Bernhard Schröder, Peter Koepke

ProofML – eine Annotationssprache für natürliche Beweise

Zusammenfassung: Ein Korpus natürlichsprachlich formulierter mathematischer Beweise soll auf das Verhältnis von Ausdruck und argumentativ-semantischer Struktur hin untersucht werden. Zu diesem Zweck werden die relevanten Strukturen mithilfe einer Annotationssprache ProofML in den Beweisen des Korpus annotiert. Die ProofML-Annotation soll dabei einerseits empirisch-analytischen Zwecken dienen, gleichzeitig aber auch eine geeignete Datenstruktur für eine automatische Textanreicherung darstellen, die für einen Beweis-Checker, also ein Programm, das weitgehend natürlichsprachlich formulierte formalwissenschaftliche Beweise auf Ihre Korrektheit hin überprüfen soll, relevante Information kodiert. Derartige Komponenten werden im Rahmen des Projektes NAPROCHE (Natural Proof Checker) entwickelt.

1 Hintergrund

Mathematisches Wissen gilt als Musterbeispiel exakten und expliziten Wissens. Verwendete Begriffe müssen eindeutig definiert sein, Argumente müssen zwingend und ohne Ausnahme sein. Hierzu hat die Mathematik eine leistungsfähige Formelsprache ausgebildet, die oft als Inbegriff mathematischer Genauigkeit angesehen wird.

G. Boole, G. Frege und D. Hilbert schufen mit Aussagenlogik und Quantorenlogik eine formale mathematische Sprache, in der sich mit der mengentheoretischen Begründung der Mathematik alle mathematischen Eigenschaften ausdrücken lassen. Im Gödelschen Vollständigkeitssatz (Gödel 1930) wurde gezeigt, dass die Beweiskalküle von Frege und Hilbert grundsätzlich für alle Zwecke ausreichen: Wenn eine Aussage mathematisch bewiesen werden kann, so kann sie innerhalb eines Beweiskalküls aus den zugrunde gelegten Axiomen abgeleitet werden. Hiermit ist prinzipiell der Weg zur mathematischen *ars iudicandi* gewiesen: Ein formales Argument ist korrekt, wenn es eine Folge von formalen Aussagen ist, von denen jede mittels der Regeln des Beweiskalküls aus Axiomen oder früheren Aussagen gebildet werden kann. Die Überprüfung eines Textes auf diese Kriterien kann mit Hilfe einfacher Algorithmen vorgenommen werden.

Die tatsächliche mathematische Fachsprache weicht aber von dem Ideal formaler Texte und formal-logischer Argumentationen erheblich ab. Argumente in mathematischen Publikationen sind zumeist natürlichsprachliche Texte, in die formalsprachliche Konstrukte eingestreut sind. Diese hybride Sprache soll im Folgenden als *natürliche Beweissprache* bezeichnet werden. Je nach Gebiet und Textsorte können die formalen und die natürlichsprachlichen Textteile in der natürlichen Beweissprache unterschied-

lich distribuiert sein. Die auf diese Weise z.T. informelle Art der Beweisformulierung erschwert die formale Überprüfung der Korrektheit der Beweise. Zwar gilt die prinzipielle Übersetzbarkeit der Beweise in Ableitungen eines prädikatenlogischen Beweiskalküls – der dann mithilfe eines Beweis-Checkers automatisch überprüfbar wäre – als Korrektheitskriterium, gleichzeitig aber wird die praktische Durchführbarkeit eines derartigen Programms traditionell skeptisch beurteilt.

Das Beweisüberprüfungssystem Mizar von A. Trybulec u.a. (Rudnicki 1992) überbrückt die Kluft zwischen der mathematischen Publikationssprache und traditionellen formalen Kalkülsprachen ein Stück weit. Mizar zeichnet sich durch eine formale Sprache aus, die wichtige Ähnlichkeiten mit der natürlichen Beweissprache besitzt. Die Mizar-Sprache ist auf Grund einer ansatzweise natürlichsprachlichen Grammatik und geschickter Konventionen "lesbar". Zudem erlaubt der Mizar-Beweis-Checker das Überspringen einiger "trivialer" Beweisschritte, die auch in mathematischen Publikationen üblicherweise nicht genannt werden.

