О верификации формализованных математических доказательств

Вторушин Ю.И.[1]

Рассматривается алгоритм верификации формализованных математических доказательств. Основная его часть формулируется в виде продукционной системы с метапеременными. Доказываются теоремы о его корректности и полноте.

Ключевые слова: автоматическое доказательство теорем, система автоматизации дедукции, верификация доказательств, язык первого порядка, исчисление предикатов, продукционная система, искусственный интеллект.

Введение

Автоматическое доказательство теорем является весьма важной областью современных исследований [1]. В этой области, кроме задачи поиска доказательств теорем, другой важной проблемой является верификация полученных каким-либо способом формализованных доказательств [2]. В данной статье рассматривается вторая из указанных проблем. Основные определения и обозначения понятий, связанных с автоматическим доказательством теорем, можно найти в статьях [3, 4]. В настоящей работе мы будем использовать принятые там понятия и обозначения. В статьях [3, 4] рассматривается алгоритм поиска натурального вывода. В общем случае такие алгоритмы имеют экспоненциальную сложность, которые либо заканчивают свою работу и выдают искомый результат, либо заканчивают работу и выдают слово No, либо не заканчивают работу. В отличие от алгоритма поиска, алгоритм верификации формализованного математического текста должен быть эффективным разрешающим алгоритмом. Такой алгоритм должен всегда заканчивать свою работу

[1] *Вторушин Юрий Игоревич* — фронтенд-разработчик в DSS Lab, e-mail: urchick@mail.ru.

Vtorushin Yurii Igorevich — front-end developer in DSS Lab.

и выдавать слово Yes, если текст является доказательством, и No — в противном случае.

В статье [3] рассмотрены системы натуральной дедукции $\mathfrak{D}_i$, где $i = 1, 2, 3$. При этом случай $i = 1$ соответствует случаю классической логики, $i = 2$ — интуиционистской, а $i = 3$ — минимальной. Через $\mathcal{D}_i$ будем обозначать систему натуральной дедукции, которая получается из $\mathfrak{D}_i$ в результате удаления аксиом и правил вывода, описывающих предикат равенства. В этой статье мы ограничимся рассмотрением только исчислений $\mathcal{D}_i$ для $i = 1, 2, 3$.

Во многих системах автоматизации дедукции (САД) пользователь имеет возможность записывать математический текст в виде, близком к тому, как текст выглядит в обычной математической статье. Пользователь может не дробить доказательство до уровня правил вывода системы $\mathcal{D}_i$. Такие алгоритмы верификации в соответствии с [5] мы будем называть *алгоритмами очевидности.*

В соответствии с синтаксисом языка САД обоснование формулы Φ в тексте формальной статьи может быть сделано двумя способами:

$$\text{“}\Phi \text{ by } \alpha_1, \ldots, \alpha_m\text{”} \quad \text{и} \quad \text{“}\Phi \text{ proof } \ldots \text{ qed”}.$$

В первом случае речь идет о непосредственном обосновании, а во втором — о более сложном доказательстве или локальном подвыводе. При этом метки $\alpha_1, \ldots, \alpha_m$ должны идентифицировать некоторые формулы $A_1, \ldots, A_m$ в тексте статьи, которые должны быть в области видимости формулы Φ. В статье [3] все обоснования должны соответствовать аксиомам и правилам вывода системы натуральной дедукции $\mathcal{D}_i$. Здесь мы описываем более общий алгоритм очевидности, соответствующий требованиям [5].

В настоящее время разработано много алгоритмов очевидности, которые лежат в основе существующих САД. В качестве примера можно привести широко известные системы Mizar [6] и SAD [7]. Проект SAD выполняется в рамках работ по реализации программы [5]. Верификатор системы SAD может использовать уже разработанные различные алгоритмы очевидности для классической логики. Также алгоритм верификатора системы Mizar описан в литературе и реализован только для классической логики.

По нашему мнению, недостаток используемых алгоритмов очевидности в существующих САД заключается в том, что в них непосредственное обоснование формулы Φ из формул $A_1, \ldots, A_n$ не тождественно

с тем, что обычно математик понимает под этим. Кроме того, используемые методы верификации для разных логик описываются на основе отличающихся концепций. В этой работе описываются единообразно для всех трех логических систем $\mathcal{D}_i$ соответствующие им алгоритмы очевидности $\mathcal{E}_i$, которые лишены отмеченных недостатков.

Алгоритм очевидности формулируется в разделе 1. При этом основная часть алгоритма формулируется в виде продукционной системы с метапеременными. Тем самым на теоретическом уровне алгоритм задается с точностью до стратегии построения дерева поиска типа И/ИЛИ в рамках этой продукционной системы. Теоремы о корректности и полноте рассматриваемого в статье алгоритма очевидности формулируются и доказываются соответственно в разделах 2 и 3.

1. Алгоритм верификации доказательств

В этом разделе для каждой из трех систем натуральной дедукции $\mathcal{D}_i$ определяется алгоритм очевидности $\mathcal{E}_i$. Для этого, прежде всего, описывается алгоритм верификации непосредственных обоснований "Φ by $\alpha_1, \ldots, \alpha_m$". Этот алгоритм формулируется в виде продукционной системы $\mathcal{V}_i$. Алгоритм очевидности $\mathcal{E}_i$ включает $\mathcal{V}_i$ как часть. Тем самым алгоритм $\mathcal{E}_i$ задается с точностью до стратегии построения дерева поиска типа И/ИЛИ в рамках $\mathcal{V}_i$.

При построении данной теории мы следуем [4]. Задачи продукционной системы формулируются с использованием псевдоформул и псевдотермов некоторого фиксированного языка Ω. При этом свободные предметные переменные псевдоформул считаются константами языка Ω. Мы не будем различать псевдоформулы, отличающиеся лишь переименованием связанных переменных. Будем всегда предполагать, что рассматриваемые псевдоформулы обладают свойством чистоты переменных [4]. Для унифицируемых псевдоформул A и B через $\mathbf{MGU}(A, B)$ ниже обозначается их наиболее общий унификатор.

Предложениями будем называть пары вида $\langle \alpha, \Phi \rangle$, где α — метка и Φ — псевдоформула.

