не бреется сам. К несчастью, как только человек назначается на должность парикмахера, он неизбежно становится нарушителем закона! Почему?

Парикмахер должен быть чисто выбрит, так как он сам житель этого города. Однако, побреется он или нет, он в любом случае будет нарушителем закона! Программа DC Proof может быть использована для создания доказательства, разрешающего этот очевидный парадокс. Прежде чем мы этим займемся, приведем некоторые философские рассуждения.

ФИЛОСОФИЯ

Логика и математика обеспечивают нас символическим языком для описания окружающего мира, а также средствами логического вывода. Существуют правила синтаксиса этого языка. Существуют также правила логического вывода, которые позволяют нам одни утверждения получать из других. Математическое доказательство – это одно или несколько утверждений. В начале – некоторые исходные условия или допущения, предполагаемые верными в каком-то смысле, на основе которых с помощью правил логического вывода производится ряд утверждений.

[1] Скачать текущую версию DC Proof можно по адресу http://www.dcproof.com

[2] Эта логическая задача основана на известном Парадоксе Парикмахера, приписываемому обычно британскому логику Бертрану Расселу.

ЯЗЫК ПРОГРАММЫ DC PROOF

Язык программы DC Proof близок, насколько это возможно, языку логики и математики, используемому в математических учебниках, при одном исключении – он использует в качестве алфавита символы стандартной компьютерной клавиатуры за исключением греческой буквы «ε», принятой для обозначения принадлежности множеству (печатается клавишей «@»). В языке есть три вида «слов»: высказывания, предикаты и переменные. Эти «слова» можно объединять или модифицировать с помощью 32 операторов, соединителей и знаков препинания, небольшую часть которых мы здесь увидим.

ЛОГИЧЕСКИЕ ВЫСКАЗЫВАНИЯ

Простейшие утверждения в DC Proof – это логические высказывания. Их можно рассматривать как элементарные факты или идеи, которые могут быть истинными или ложными. В языке они представлены в виде строк любой длины, начинающихся с заглавной буквы с последующими буквами, цифрами или апострофами. Вот примеры законных высказываний в DC Proof:

```
P A1'' Sunny Hot
```

В DC Proof мы не приписываем высказыванию значения истинно-ложно, как это делается во многих языках программирования. Чтобы указать, что высказывание «Sunny» истинно, мы просто пишем:

```
Sunny
```

NOT-ОПЕРАТОР (~)

Чтобы указать ложность высказывания «Sunny», мы пишем:

```
~Sunny
```

(~) символ – оператор отрицания. Высказывание читается: «не Sunny».

ОПЕРАТОР AND (&)

Два или более логических высказываний можно соединить, образовав новое высказывание. Например, чтобы указать, что высказывания «Sunny» и «Hot» оба верны, мы просто пишем выражение:

```
Sunny & Hot
```

где символ «&» – это оператор AND.

Чтобы указать, что «Sunny» верно, а «Hot» – нет, мы пишем:

```
Sunny & ~Hot
```

Табл. 1 содержит список логических операторов, используемых в DC Proof.

ЛОГИЧЕСКИЕ ПРЕДИКАТЫ

Под предикатом понимается функция от одной или более переменных, принимающая значения *истинно* или *ложно*.

Пример

```
Man(socrates)
```

Здесь имя предиката «Man», а имя переменной «socrates». Это можно понимать как высказывание «Сократ – человек». В DC Proof имена предикатов начинаются с заглавной буквы, за которой следует любое число букв, цифр и апострофов. Имена переменных начинаются со строчной буквы, за которой следует любое число букв, цифр и апострофов. Приведем примеры правильных имен переменных:

```
x y2' barber 12
```

В DC Proof не различаются переменные и константы. Например, «12» ничем не отличается от других переменных.