In Zinn 2000 wird ein Ansatz für ein System zur maschinellen Interpretation natürlicher Beweise beschrieben. Das System übersetzt die natürlichsprachliche Eingabe zunächst in eine PRS genannte DRS-ähnliche teilweise semantisch unterspezifizierte Struktur, die von einem Beweisplaner anhand von Beweismustern disambiguiert wird. Der Ansatz von NAPROCHE und ProofML weicht von dem beschriebenen Ansatz insofern ab, als die semantische Vorstrukturierung durch Textanreicherungswerkzeuge nicht in einer separaten Datenstruktur, sondern möglichst weitgehend im Text selbst geschehen soll. Prinzipiell soll die Möglichkeit der Kombination automatisierter und intellektueller Textanreicherungsprozesse gegeben sein. Die Schaffung von Autorenwerkzeugen, die das Schreiben von Beweisen mit klar spezifizierbaren Anforderungen an die formale Überprüfbarkeit unterstützen, soll auf ProofML-Basis möglich sein.

ProofML muss zu diesem Zweck Mittel bieten, mit denen die natürlich-beweissprachlichen Ausdrücke auf möglichst feinkörniger phrasaler Ebene mit der intendierten Interpretation in linguistisch motivierter Weise annotiert werden. Diese werden im folgenden Abschnitt diskutiert. ProofML wird auf der Basis von XML 1.0 definiert.

2 ProofML

In ProofML werden die natürlichsprachlichen Äquivalente der aussagen- und prädikatenlogischen Operatoren mit ihren Argumenten, also die logischen Ausdrücke, und die Beweisstruktur in einer Weise annotiert, dass bei korrekter Interpretation der deskriptiven Ausdrücke, also der prädikativen und auf Objekte referierenden Ausdrücke, die Beweisstruktur mit maschinellen Mitteln formalsprachlich rekonstruierbar ist.

ProofML-Annotationen werden derzeit rein intellektuell erstellt. Durch die linguistische Analyse von Beweisstrukturen sollen zunehmend automatische Textanreicherungsverfahren entwickelt werden, die ProofML-Strukturen in Beweistexten ergänzen. Da mit rein linguistischen Mitteln viele Ambiguitäten – besonders Skopus-Ambiguitäten, wie z.B. Unsicherheit bezüglich Quantoren-Skopus oder der Reichweite eines Unterbeweises – nicht automatisch auflösbar sind, wird ProofML zum Zweck der automatischen Textanreicherung um Möglichkeiten unterspezifizierter Repräsentation ambiger Phänomene zu ergänzen sein.

2.1 ProofML und die Makrostruktur von Beweisen

Natürliche Beweise werden als Folgen linguistischer Ausdrücke, in der Regel Sätze bzw. Teilsätze, aufgefasst, die Prämissen (<premise>) und Folgerungen (<consequence>) aus diesen Prämissen beinhalten. Das Beweisziel, das Demonstrandum, kann außerhalb dieser Folge – in der Regel vorangestellt – genannt werden oder einfach als Folgerung, in der Regel als letzte, auftauchen. Einige Folgerungen erfordern spezielle Teilbeweise, die hier gewählte Annotation bettet die entsprechenden Teilbeweise in das Folgerungselement ein. Neben diesen Kernbestandteilen beinhalten Beweise vielfach noch motivierende oder beweisgliedernde Hinweise.

Prämissen können in Beweisen unterschiedliche Formen aufweisen. Sie können in Form konditionaler Nebensätze eingeführt werden, in Hauptsätzen mit bestimmten Schlüsselwörtern wie *angenommen* oder in Hauptsätzen mit einem adhortativen Modus, wie *Sei n eine natürliche Zahl.* Im Deutschen wird hier in der Regel der Konjunktiv I gewählt, oft mit Verberststellung. Es scheint, dass diese unterschiedlichen Formen Indikatoren für pragmatisch unterschiedliche Prämissenverwendungsweisen sind, die dem Leser Hinweise auf den Status der Prämisse geben sollen und die bei der formalen Rekonstruktion des Beweises am ehesten durch den Skopus der Prämisse reflektiert werden. Insbesondere scheinen die adhortativen Konstruktionen zu maximal möglichem Prämissenskopus zu tendieren. Der Prämissenskopus wird in ProofML durch die Beweisklammerung repräsentiert.