После сделанных определений теперь точным образом опишем продукционные системы $\mathcal{V}_i$ для $i = 1, 2, 3$. Задачами этих систем являются упорядоченные пары вида $(\Gamma \rhd \Psi)$, где Γ есть конечное множество предложений и Ψ есть псевдоформула. Примитивными задачами для $\mathcal{V}_1$ и $\mathcal{V}_3$

являются задачи следующих двух видов:

$$\langle \alpha_1,\, \Phi_1 \rangle \,\ldots\, \langle \alpha_i,\, \Phi_i \rangle \,\ldots\, \langle \alpha_m,\, \Phi_m \rangle \,\rhd\, \Phi_i,$$

$$\langle \alpha_1,\, \Phi_1 \rangle \,\ldots\, \langle \alpha_m,\, \Phi_m \rangle \,\rhd\, \top.$$

Продукционная система $\mathcal{V}_2$, кроме указанных примитивных задач, включает также примитивные задачи следующего вида:

$$\langle \alpha_1,\, \Phi_1 \rangle \,\ldots\, \langle \alpha_i,\, \bot \rangle \,\ldots\, \langle \alpha_m,\, \Phi_m \rangle \,\rhd\, \Psi.$$

Будем говорить, что содержащая метапеременные $X_1,\, \ldots,\, X_k$ дедуктивная задача P *примитивизируема*, если существует подстановка $\theta = \begin{pmatrix} X_1 & \ldots & X_k \\ t_1 & \ldots & t_k \end{pmatrix}$ вместо метапеременных $X_1,\, \ldots,\, X_k$ такая, что дедуктивная задача $P\theta$ является примитивной.

Прежде всего, в нашем случае задачу

$$\langle \alpha_1,\, \Phi_1 \rangle \,\ldots\, \langle \alpha_i,\, \Phi_i \rangle \,\ldots\, \langle \alpha_m,\, \Phi_m \rangle \,\rhd\, \Psi$$

примитивизируют подстановки θ вида $\mathbf{MGU}(\Psi,\, \Phi_i)$ в случае, когда псевдоформулы Ψ и Φ_i унифицируемы. Кроме того, если $\Psi = \top$ или $\Phi_i = \bot$, то задачу примитивизирует любая подстановка, в частности пустая ε.

Формулируемые ниже правила 1 – 6 определяют общие для систем $\mathcal{V}_i$ правила декомпозиции.

1. Если $\Gamma \,\rhd\, \Psi_1$ и $\Gamma \,\rhd\, \Psi_2$ суть две разрешимые задачи, у которых существуют совместимые допустимые подстановки θ_1 и θ_2, то тогда задача $\Gamma \,\rhd\, \Psi_1 \,\&\, \Psi_2$ также разрешима и ее допустимой подстановкой считается комбинация θ_1 и θ_2. Символически сформулированное правило декомпозиции запишем в виде

$$\frac{\Gamma \,\rhd\, \Psi_1; \qquad \Gamma \,\rhd\, \Psi_2}{\Gamma \,\rhd\, \Psi_1 \,\&\, \Psi_2}.$$

Это удобная запись того, что задача вида $\Gamma \,\rhd\, \Psi_1 \,\&\, \Psi_2$ сводится к двум задачам $\Gamma \,\rhd\, \Psi_1$ и $\Gamma \,\rhd\, \Psi_2$.

2. Если для $j = 1$ или $j = 2$ разрешима задача $\Gamma \,\rhd\, \Psi_j$ и θ — ее допустимая подстановка, то задача $\Gamma \,\rhd\, \Psi_1 \vee \Psi_2$ также считается разрешимой и θ считается ее допустимой подстановкой:

$$\frac{\Gamma \,\rhd\, \Psi_j}{\Gamma \,\rhd\, \Psi_1 \vee \Psi_2}.$$

3. Если разрешима задача $\Gamma \rhd \Psi(X)$, где X — новая метапеременная в поисковом дереве, а θ — ее допустимая подстановка, то считается разрешимой задача $\Gamma \rhd \exists x\, \Psi(x)$ и θ считается ее допустимой подстановкой. То есть, задача $\Gamma \rhd \exists x\, \Psi(x)$ может быть сведена к дочерней задаче $\Gamma \rhd \Psi(X)$:

$$\frac{\Gamma \rhd \Psi(X)}{\Gamma \rhd \exists x\, \Psi(x)}.$$

4. Если разрешимы две задачи $\Gamma \rhd \Phi_1 \supset \Psi$ и $\Gamma \rhd \Phi_2 \supset \Psi$, которые имеют совместимые допустимые подстановки θ_1 и θ_2, то разрешима также задача $\langle \alpha,\, \Phi_1 \vee \Phi_2 \rangle\, \Gamma \rhd \Psi$ и ее допустимой подстановкой считается комбинация θ_1 и θ_2:

$$\frac{\Gamma \rhd \Phi_1 \supset \Psi; \qquad \Gamma \rhd \Phi_2 \supset \Psi}{\langle \alpha,\, \Phi_1 \vee \Phi_2 \rangle\, \Gamma \rhd \Psi}.$$

5. Если для $j = 1$ или $j = 2$ разрешима задача $\langle \alpha,\, \Phi_j \rangle\, \Gamma \rhd \Psi$ и θ — ее допустимая подстановка, то задача $\langle \alpha,\, \Phi_1 \,\&\, \Phi_2 \rangle\, \Gamma \rhd \Psi$ также считается разрешимой и θ считается ее допустимой подстановкой:

$$\frac{\langle \alpha,\, \Phi_j \rangle\, \Gamma \rhd \Psi}{\langle \alpha,\, \Phi_1 \,\&\, \Phi_2 \rangle\, \Gamma \rhd \Psi}.$$

6. Если разрешимы две задачи $\Gamma \rhd \Phi_1$ и $\langle \alpha,\, \Phi_2 \rangle\, \Gamma \rhd \Psi$, у которых существуют совместимые допустимые подстановки θ_1 и θ_2, то разрешима также задача $\langle \alpha,\, \Phi_1 \supset \Phi_2 \rangle\, \Gamma \rhd \Psi$ и ее допустимой подстановкой считается комбинация θ_1 и θ_2:

$$\frac{\Gamma \rhd \Phi_1; \qquad \langle \alpha,\, \Phi_2 \rangle\, \Gamma \rhd \Psi}{\langle \alpha,\, \Phi_1 \supset \Phi_2 \rangle\, \Gamma \rhd \Psi}.$$

7. Если разрешима задача $\langle \alpha,\, \Phi(X) \rangle\, \Gamma \rhd \Psi$, где X — новая метапеременная в поисковом дереве, а θ — ее допустимая подстановка, то считается разрешимой задача $\langle \alpha,\, \forall x\, \Phi(x) \rangle\, \Gamma \rhd \Psi$ и θ считается ее допустимой подстановкой:

$$\frac{\langle \alpha,\, \Phi(X) \rangle\, \Gamma \rhd \Psi}{\langle \alpha,\, \forall x\, \Phi(x) \rangle\, \Gamma \rhd \Psi}.$$