Табл. 1 Логические операторы: Исчисление высказываний

Оператор	Символ	Пример
NOT	**~**	**~P**
AND	**&**	**P & Q**
OR	**|**	**P | Q**
IMPLIES	**=>**	**P => Q**
IF-AND-ONLY-IF (IFF)	**<=>**	**P <=> Q**

Табл. 2

Выражение	Значение
`Man(x)`	x – житель этого города
`Man(barber)`	'barber' - житель этого города
`Shaves(x,y)`	x бреет y
`Shaves(x,x)`	x бреет x, или x бреется сам

УНИВЕРСАЛЬНЫЕ КВАНТОРЫ

Чтобы указать, что логический предикат или любое высказывание, содержащее переменную *x*, верны для любого значения *x*, нужно перед ними поставить универсальный квантор.

Пример

```
ALL(x):[Man(x) => Mortal(x)]
```

Здесь «ALL» – универсальный квантор, а «x» – кванторная переменная. Для кванторных переменных используется обычный шрифт черного цвета.

Это высказывание нужно понимать так: «Для всех *x*, если *x* – это человек, то *x* смертен». Или проще: «Все люди смертны».

КВАНТОРЫ СУЩЕСТВОВАНИЯ

Чтобы указать, что логический предикат или любое высказывание, содержащее переменную *x*, верны хотя бы для одного значения *x*, нужно перед ними поставить квантор существования.

Пример

```
EXIST(x): Man(x)
```

Здесь «EXIST» – квантор существования, а *x* – кванторная переменная. Это высказывание нужно понимать так: «существует *x*, являющийся человеком».

РЕШЕНИЕ ЗАДАЧИ О ПАРИКМАХЕРЕ [1]

Сначала мы должны описать ситуацию на языке DC Proof.

Нам понадобятся только два предиката: Man (с одной переменной) и Shaves (с двумя переменными) (см. табл. 2).

Напомним, что парикмахер – житель города и для любого жителя города парикмахер его бреет тогда и только тогда, когда он не бреется сам. Эту ситуацию можно описать так:

```
Man(barber) &
ALL(x):[Man(x) =>
[Shaves(barber,x) <=>
~Shaves(x,x)]]
```

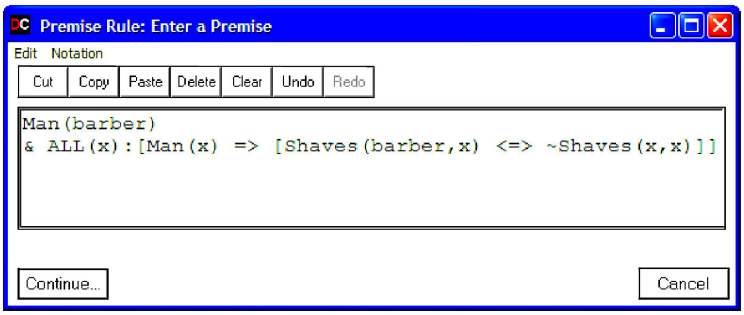

Рис. 1. Форма Premise Rule

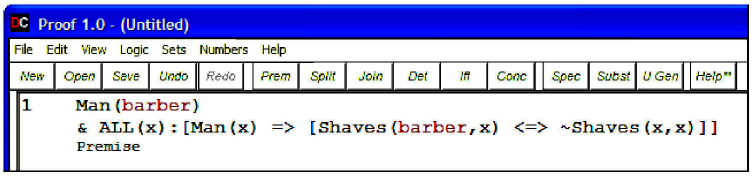

Рис. 2. Главное окно вывода

Здесь свободная переменная «barber» выделена красным цветом. Чтобы ввести это высказывание в качестве исходного пункта доказательства, мы щелкаем по кнопке «Prem» главного меню, открываем форму Premise Rule, в которую вводится указанное высказывание (см. рис. 1).

Далее щелчком по «Continue» мы получаем первую строку доказательства в главном окне вывода (см. рис. 2).