Bei einigen Prämissenindikatoren scheint auch eine mit ihnen verknüpfte und für das Beweisverständnis evtl. wichtige Präsupposition bei der Wahl des Indikators ausschlaggebend, so beim Indikator *wähle ein P x* (für ein Prädikat *P*) die Präsupposition, dass es auch mindestens ein *P* gibt. Diese pragmatischen Aspekte der Prämissenformulierung sind noch weiter zu untersuchen. Derzeit wird deshalb in ProofML nur die äußere Form des Prämissenindikators im Attribut mode annotiert.

Demonstranda, Prämissen und Folgerungen und ihre linguistischen Konstituenten werden als Ausdrücke (<expression>) unterschiedlichen Typs (Proposition, Prädikat, Objekt) und unterschiedlicher Form (Quantoren, Konjunktionen usw.), sofern sie wei-

ter zergliederbar sind, klassifiziert; die Form gibt Auskunft darüber, welche Teilausdrücke zu erwarten sind.

2.2 ProofML und MathML

Zur Repräsentation der im Beweis eingebetteten mathematischen Formeln wird MathML 2.0 verwendet. In MathML wird zwischen einem an der Semantik mathematischer Ausdrücke orientiertem und von der Darstellung weitgehend abstrahierenden Content-Markup und einem eher an der Erscheinungsform der Formeln orientierten Presentation-Markup unterschieden. Für die Zwecke automatischen Überprüfens von Beweisen ist prinzipiell das an der Semantik mathematischer Ausdrücke orientierte Content-Markup vorzuziehen. Nicht immer erlaubt Content-Markup aber die getreue Wiedergabe der linguistischen Struktur. Im Beispiel, das im Anhang wiedergegeben wird, musste einem Presentation-Markup der Vorzug gegeben werden, da die Übertragung von Ausdrücken, die fortzusetzende Reihen andeuten, wie

(1) $p_1,...,p_r$

ins Content-Markup einen starken Eingriff in die linguistische Struktur bedeutet hätte. Der andeutende Ausdruck (1) dient nicht nur zur Bezeichnung einer Menge, sondern führt darüber hinaus eine natürliche Zahl r und Bezeichner $p_1,...,p_r$ für Primzahlen als neue explizit benannte Diskursreferenten ein (im Code durch <qvar> markiert), auf die im späteren Verlauf des Beweises anaphorisch Bezug genommen werden kann. Für Mengen variabler Größe steht im Content-Markup aber nur die Möglichkeit offen, sie durch eine Prädikatsnotation einzuführen, also etwa $\{p \mid p$ ist eine Primzahl$\}$. In dieser Notation werden keine Diskursreferenten außer der Menge eingeführt.

2.3 ProofML und linguistische Struktur

Die Proof-ML-Annotation, die auf eine Explizierung der semantischen Beweisstruktur zielt, hält sich möglichst eng an die syntaktische Form der Beweise, damit eine möglichst enge und feingliedrige Annotation an den sprachlichen Ausdrücken vorgenommen werden kann. Das hat für die Annotation logischer Strukturen einige Konsequenzen, um die es in den folgenden Abschnitten gehen soll. Die ProofML-Konstrukte sollen jeweils im Hinblick auf deren intendierte semantische Interpretation vorgestellt werden.

Generalisierte Quantoren

Natürlichsprachliche Quantoren werden im Sinne von (Barwise and Cooper 1980) als generalisierte Quantoren interpretiert, in quantifizierten Ausdrücken werden Determinierer <det>, Restriktor <restr> und Skopus <scope> unterschieden. Damit bleibt

die semantische Annotation sehr nah an der natürlichsprachlichen Syntax, Nominalphrasen (und je nach Analyse auch Präpositionalphrasen) als ganze werden als Quantoren annotiert. Die Einführung von gebundenen Variablen über die im Text genannten hinaus erübrigt sich zumeist.