Формулировка продукционных систем $\mathcal{V}_i$ завершена. Каждая продукционная система $\mathcal{V}_i$ задает алгоритм поиска решения дедуктивной задачи не уточняя стратегию построения дерева поиска типа И/ИЛИ. Зафиксировав конкретную стратегию мы получим некоторую реализацию так

сформулированного алгоритма. Стратегия построения дерева поиска заключается в задании способа выбора листа для формирования у него, во-первых, множества всех примитивизирующих этот лист подстановок, во-вторых, всех дочерних связок вершин, соответствующих правилам декомпозиции. Затем каждая примитивизирующая подстановка "поднимается" по соединяющей лист с корнем ветви с помощью операции комбинации [8]. Тем самым в ходе процесса построения дерева поиска у его вершин формируются наборы допустимых подстановок. В качестве символических названий сформулированных правил декомпозиции 1 – 7 будем соответственно использовать следующие обозначения: $(\triangleright\&)$, $(\triangleright\vee)$, $(\triangleright\exists)$, $(\vee\triangleright)$, $(\&\triangleright)$, $(\supset\triangleright)$, $(\forall\triangleright)$.

Если корень дерева поиска имеет хотя бы одну допустимую подстановку θ, то *соответствующее* этой подстановке *деривационное поддерево* Θ вычисляется согласно следующей процедуре. Прежде всего, включаем в Θ исходную задачу P. В случае, когда подстановка θ примитивизирует задачу P, поддерево Θ содержит вершину P в качестве листа. В противном случае мы перебираем у родительской вершины P все дочерние связки вершин. Пусть r — количество таких связок и $P_1^j,\ \ldots,\ P_{q_j}^j$ — одна из них, где $j = 1, \ldots, r$. Рассмотрим всевозможные комбинации подстановок $\theta_1,\ \ldots,\ \theta_{q_j}$, где каждая θ_s берется из набора допустимых подстановок задачи P_s^j. Среди связок дочерних вершин задачи P обязательно найдется хотя бы одна такая, что указанная комбинация совпадет с θ. Такую связку $P_1^j, \ldots, P_{q_j}^j$ добавим в текущее поддерево Θ. Затем этот процесс приме́ним для всех вершин $P_1^j,\ \ldots,\ P_{q_j}^j$ и соответствующих подстановок $\theta_1,\ \ldots,\ \theta_{q_j}$. Последнее из полученных поддеревьев и есть искомое деривационное поддерево Θ.

Теорема. *Алгоритм проверки непосредственных обоснований всегда завершает свою работу.*

Доказательство. Пусть n есть общее число вхождений всех подформул в исходную задачу

$$\langle\alpha_1,\, \Phi_1\rangle\ \ldots\ \langle\alpha_m,\, \Phi_m\rangle\ \triangleright\ \Psi. \tag{1}$$

Введем обозначение $h = 2 \cdot (m + 1)$. Чтобы установить применимость алгоритма к любой задаче (1) покажем, что дерево поиска конечно и содержит не более, чем $p_n(h) = 1 + h + h^2 + \ldots + h^n$ вершин.

Действительно, применение правил декомпозиции $(\triangleright\&)$, $(\triangleright\vee)$, $(\triangleright\exists)$, $(\vee\triangleright)$, $(\&\triangleright)$, $(\supset\triangleright)$, $(\forall\triangleright)$ всегда приводит к тому, что значение параметра n у каждой дочерней задачи становится меньше значения этого

параметра у родительской задачи. Отсюда следует, что, во-первых, высота дерева поиска T задачи (1) не превосходит n; во-вторых, у каждой родительской вершины может быть не более, чем h дочерних вершин. В то же время конечное дерево высоты n, у которого все отличные от листьев вершины имеют ровно h дочерних вершин, содержит $p_n(h)$ вершин. Таким образом, дерево поиска T задачи (1) конечно и содержит не более, чем $p_n(h)$ вершин. □

Сформулированный в [3] для $\mathcal{D}_i$ алгоритм очевидности теперь распространим на более сложный вариант непосредственных обоснований "Φ by $\alpha_1, \ldots, \alpha_m$", который верифицируется продукционной системой $\mathcal{V}_i$. Получаемый таким способом алгоритм очевидности будем обозначать $\mathcal{E}_i$. Новый алгоритм $\mathcal{E}_i$ теперь зависит от выбора стратегии построения дерева поиска типа И/ИЛИ в рамках $\mathcal{V}_i$.

В соответствии с [3] новый алгоритм очевидности $\mathcal{E}_i$ обрабатывает исходный текст следующим образом. Алгоритм $\mathcal{E}_i$ последовательно проверяет, во-первых, правильность непосредственных умозаключений, которые соответствует обоснованиям вида "Φ by $\alpha_1, \ldots, \alpha_m$"; во-вторых, правильность структуры доказательств, которые соответствуют обоснованиям вида "Φ proof ... qed". В последнем случае проверяется соответствие тезиса доказательства доказываемой формуле.

Таким образом, изменения касаются только процедуры обработки непосредственных обоснований "Φ by $\alpha_1, \ldots, \alpha_m$". Процедура обработки обоснований вида "Φ proof ... qed" не меняется. А именно, рассуждение "proof ... qed", кроме предложений и их обоснований вида "$\alpha : \Psi$ ...", может содержать также специфические фразы четырех видов: "let $\xi_1, \ldots \xi_k$", "consider $\xi_1, \ldots \xi_k$ such that $\alpha : \Psi$", "assume $\alpha : \Psi$" и "thus Ψ", где $\xi_1, \ldots \xi_k$ суть фиксируемые предметные переменные и Ψ есть формула. Алгоритм $\mathcal{E}_i$ составляет список всех допущений и заключений $\Psi_1, \Psi_2, \ldots, \Psi_n, \Psi_{n+1}$ рассуждения "proof ... qed", соответствующий фразам "assume $\alpha : \Psi_j$" и "thus Ψ_j". Затем $\mathcal{E}_i$ сопоставляет с формулой Φ прямой тезис

$$\forall\bar{\xi}_1(\Psi_1\,\lambda_1\,\forall\bar{\xi}_2(\Psi_2\,\lambda_2\,(\ldots\forall\bar{\xi}_n(\Psi_n\,\lambda_n\,\forall\bar{\xi}_{n+1}\Psi_{n+1}))\ldots)$$

рассуждения "proof ... qed", где $\bar{\xi}_j$ — список всех переменных из фраз "let ...", находящихся между пунктами Ψ_{j-1} и Ψ_j, а для связки λ_j имеет место

$$\lambda_j = \begin{cases} \&, & \text{если } \Psi_j \text{ — заключение,} \\ \supset, & \text{если } \Psi_j \text{ — допущение.} \end{cases}$$

Кроме того, для доказательств в рамках классической логики $\mathcal{D}_1$ в случае, когда последним заключением Ψ_{n+1} рассуждения "proof ... qed" является противоречие $\bot$, а также Ψ_n имеет вид $\neg\,\Theta$ и λ_n есть $\supset$, помимо прямого тезиза, рассматривается для сравнения с Φ также косвенный тезис

$$\forall\bar{\xi}_1(\Psi_1\,\lambda_1\,\forall\bar{\xi}_2(\Psi_2\,\lambda_2\,(\ldots\forall\bar{\xi}_{n-1}(\Psi_{n-1}\,\lambda_{n-1}\,\forall\bar{\xi}_n\Theta))\ldots).$$

Очевидным следствием теоремы о конечности поисковых деревьев у продукционной системы $\mathcal{V}_i$ и так сформулированного алгоритма очевидности $\mathcal{E}_i$ является следующее утверждение.