[1] Более подробное доказательство смотри в примере 7 руководства DC Proof.

Метка серого цвета показывает, каким образом была получена эта строка – с помощью Premise Rule в данном случае.

Расщепим оператор AND в строке 1, чтобы получить две новых строки доказательства:

```
2  Man(barber)
   Split, 1
3  ALL(x):[Man(x) => [Shaves(barber,x)
   <=> ~Shaves(x,x)]]
   Split, 1
```

Метка «Split, 1» в строках 2 and 3 указывает, что мы применили Split к высказыванию строки 1. Это сделано, чтобы можно было делать выводы отдельно для каждой части.

Далее применяем универсальный квантор строки 3 к самому парикмахеру. Для этого щелкаем по кнопке «Spec» главного меню и затем по строке 3 и получаем форму U Spec Rule (см. рис. 3).

Вводим «barber» и после «Continue» получаем четвертую строку доказательства:

```
4  Man(barber) => [Shaves(barber,barber)
   <=> ~Shaves(barber,barber)]
   U Spec, 3
```

Так как требуется, чтобы парикмахер был жителем города (Man(barber)), мы можем применить инструкцию Detachment(кнопка «Detach»), используя строки 4 и 2:

```
5  Shaves(barber,barber) <=> ~Shaves(barber,barber)
   Detach, 4, 2
```

DC Proof обнаруживает противоречие и, значит, исходное утверждение было ложным. Теперь применяем инструкцию Conclusion (кнопка «Conc») и получаем:

```
6  ~[Man(barber)
   & ALL(x):[Man(x) => [Shaves(barber,x)
   <=> ~Shaves(x,x)]]]
   4 Conclusion, 1
```

Заметим что «barber» изменил цвет с красного на зеленый. Это изменение указывает на возможность универсального обобщения относительно «barber.» Щелкаем по кнопке «U Gen» главного меню и затем по «barber» в строке 6. Получаем запрос имени (см. рис. 4).

Далее «Continue» и приходим к универсальному обобщению:

```
7  ALL(b):~[Man(b)
   & ALL(x):[Man(x) => [Shaves(b,x) <=> ~Shaves(x,x)]]]
   U Gen, 6
```

Заменим универсальный квантор на квантор существования, выбрав «Quantifier Switch» в меню «Logic» и щелкнув по этому квантору:

```
8  ~EXIST(b):~~[Man(b)
   & ALL(x):[Man(x) => [Shaves(b,x) <=> ~Shaves(x,x)]]]
   Quant, 7
```

Наконец, уберем двойное отрицание с помощью «Remove ~~» в меню «Logic», щелкнув по строке «~~» в строке 8:

```
9  ~EXIST(b):[Man(b)
   & ALL(x):[Man(x) => [Shaves(b,x) <=> ~Shaves(x,x)]]]
   Rem DNeg, 8
```

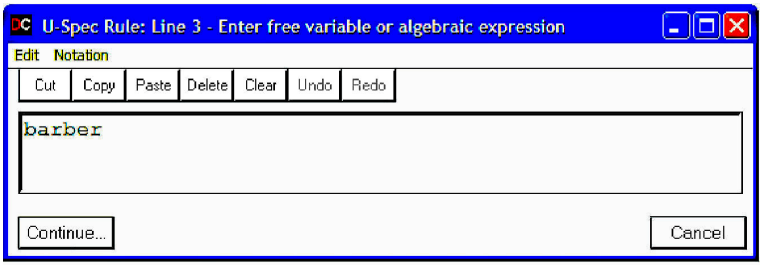

Рис. 3. Форма для ввода универсальной спецификации (U Spec) Rule

Используя доказательство от противного, мы увидели, что закон предъявляет невыполнимые требования к несчастному парикмахеру! Мы здесь показали,что не может быть человека, который сам является жителем города и при этом бреет всех тех и только тех жителей, которые не бреются сами. (Противоречия не было бы, если бы сам парикмахер не был бы жителем города или в законе было бы сделано исключение).