Im Beispielfall des ersten Satzes des Beweises im Anhang ist die Präpositionalphrase *für eine beliebige endliche Menge* $\{p_1,...,p_r\}$ *von Primzahlen* als ein Quantor zu interpretieren und der Rest des Satzes als der zugehörige Skopus, der seinerseits wieder Quantoren enthält. Der Quantor enthält den Determinierer *eine (beliebige)* und das als Restriktor zu interpretierende Prädikat *endliche Menge* $\{p_1,...,p_r\}$ *von Primzahlen*, in dem durch die Mengennotation Variablen eingeführt werden. Sei *ex* beispielsweise ein Determinierer mit

$$\mathrm{ex}(P)(Q) \Leftrightarrow (\exists x\,[P(x) \wedge Q(x)])$$

und *ex_pl* ein Determinierer mit

$$\mathrm{ex\_pl}(P)(Q) \Leftrightarrow (\exists X\,[\forall x \in X\,[P(x)] \wedge Q(X)])$$

Dann ließe sich der Quantor *Primzahlen* als Quantor *ex_pl(primzahl)* wiedergeben. *Menge von* ist als zweistellige Relation zu interpretieren, die hier durch *endliche* modifiziert wird, so dass der Ausdruck *eine endliche Menge von Primzahlen* als der Quantor

$$\mathrm{ex}(\lambda x[\mathrm{ex\_pl}(\mathrm{primzahl})(\lambda Y\,[\mathrm{endlich}(\mathrm{menge\_von})(x,Y)])])$$

zu interpretieren ist, dies ist äquivalent mit

$$\lambda S\,[\exists x,Y\,[\mathrm{endlich}(\mathrm{menge\_von})(x,Y) \wedge \forall y \in Y\,[\mathrm{primzahl}(y)] \wedge S(x)]]$$

Man beachte, dass das zweite Argument der Relation *Menge von* eine kollektive Entität bestehend aus mehreren Entitäten ist, die hier selbst als Menge notiert wird. Der (leere) Determinierer des Ausdrucks *von Primzahlen* ist im Anhang entsprechend als kollektiv zu interpretierender pluralischer Determinierer annotiert (<det type="ex" number="plural" pluraltype="collective"/>) (zur Unterscheidung kollektiv/distributiv s.u.). Auf die in der Mengennotation eingeführten Variablen wird im nachfolgenden Abschnitt eingegangen.

Da natürlichsprachliche Prädikate auch diskontinuierliche Konstituenten sein können, wird syntaktisch zugelassen, dass Determinierer und Restriktor im Skopus eingeschachtelt sind, wenn das Skopus-Prädikat den Quantor umschließt.

Dynamische Existenzquantoren

Existenzquantoren (also Quantoren mit einem Determinierer <det type="ex">) werden im Sinne der dynamischen Prädikatenlogik (DPL) (Groenendijk and Stockhof 1991) interpretiert, es gilt also u.A.

$$(\exists x\,[\alpha]) \wedge \beta \Leftrightarrow \exists x\,[\alpha \wedge \beta]$$

$$(\exists x\,[\alpha]) \rightarrow \beta \Leftrightarrow \forall x\,[\alpha \rightarrow \beta]$$

auch wenn x in β frei vorkommt. Die Feststellung, dass ein Existenzquantor im Antezedenz wie ein Allquantor über Antezedenz und Konsequenz zu interpretieren ist, gilt

dabei keineswegs nur in Konditionalen innerhalb eines Aussagesatzes, sondern auch im Verhältnis von Beweisannahmen und Folgerungen. Ein Syntagma wie

> Sei $p(x)$ ein reelles Polynom vom Grad $n \geq 1$ mit höchstem Koeffizienten 1, dessen Nullstellen alle reell sind. Dann [...]
> (Aigner und Ziegler 2002, 128)

ist zu interpretieren als

> Für jedes reelle Polynom vom Grad $n \geq 1$ mit höchstem Koeffizienten 1, dessen Nullstellen alle reell sind, gilt: Wenn es gleich $p(x)$ ist, dann [...]

Da in der DPL-Semantik Formeln als partielle Funktionen auf Variablenbelegungen interpretiert werden, ist die Assoziation von dynamischen Existenzquantoren mit Variablen wesentlich, ganz im Gegensatz zum Ansatz, der Quantoren im Sinne generalisierter Quantoren versteht und auf an Quantoren gebundene Variablen verzichtet. DPL-Variablen entsprechen in natürlichsprachlichen Ausdrücken anaphorische Ausdrücke wie Pronomina und definite Nominalphrasen. Zu DPL-Variablen äquivalente Information kann man in die Annotation integrieren, indem man den Quantor bzw. seinen Determinierer mit einem eindeutigen id-Attribut versieht und von den anaphorischen Ausdrücken aus mit einem idref-Attribut auf den zugehörigen Quantor als Antezedenten verweist.