Следствие. *Алгоритм очевидности $\mathcal{E}_i$ всегда завершает свою работу.*

Замечание. Будем называть пару $\langle m,\, n\rangle$ *сложностью* дедуктивной задачи (1). С практической точки зрения можно считать, что параметр n ограничен некоторым достаточно большим натуральным значением N. Что фактически наблюдается в реальных математических статьях. Тогда, при проверке непосредственных обоснований, число вершин в дереве поиска не превосходит $p_N(h)$, а это значит, что алгоритм очевидности $\mathcal{E}_i$ можно считать полиномиальным.

В качестве примера рассмотрим формальное доказательство теоремы о том, что множество является пустым в том и только в том случае, когда оно является подмножеством любого множества. Подробное доказательство в рамках системы натуральной дедукции $\mathcal{D}_1$ выглядит следующим образом:

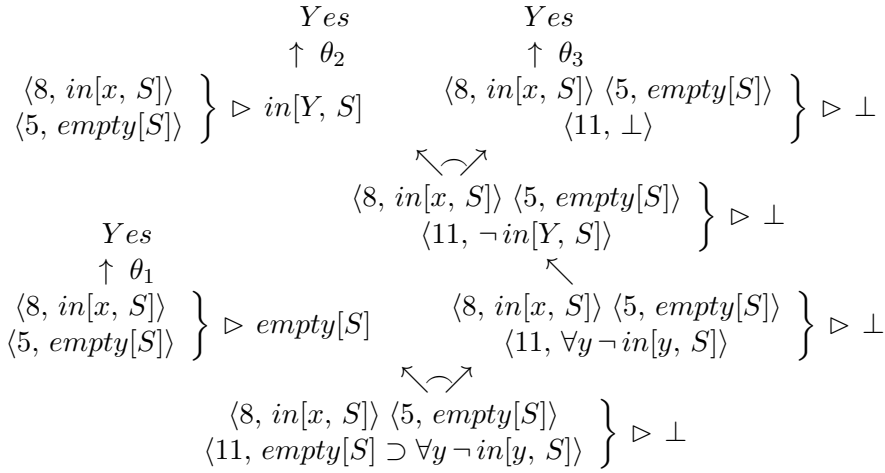
```
environ
  type Set;
  type Element;
  pred in[Element, Set];
  pred subset[Set, Set];
  pred empty[Set];
  reserve A, S, T for Set;
  reserve x, y, z for Element;
  1: for S,A holds subset[A,S] iff for x st in[x,A] holds in[x,S];
  2: for S holds empty[S] iff for x holds not in[x,S];
  3: ex S st empty[S];
text

  for S holds (for T holds subset[S,T]) iff empty[S]
proof
  let S be Set;
  4: (for T holds subset[S,T]) implies empty[S]
  proof
```

```
    assume 4: for T holds subset[S,T];
    5: empty[S] iff (for x holds not in[x,S]) by 2;
    6: (for x holds not in[x,S]) implies empty[S] by 5;
    7: for x holds not in[x,S]
    proof
      let x be Element;
      assume 7: in[x,S];
      consider U be Set such that 8: empty[U] by 3;
      9: empty[U] iff (for y holds not in[y,U]) by 2;
      10: empty[U] implies (for y holds not in[y,U]) by 9;
      11: for y holds not in[y,U] by 10,8;
      12: not in[x,U] by 11;
      13: subset[S,U] iff (for x holds in[x,S] implies in[x,U]) by 1;
      14: subset[S,U] implies (for z holds in[z,S] implies in[z,U]) by 13;
      15: subset[S,U] by 4;
      16: for z holds in[z,S] implies in[z,U] by 14,15;
      17: in[x,S] implies in[x,U] by 16;
      18: in[x,U] by 17,7;
      thus false by 12,18;
    qed;
    thus empty[S] by 6,7;
  qed;
  thus (for T holds subset[S,T]) implies empty[S] by 4;
  assume 5: empty[S];
  let V be Set;
  6: subset[S,V] iff (for x holds in[x,S] implies in[x,V]) by 1;
  7: (for x holds in[x,S] implies in[x,V]) implies subset[S,V] by 6;
  8: for x holds in[x,S] implies in[x,V]
  proof
    let x be Element;
    assume 8: in[x,S];
    9: in[x,V]
    proof
      assume 9: not in[x,V];
      10: empty[S] iff (for x holds not in[x,S]) by 2;
      11: empty[S] implies (for y holds not in[y,S]) by 10;
      12: for y holds not in[y,S] by 11,5;
      13: not in[x,S] by 12;
      thus false by 13,8;
    qed;
    thus in[x,V] by 9;
  qed;
  thus subset[S,V] by 7,8;
qed;
```

Более короткое “человеческое” доказательство, которое успешно верифицируется алгоритмом очевидности $\mathcal{E}_1$, имеет следующий вид:

```
  for S holds (for T holds subset[S,T]) iff empty[S]
proof
  let S be Set;
  thus (for T holds subset[S,T]) implies empty[S]
  proof
    assume 4: for T holds subset[S,T];
    7: for x holds not in[x,S]
    proof
      let x be Element such that 7: in[x,S];
      consider U be Set such that 8: empty[U] by 3;
      11: for y holds not in[y,U] by 2,8;
      12: not in[x,U] by 11;
      14: subset[S,U] implies (for z holds in[z,S] implies in[z,U]) by 1;
      15: subset[S,U] by 4;
      thus contradiction by 12,14,15,7;
    qed;
    thus empty[S] by 2,7;
  qed;
  assume 5: empty[S];
  let V be Set;
  8: for x holds in[x,S] implies in[x,V]
  proof
    let x be Element such that 8: in[x,S];
    thus in[x,V]
    proof
      assume 9: not in[x,V];
      11: empty[S] implies (for y holds not in[y,S]) by 2;
      thus contradiction by 11,5,8;
    qed;
  qed;
  thus subset[S,V] by 1,8;
qed;
```

В этом доказательстве рассмотрим более подробно непосредственное обоснование "$\perp$ by 11,5,8". Алгоритм очевидности $\mathcal{E}_1$ сформирует дерево поиска типа И/ИЛИ, в котором содержится соответствующее допустимой подстановке $\theta = \begin{pmatrix} Y \\ x \end{pmatrix}$ следующее деривационное поддерево:



Здесь три листа этого дерева примитивизируются подстановками $\theta_1 = \varepsilon$, $\theta_2 = \begin{pmatrix} Y \\ x \end{pmatrix}$ и $\theta_3 = \varepsilon$. Комбинация совместимых подстановок θ_1, θ_2 и θ_3 дает допустимую подстановку θ для корня дерева поиска.