ЗАЧЕМ УЧИТЬ ФОРМАЛЬНОЙ ЛОГИКЕ И ТЕОРИИ МНОЖЕСТВ?

Было много дискуссий о наилучших способах преподавания методов доказательств младшим и старшим школьникам различных уровней подготовки. Традиционный подход основан на Евклидовой геометрии. Такой подход опирается на присущее ученикам с детства и развивающееся с годами геометрическое мышление. Однако исследования показали, что умение писать доказательства, приобретенное в одной ветви математики, например, геометрии, не переносится легко в такие области, как абстрактная алгебра или анализ. F.A. Ersoz утверждает, что многие неформальные «аксиомы» Евклидовой геометрии, преподаваемые обычным образом, основаны в большой степени на личной интуиции и воображении [1: с. 163]. Это позволяет использовать их как хорошую основу для обсуждения, однако при этом стирается граница между формальным и неформальным рассуждением и мешает пониманию того, что же является законным доказательством в различных областях математики. Ersoz утверждает также, что в геометрии мало используются многие методы доказательства, применяемые в более абстрактных областях – такие, как индукция, контрапозиции или приведение к противоречию [1: с. 164].

Почему же не преподавать формальную теорию геометрии? Во-первых, длинный список часто противоречащих интуиции аксиом, которые требуются даже для планиметрии, подавляет ученика. (См., например, работу Гильберта или Тарского на эту тему.) По моему личному опыту, доказательство простейшего геометрического утверждения разбухает в доказательство длиной в несколько сот строк.

Кажется разумным при первом знакомстве с методами доказательства использовать примеры из простейших областей, то есть из систем, содержащих *минимальное число правил и аксиом*. Такими областями являются логика и теория множеств. Подход, основанный на формальной логике и теории множеств, возможно, является наилучшим способом обучения методам доказательств и может быть широко применен в любой области математики.

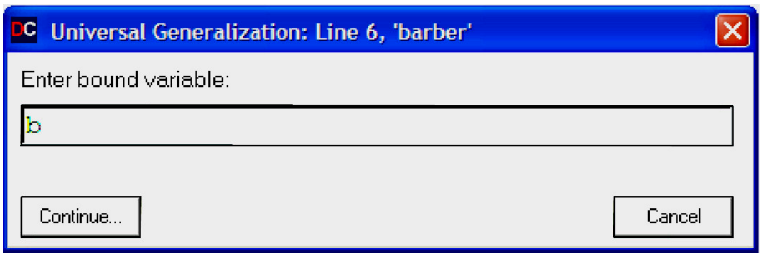

Рис. 4. Ввод имени переменной для инструкции U Gen

Хотя и не стоит представлять большую часть доказательств в формальном виде, нужно заметить, что представленные здесь правила и аксиомы логики и теории множеств понятны любому математику. На-

пример, каждый геометр должен понимать закон контрапозиции. Каждый специалист по теории чисел должен понимать законы De Morgan'a и т. д.

Включенное в DC Proof руководство содержит не только описание основных свойств программы, но также может служить вводным курсом для самообучения символьной логике и методам доказательства в университете, колледже или школе с углубленным изучением математики. Руководство включает ряд и упражнений с указаниями и полными решениями. Отходя от традиционного геометрического подхода, руководство использует примеры из логики, теории множеств и элементов теории чисел, общих для всех разделов математики.

Литература

1. *Ersoz F.A.* Proof in different mathematical domains // Proceedings of the ICMI Study 19 Conference, 2009. Vol. 1 / http://140.122.140.1/~icmi19/files/Volume_1.pdf

Дэн Кристенсен,
учитель средней школы из Канады на пенсии, имеет степень бакалавра по математике от университета Ватерлоо и степень бакалавра по педагогике от университета Торонто

Перевод с англ. М.И. Юдовина.