Wegen des hohen Ambiguitätspotenzials von Pronomina wird in natürlichen Beweisen von den Autoren zumeist auf Pronominalanaphern zugunsten expliziter Variablen verzichtet, ebenso werden zu Nominalphrasenanaphern häufig disambiguierende explizite Variablen als Apposition hinzugesetzt, vgl. die Verwendung von *p* und *n* im zweiten Satz des Beweises im Anhang. Bei ihrer Einführung in den Diskurs werden derartige Variablen mit <qvar> annotiert.

Neue Variablen oder Ausdrücke mit neuen Variablen werden (a) als eigenständige Nominalphrasen, wie in *sei M eine endliche Menge*, oder (b) als Apposition zu einer Nominalphrase eingeführt, also begleitend zu einem Quantor, eingeführt. Fall (a) tritt in der Regel in Prämissen auf, hier kann die Variable immer als durch einen dynamischen Existenzquantor gebunden aufgefasst werden.

> Sei *M* eine endliche Menge. Dann α.

ist zu interpretieren als

> $(\exists M[\text{endlich(menge)(M)}]) \rightarrow \alpha$

Durch diese Interpretation wird auch der maximale Geltungsbereich der Variablen, der mit dem Skopus der Prämisse identisch ist, korrekt beschrieben.

Im Falle (b) einer Variablen als Apposition wie in

> *für eine beliebige endliche Menge M*

ist die Nominalphrase als

> $\text{ex}_M(\text{endlich(menge)})$

mit

> $\text{ex}_\xi(P)(Q) \Leftrightarrow (\exists x\,[P(x) \wedge \exists\xi'[\xi{=}x] \wedge Q(x)])$

zu analysieren, für andere Determinierer entsprechend. Für gewöhnlich ist dabei $\xi'=\xi$. Nur in Fällen, in denen statt einer einfachen Variable ein komplexer Ausdruck mit Variablen steht, ist ξ' als die Folge der neuen Variablen in ξ zu interpretieren.

für eine beliebige endliche Menge $\{p_1,...,p_r\}$ von Primzahlen
wäre also zu interpretieren als

$$\mathrm{ex}_{\{p1,...,pr\}}(\lambda x[\mathrm{ex\_pl(primzahl)}(\lambda Y\,[\mathrm{endlich(menge\_von)}(x,Y)])])$$

also

$$\lambda S\,[\exists x,Y\,[\mathrm{endlich(menge\_von)}(x,Y) \wedge \exists p,r[\,\{p_1,...,p_r\}=x] \wedge \forall y \in Y\,[\mathrm{primzahl}(y)] \wedge S(x)]]$$

Kollektiv vs. distributiv

Eine Quelle von Ambiguitäten in natürlichen Beweisen ist die Möglichkeit, Konjunktionsoperatoren und pluralische Nominalphrasen kollektiv, wie in *die ersten r Primzahlen bilden eine endliche Menge*, oder distributiv, wie in *die ersten r Primzahlen sind von p verschieden*, zu interpretieren. Deshalb wird in ProofML zwischen der kollektiven und distributiven Verwendungsweise des Konjunktionsoperators und der Pluralverwendung unterschieden. Zu diesem Zweck wird bei Determinierern grundsätzlich zwischen Numerus Singular und Plural mit Hilfe des Attributs number unterschieden und beim Plural zwischen der kollektiven und der distributiven Verwendungsweise durch das Attribut pluraltype. Bei Konjunktionen, also Ausdrücken vom Typ conj leistet dies das Attribut conjtype.

Ellipsen

Natürliche Beweise können verschiedenste Typen andeutender Konstruktionen beinhalten, die beim Leser die Kompetenz voraussetzen, Leerstellen zu füllen und Analogien zu ziehen, etwa bei Ausdrücken wie *in den übrigen Fällen verfahre man ebenso* im Kontext einer Fallunterscheidung, bei der nur einige Fälle ausführlich behandelt werden. Lücken dieser Art sind nicht durch linguistische Kompetenz zu schließen, auch nicht durch Auswahl aus endlich vielen Interpretationsmöglichkeiten, die sich bei Ambiguitäten anbieten. *ebenso* verweist in derartigen Kontexten in der Regel nicht auf ein textuell gegebenes linguistisches Objekt. Die disambiguierenden Annotationsmittel von ProofML sind zur Füllung derartiger Beweislücken wenig geeignet.