2. Теорема о корректности

Теорема о корректности устанавливает допустимость в $\mathcal{D}_i$ верифицируемых в продукционной системе $\mathcal{V}_i$ правил вывода "Φ by $\alpha_1, \ldots, \alpha_m$". Тем самым мы получаем метод, который позволяет правильный с точки зрения алгоритма очевидности $\mathcal{E}_i$ текст доказательства "раздробить" до уровня применений правил вывода системы натуральной дедукции $\mathcal{D}_i$.

Теорема о корректности. *Если задача* $\langle \alpha_1, \Phi_1 \rangle \ldots \langle \alpha_m, \Phi_m \rangle \rhd \Psi$ *разрешима в* $\mathcal{V}_i$*, то существует цепочка непосредственных обоснований, которая устанавливает выводимость формулы* Ψ *из формул* $\Phi_1, \ldots, \Phi_m$ *в* $\mathcal{D}_i$.

Доказательство. Пусть $\theta = \begin{pmatrix} X_1 & \ldots & X_r \\ t_1 & \ldots & t_r \end{pmatrix}$ есть подстановка из набора допустимых подстановок задачи $P = (\langle \alpha_1, \Phi_1 \rangle \ldots \langle \alpha_m, \Phi_m \rangle \rhd \Psi)$. Мы предполагаем, что подстановка θ является основной. В противном случае мы можем заменить "нерешенные" метапеременные на локальные константы. Пусть Θ является соответствующим этой подстановке

θ деривационным деревом. Будем обозначать штрихом следующее соответствие: $\Phi' = \Phi\theta$. Индукцией по высоте дерева Θ покажем, как в рамках $\mathcal{D}_i$ построить цепочку непосредственных обоснований, которая устанавливает выводимость формулы Ψ' из формул $\Phi'_1, \ldots, \Phi'_m$ в $\mathcal{D}_i$. Рассмотрим все возможные варианты получения в рамках $\mathcal{V}_i$ дерева Θ с корнем P.

1. Пусть P примитивизируется подстановкой θ. Тогда либо $\Psi = \top$, либо $\Psi' = \Phi'_i$, либо $\Phi_i = \bot$. В первых двух случаях для всех $i = 1, 2, 3$, а в третьем случае для $i = 2$, считается, что в $\mathcal{D}_i$ формула Ψ' является непосредственным следствием формул $\Phi'_1, \ldots, \Phi'_m$.

2. Пусть $P = (\Gamma \rhd \Psi_1 \,\&\, \Psi_2)$ и получается по правилу $(\rhd\&)$. Пусть дочерние задачи $(\Gamma \rhd \Psi_1)$ и $(\Gamma \rhd \Psi_2)$ являются корнями соответственно поддеревьев Θ_1 и Θ_2. Пусть по индуктивному предположению деривационным поддеревьям Θ_1 и Θ_2 сопоставляются соответственно цепочки непосредственных обоснований

$$\Delta_1\,(\beta_1 : \Psi'_1 \text{ by } \gamma_1, \ldots, \gamma_k) \;\text{ и }\; \Delta_2\,(\beta_2 : \Psi'_2 \text{ by } \delta_1, \ldots, \delta_l).$$

Указанные цепочки устанавливают соответственно выводимость Ψ'_1 и Ψ'_2 из $\Phi'_1, \ldots, \Phi'_m$ в $\mathcal{D}_i$. Тогда дереву Θ мы можем сопоставить цепочку непосредственных обоснований

$$\begin{array}{l}\Delta_1\,(\beta_1 : \Psi'_1 \text{ by } \gamma_1, \ldots, \gamma_k)\\ \Delta_2\,(\beta_2 : \Psi'_2 \text{ by } \delta_1, \ldots, \delta_l)\\ (\Psi'_1 \,\&\, \Psi'_2 \text{ by } \beta_1, \beta_2).\end{array}$$

Полученная цепочка устанавливает выводимость Ψ' из $\Phi'_1, \ldots, \Phi'_m$ в $\mathcal{D}_i$.

3. Пусть $P = (\Gamma \rhd \Psi_1 \vee \Psi_2)$ получается по правилу $(\rhd\vee)$. Пусть для $j = 1$ или 2 по индуктивному предположению деривационному поддереву Θ_j сопоставляется цепочка непосредственных обоснований

$$\Delta\,(\beta : \Psi'_j \text{ by } \gamma_1, \ldots, \gamma_k).$$

Тогда дереву Θ сопоставим цепочку непосредственных обоснований

$$\Delta\,(\beta : \Psi'_j \text{ by } \gamma_1, \ldots, \gamma_k)\,(\Psi'_1 \vee \Psi'_2 \text{ by } \beta).$$

4. Пусть $P = (\Gamma \rhd \exists x\, \Psi_1(x))$ получается по правилу $(\rhd\exists)$. Тогда имеем $\Psi_1(X) = \Psi_1(X_j)$ и $\Psi_1(X)\theta = \Psi'_1(t_j)$ для некоторого $j \in \{1, \ldots, r\}$. Пусть по индуктивному предположению соответствующему задаче $(\Gamma \rhd \Psi_1(X))$ деривационному поддереву в Θ сопоставляется цепочка непосредственных обоснований

$$\Delta\,(\beta : \Psi'_1(t_j) \text{ by } \gamma_1, \ldots, \gamma_k).$$

В таком случае дереву Θ мы можем сопоставить цепочку непосредственных обоснований

$$\Delta\,(\beta : \Psi'_1(t_j)\ \text{by}\ \gamma_1, \ldots, \gamma_k)\,(\exists x\,\Psi'_1(x)\ \text{by}\ \beta).$$

5. Пусть $P = (\langle\alpha_i,\ \Upsilon_1 \vee \Upsilon_2\rangle\,\Pi \rhd \Psi)$ получается по правилу $(\vee\rhd)$. Пусть дочерние задачи $(\Pi \rhd \Upsilon_1 \supset \Psi)$ и $(\Pi \rhd \Upsilon_2 \supset \Psi)$ являются корнями соответственно поддеревьев Θ_1 и Θ_2. Пусть по индуктивному предположению деривационным поддеревьям Θ_1 и Θ_2 сопоставляются соответственно цепочки непосредственных обоснований

$$\Delta_1\,(\Upsilon'_1 \supset \Psi'\ \text{by}\ \gamma_1, \ldots, \gamma_k)\ \ \text{и}\ \ \Delta_2\,(\Upsilon'_2 \supset \Psi'\ \text{by}\ \delta_1, \ldots, \delta_l).$$

Тогда дереву Θ сопоставим цепочку непосредственных обоснований

$$\begin{array}{l}(\alpha_i : \Upsilon'_1 \vee \Upsilon'_2)\\ \Delta_1\,(\beta_1 : \Upsilon'_1 \supset \Psi'\ \text{by}\ \gamma_1, \ldots, \gamma_k)\\ \Delta_2\,(\beta_2 : \Upsilon'_2 \supset \Psi'\ \text{by}\ \delta_1, \ldots, \delta_l)\\ (\Psi'\ \text{by}\ \alpha_i,\ \beta_1,\ \beta_2).\end{array}$$

6. Пусть $P = (\langle\alpha_i,\ \Upsilon_1\,\&\,\Upsilon_2\rangle\,\Pi \rhd \Psi)$ получается по правилу $(\&\rhd)$. Пусть для $j = 1$ или 2 по индуктивному предположению деривационному поддереву Θ_j сопоставляется цепочка непосредственных обоснований

$$(\beta : \Upsilon'_j)\,\Delta\,(\Psi'\ \text{by}\ \gamma_1, \ldots, \gamma_k).$$

Тогда дереву Θ сопоставим цепочку непосредственных обоснований

$$(\alpha_i : \Upsilon'_1\,\&\,\Upsilon'_2)\,(\beta : \Upsilon'_j\ \text{by}\ \alpha_i)\,\Delta\,(\Psi'\ \text{by}\ \gamma_1, \ldots, \gamma_k).$$

7. Пусть $P = (\langle\alpha_i,\ \Upsilon_1 \supset \Upsilon_2\rangle\,\Pi \rhd \Psi)$ получается по правилу $(\supset \rhd)$. Пусть дочерние задачи $(\Pi \rhd \Upsilon_1)$ и $(\langle\alpha_i,\ \Upsilon_2\rangle\,\Pi \rhd \Psi)$ являются корнями соответственно поддеревьев Θ_1 и Θ_2. Пусть по индуктивному предположению деривационным поддеревьям Θ_1 и Θ_2 сопоставляются соответственно цепочки непосредственных обоснований

$$\Delta_1\,(\Upsilon'_1\ \text{by}\ \gamma_1, \ldots, \gamma_k),$$

$$(\beta_2 : \Upsilon'_2)\,\Delta_2\,(\Psi'\ \text{by}\ \delta_1, \ldots, \delta_l).$$

Тогда дереву Θ сопоставим цепочку непосредственных обоснований

$$\begin{array}{l}(\alpha_i : \Upsilon'_1 \supset \Upsilon'_2)\\ \Delta_1\,(\beta_1 : \Upsilon'_1\ \text{by}\ \gamma_1, \ldots, \gamma_k)\\ (\beta_2 : \Upsilon'_2\ \text{by}\ \alpha_i,\ \beta_1)\,\Delta_2\\ (\Psi'\ \text{by}\ \delta_1, \ldots, \delta_l).\end{array}$$

8. Пусть $P = (\langle \alpha_i,\, \forall x\, \Upsilon(x) \rangle\, \Pi \rhd \Psi)$ получается по правилу $(\forall\rhd)$. Тогда имеем $\Upsilon(X) = \Upsilon(X_j)$ и $\Upsilon(X)\theta = \Upsilon'(t_j)$ для некоторого $j \in \{1, \ldots, r\}$. Пусть по индуктивному предположению деривационному поддереву в Θ, соответствующему задаче $(\langle \beta,\, \Upsilon(X) \rangle\, \Pi \rhd \Psi)$, сопоставляется цепочка непосредственных обоснований

$$(\beta : \Upsilon'(t_j))\, \Delta\, (\Psi' \text{ by } \gamma_1, \ldots, \gamma_k).$$

В этом случае дереву Θ мы можем сопоставить цепочку непосредственных обоснований

$$(\alpha_i : \forall x\, \Upsilon'(x))\, (\beta : \Upsilon'(t_j) \text{ by } \alpha_i)\, \Delta\, (\Psi' \text{ by } \gamma_1, \ldots, \gamma_k).$$

□

Следствие. *Если алгоритм очевидности $\mathcal{E}_i$ распознал, что "Δ есть обоснование формулы Ψ с посылками $\Phi_1, \ldots, \Phi_m$", то существует натуральный вывод Δ' в $\mathcal{D}_i$ формулы Ψ из посылок $\Phi_1, \ldots, \Phi_m$.*

Доказательство. Пусть алгоритм очевидности $\mathcal{E}_i$ распознал, что "Δ есть обоснование формулы Ψ с посылками $\Phi_1, \ldots, \Phi_m$". Пусть Δ' получается из Δ в результате замены всех непосредственных обоснований вида "Φ by $\alpha_1, \ldots, \alpha_n$" на цепочки непосредственных обоснований, о которых говорится в теореме о корректности $\mathcal{V}_i$. Тогда тривиальной индукцией по числу подвыводов мы получаем, что Δ' есть натуральный вывод в $\mathcal{D}_i$ формулы Ψ из посылок $\Phi_1, \ldots, \Phi_m$. Что и требовалось доказать. □

3. Теорема о полноте

Формулируемая в этом разделе теорема о полноте обеспечивает верифицируемость в рамках продукционной системы $\mathcal{V}_i$ всех правильных непосредственных умозаключений, получаемых в системе натуральной дедукции $\mathcal{D}_i$. Это влечет полноту алгоритма очевидности $\mathcal{E}_i$ в отношении $\mathcal{D}_i$.

Теорема о полноте. *Если формула Ψ является непосредственным следствием формул $\Phi_1, \ldots, \Phi_m$ в системе натуральной дедукции $\mathcal{D}_i$, то задача $\langle \alpha_1, \Phi_1 \rangle \ldots \langle \alpha_m, \Phi_m \rangle \rhd \Psi$ разрешима в $\mathcal{V}_i$.*

Доказательство. Пусть формула Ψ получает обоснование с помощью одного из прямых правил вывода из формул $\Phi_1, \ldots, \Phi_m$:

$$(\alpha_1 : \Phi_1) \ldots (\alpha_m : \Phi_m)\, \Delta\, (\Psi \text{ by } \alpha_1, \ldots, \alpha_m).$$

Рассмотрим все возможные варианты непосредственных обоснований в $\mathcal{D}_i$ формулы Ψ из формул $\Phi_1, \ldots, \Phi_m$.

1. Пусть формула Ψ является аксиомой $\top$. Тогда задача

$$\langle \alpha_1,\, \Phi_1 \rangle \ldots \langle \alpha_m,\, \Phi_m \rangle \rhd \Psi$$

является примитивной. Следовательно, она разрешима в $\mathcal{V}_i$.

2. Пусть формула $\Psi = (\Phi_1 \,\&\, \Phi_2)$ получает обоснование с помощью правила (&в) из формул Φ_1 и Φ_2:

$$(\alpha_1 : \Phi_1)\,(\alpha_2 : \Phi_2)\,\Delta\,(\Phi_1 \,\&\, \Phi_2 \text{ by } \alpha_1,\, \alpha_2).$$

Тогда задача $(\langle \alpha_1,\, \Phi_1 \rangle \langle \alpha_2,\, \Phi_2 \rangle \Gamma \rhd \Phi_1 \,\&\, \Phi_2)$ разрешима в $\mathcal{V}_i$. Действительно, в этом случае поисковое дерево содержит соответствующее допустимой подстановке ε возникающее в результате применения правила декомпозиции ($\rhd\&$) следующее поддерево решения:

$$\begin{array}{ccc} \langle \alpha_1,\, \Phi_1 \rangle \langle \alpha_2,\, \Phi_2 \rangle \Gamma \rhd \Phi_1 & & \langle \alpha_1,\, \Phi_1 \rangle \langle \alpha_2,\, \Phi_2 \rangle \Gamma \rhd \Phi_2 \\ & \nwarrow \!\frown\! \nearrow & \\ & \langle \alpha_1,\, \Phi_1 \rangle \langle \alpha_2,\, \Phi_2 \rangle \Gamma \rhd \Phi_1 \,\&\, \Phi_2 & \end{array}$$

3. Пусть формула $\Psi = (\Upsilon_1 \vee \Upsilon_2)$ является непосредственным следствием формулы Υ_j по правилу вывода ($\vee$в):

$$(\alpha_1 : \Upsilon_j)\,\Delta\,(\Upsilon_1 \vee \Upsilon_2 \text{ by } \alpha_1).$$

Тогда задача $(\langle \alpha_1,\, \Upsilon_j \rangle \Gamma \rhd \Upsilon_1 \vee \Upsilon_2)$ разрешима в $\mathcal{V}_i$. Действительно, в этом случае поисковое дерево содержит соответствующее допустимой подстановке ε деривационное поддерево, которое получается в результате применения правила декомпозиции ($\rhd\vee$):

$$\begin{array}{c} \langle \alpha_1,\, \Upsilon_j \rangle \Gamma \rhd \Upsilon_j \\ \uparrow \\ \langle \alpha_1,\, \Upsilon_j \rangle \Gamma \rhd \Upsilon_1 \vee \Upsilon_2 \end{array}$$

4. Пусть формула $\Psi = (\exists x\, \Upsilon(x))$ получает обоснование с помощью правила ($\exists$в) из формулы $\Upsilon(t)$:

$$(\alpha_1 : \Upsilon(t))\,\Delta\,(\exists x\, \Upsilon(x) \text{ by } \alpha_1).$$

Тогда задача $(\langle \alpha_1,\, \Upsilon(t) \rangle \Gamma \rhd \exists x\, \Upsilon(x))$ разрешима в $\mathcal{V}_i$. Действительно, в этом случае поисковое дерево содержит соответствующее допустимой

подстановке $\theta = \begin{pmatrix} X \\ t \end{pmatrix}$ возникающее в результате применения правила декомпозиции ($\triangleright\exists$) следующее деривационное поддерево:

$$\begin{array}{c} \langle \alpha_1,\, \Upsilon(t) \rangle\, \Gamma \;\triangleright\; \Upsilon(X) \\ \uparrow \\ \langle \alpha_1,\, \Upsilon(t) \rangle\, \Gamma \;\triangleright\; \exists x\, \Upsilon(x) \end{array}$$

5. Пусть формула $\Psi = \Upsilon_j$ является непосредственным следствием формулы $\Upsilon_1 \,\&\, \Upsilon_2$ по правилу вывода (&и):

$$(\alpha_1 : \Upsilon_1 \,\&\, \Upsilon_2)\, \Delta\, (\Upsilon_j \text{ by } \alpha_1).$$

Тогда задача ($\langle \alpha_1,\, \Upsilon_1 \,\&\, \Upsilon_2 \rangle\, \Gamma \triangleright \Upsilon_j$) разрешима в $\mathcal{V}_i$. В этом случае поисковое дерево содержит соответствующее допустимой подстановке ε поддерево решения, которое получается в результате применения правила декомпозиции (&$\triangleright$):

$$\begin{array}{c} \langle \alpha_1,\, \Upsilon_j \rangle\, \Gamma \;\triangleright\; \Upsilon_j \\ \uparrow \\ \langle \alpha_1,\, \Upsilon_1 \,\&\, \Upsilon_2 \rangle\, \Gamma \;\triangleright\; \Upsilon_j \end{array}$$

6. Пусть формула $\Psi = \Upsilon(t)$ получает обоснование с помощью правила ($\forall$и) из формулы $\forall x\, \Upsilon(x)$:

$$(\alpha_1 : \forall x\, \Upsilon(x))\, \Delta\, (\Upsilon(t) \text{ by } \alpha_1).$$

Тогда задача ($\langle \alpha_1,\, \forall x\, \Upsilon(x) \rangle\, \Gamma \;\triangleright\; \Upsilon(t)$) разрешима в $\mathcal{V}_i$. Действительно, в этом случае поисковое дерево содержит соответствующее допустимой подстановке $\theta = \begin{pmatrix} X \\ t \end{pmatrix}$ возникающее в результате применения правила декомпозиции ($\forall\triangleright$) следующее поддерево решения:

$$\begin{array}{c} \langle \alpha_1,\, \Upsilon(X) \rangle\, \Gamma \;\triangleright\; \Upsilon(t) \\ \uparrow \\ \langle \alpha_1,\, \forall x\, \Upsilon(x) \rangle\, \Gamma \;\triangleright\; \Upsilon(t) \end{array}$$

7. Пусть формула Ψ получает обоснование с помощью правила ($\supset$и) из формул $\Upsilon \supset \Psi$ и Υ:

$$(\alpha_1 : \Upsilon \supset \Psi)\, (\alpha_2 : \Upsilon)\, \Delta\, (\Psi \text{ by } \alpha_1,\, \alpha_2).$$

Тогда задача $(\langle \alpha_1,\, \Upsilon \supset \Psi\rangle\, \langle \alpha_2,\, \Upsilon\rangle\, \Gamma \rhd \Psi)$ разрешима в $\mathcal{V}_i$. В этом случае поисковое дерево содержит соответствующее допустимой подстановке ε поддерево решения, которое получается в результате применения правила декомпозиции $(\supset \rhd)$:

$$\begin{array}{c} \langle \alpha_2,\, \Upsilon\rangle\, \Gamma \rhd \Upsilon \qquad \langle \alpha_1,\, \Psi\rangle\, \langle \alpha_2,\, \Upsilon\rangle\, \Gamma \rhd \Psi \\ \nwarrow\!\!\smile\!\!\nearrow \\ \langle \alpha_1,\, \Upsilon \supset \Psi\rangle\, \langle \alpha_2,\, \Upsilon\rangle\, \Gamma \rhd \Psi \end{array}$$

8. Пусть формула Ψ получает обоснование с помощью правила $(\vee\text{и})$ из формул $\Upsilon_1 \vee \Upsilon_2$, $\Upsilon_1 \supset \Psi$ и $\Upsilon_2 \supset \Psi$:

$$(\alpha_1 : \Upsilon_1 \vee \Upsilon_2)\, (\alpha_2 : \Upsilon_1 \supset \Psi)\, (\alpha_3 : \Upsilon_1 \supset \Psi)\, \Delta\, (\Psi \text{ by } \alpha_1,\, \alpha_2,\, \alpha_3).$$

Тогда задача $(\langle \alpha_1,\, \Upsilon_1 \vee \Upsilon_2\rangle\, \langle \alpha_2,\, \Upsilon \supset \Psi\rangle\, \langle \alpha_3,\, \Upsilon \supset \Psi\rangle\, \Gamma \rhd \Psi)$ разрешима в $\mathcal{V}_i$. Действительно, в этом случае поисковое дерево содержит соответствующее допустимой подстановке ε возникающее в результате применения правила декомпозиции $(\supset \rhd)$ следующее поддерево решения:

$$\begin{array}{c} \begin{array}{l}\langle \alpha_2,\, \Upsilon_1 \supset \Psi\rangle \\ \langle \alpha_3,\, \Upsilon_2 \supset \Psi\rangle\end{array}\ \Gamma \rhd \Upsilon_1 \supset \Psi \qquad \begin{array}{l}\langle \alpha_2,\, \Upsilon_1 \supset \Psi\rangle \\ \langle \alpha_3,\, \Upsilon_2 \supset \Psi\rangle\end{array}\ \Gamma \rhd \Upsilon_2 \supset \Psi \\ \nwarrow\!\!\smile\!\!\nearrow \\ \langle \alpha_1,\, \Upsilon_1 \vee \Upsilon_2\rangle\, \langle \alpha_2,\, \Upsilon_1 \supset \Psi\rangle\, \langle \alpha_3,\, \Upsilon_2 \supset \Psi\rangle\, \Gamma \rhd \Psi \end{array}$$

9. Пусть формула Ψ является непосредственным следствием формулы $\bot$ в $\mathcal{D}_2$ по правилу вывода $(\bot\text{и})$:

$$(\alpha_1 : \bot)\, \Delta\, (\Psi \text{ by } \alpha_1).$$

Тогда задача $(\langle \alpha_1,\, \bot\rangle\, \Gamma \rhd \Psi)$ является примитивной в $\mathcal{V}_2$, а, следовательно, эта задача разрешима в $\mathcal{V}_2$. □

Следствие. *Если Δ есть формальное доказательство формулы Ψ из посылок $\Phi_1,\, \ldots,\, \Phi_m$ в $\mathcal{D}_i$, то алгоритм очевидности $\mathcal{E}_i$ установит, что "Δ есть обоснование формулы Ψ с посылками $\Phi_1,\, \ldots,\, \Phi_m$".*

Доказательство. Пусть Δ есть формальный вывод формулы Ψ из посылок $\Phi_1,\, \ldots,\, \Phi_m$ в системе натуральной дедукции $\mathcal{D}_i$. По теореме о полноте $\mathcal{V}_i$ тогда алгоритм очевидности $\mathcal{E}_i$ успешно верифицирует в Δ все непосредственные обоснования вида "Φ by $\alpha_1,\, \ldots,\, \alpha_n$". При этом $\mathcal{E}_i$ также успешно верифицирует в Δ все обоснования вида "Φ proof $\ldots$ qed". Это значит, что с точки зрения $\mathcal{E}_i$ текст Δ является обоснованием формулы Ψ из посылок $\Phi_1,\, \ldots,\, \Phi_m$. Что и требовалось доказать. □

Автор выражает искреннюю признательность О.А. Охотникову за помощь и поддержку при подготовке данной статьи.

Список литературы

[1] Robinson A., Voronkov A., eds, *Handbook of Automated Reasoning*, Elsevier and MIT Press, 2001, 2122 pp.

[2] Wiedijk F., "The QED Manifesto Revisited", *Studies in Logic, Grammar and Rhetoric*, **10**:23 (2007), 121–133

[3] Вторушин Ю.И., "О поиске вывода в системе натуральной дедукции логики предикатов", *Интеллектуальные системы*, **13**:3 (2009), 263–288

[4] Охотников О.А., "О поиске натурального классического логического вывода с использованием частичной скулемизации", *Интеллектуальные системы*, **23**:4 (2019), 39–90

[5] Глушков В.М., *Машина доказывает*, «Знание», Москва, 1981, 64 с.

[6] Grabowski A., Kornilowicz A., Naumowicz A., "Mizar in a Nutshell", *Journal of Formalized Reasoning*, **3**:2 (2010), 153–245

[7] Verchinine K., Lyaletski A., Paskevich A., "System for Automated Deduction (SAD): A Tool for Proof Verification", *CADE-21. Lecture Notes in Computer Science*, **4603** (2007), 398–403

[8] Конев Б.Ю., Жебелеан Т., "Метод подъема решений для работы с метапеременными в системе TH∃OREM∀", *Записки научных семинаров ПОМИ*, **293** (2002), 94–117

About verification of formalized mathematical proofs

Vtorushin Yu.I.

An algorithm for verification of formalized mathematical proofs is considered. Its main part is described in the framework of some production system with metavariables. Theorems of soundness and completeness are proved.

Keywords: automated theorem proving, system for automated deduction, first order language, predicate calculus, production system, artificial intelligence.