Anders verhält es sich jedoch bei Konstruktionen, bei denen durch Anaphern oder Ellipsen explizit oder implizit auf vorangehende Ausdrücke Bezug genommen wird. Hier kann der Bezug durch referenzierende Annotation mittels des Attributs idref hergestellt werden. Im Beispiel des Anhangs wird entweder negierend auf ein Antezedenz Bezug genommen (*sonst:* reftype="neg") oder durch Herstellung einer Parallele (*auch:* reftype="parallel"). Der Ausdruck *auch die 1* im zweiten Satz des Beweises verweist auf einen vorangehenden Teilsatz. Die Ellipse ist hier aufzulösen, indem in dem vorausgehenden Teilsatz ein in Typ und Kasus kongruenter Satzteil als parallele

Konstituente gegen *die 1* ausgetauscht wird. Zur Auflösung der Ellipse sind folglich nur der Teilsatz und die parallele Konstituente jeweils per idref zu identifizieren.

3 Zusammenfassung

Mit ProofML annotierte Beweise bieten eine Ausgangsbasis für eine korpusanalytische Untersuchung der natürlichen Beweissprache und sollen aufgrund der Disambiguierung durch die Annotation eine weitgehend automatisierte Umsetzung in maschinell interpretierbare formale Beweise erlauben, sofern man von Beweislücken absieht, die nicht durch Auflösung innertextueller Referenzen zu schließen sind. Das unmittelbare Ziel der Korpusanalyse ist die ebenfalls weitgehend automatisierte Anreicherung von Beweistexten mit ProofML-Annotation.

4 Anhänge

4.1 Euklids Beweis für die Unendlichkeit der Primzahlen

Euklids Beweis.
Für eine beliebige endliche Menge $\{p_1,...,p_r\}$ von Primzahlen sei $n := p_1\ p_2\ ...\ p_r + 1$ und p ein Primteiler von n. Wir sehen, dass p von allen p_i verschieden ist, da sonst p sowohl die Zahl n als auch das Produkt $p_1\ p_2\ ...\ p_r$ teilen würde, somit auch die 1, was nicht sein kann. Eine endliche Menge $\{p_1,...,p_r\}$ kann also niemals die Menge aller Primzahlen sein.
(Aigner und Ziegler 2002)

4.2 Euklids Beweis für die Unendlichkeit der Primzahlen in ProofML

```
<?xml version="1.0"?>
<proof xmlns="http://www.ikp.uni-bonn.de/ProofML"
       xmlns:proofml="http://www.ikp.uni-bonn.de/ProofML"
       type="tollens">
   <premise mode="adhort">
      <expression form="quantor" type="proposition">
         <det type="ex" number="singular">
            F&uuml;r eine beliebige
         </det>
         <restr>
            <expression form="quantor" type="predicate">
               <scope>
                  endliche Menge
                  <math xmlns="http://www.w3.org/1998/Math/MathML">
                    <mrow>
                     <proofml:qvar>
                       <mo> { </mo>
                        <mrow>
                           <mrow> <msub>
                              <mi>
                             <proofml:qvar type="function"
                                     argnotation="sub">p
                               </proofml:qvar>
                              </mi>
                              <mn>1</mn>
                           </msub> </mrow>
                           <mo> , </mo>
                           <mi> ... </mi>
                           <mo> , </mo>
                           <mi> <msub>
```

```
                        <mi>p</mi>
                        <mi><proofml:qvar>r</proofml:qvar></mi>
                     </msub> </mi>
                  </mrow>
                 <mo> } </mo>
               </proofml:qvar>
            </mrow>
         </math>
      </scope>
      <det type="ex" number="plural" pluraltype="collective"/>
      <restr>
         von Primzahlen
      </restr>
   </expression>
</restr>
sei
<scope>
   <expression form="conj" type="predicate"
          conjtype="distributive">
      <expression form="simple" type="predicate">
         <math xmlns="http://www.w3.org/1998/Math/MathML">
           <mrow>
              <mi> n </mi>
              <mo> := </mo>
                  <mrow>
                     <msub><mi>p</mi><mn>1</mn></msub>
                  </mrow>
                  <mo> &InvisibleTimes; </mo>
                  <mrow>
                     <msub><mi>p</mi><mn>2</mn></msub>
                  </mrow>
                  <mo> &InvisibleTimes;</mo>
                  <mi> ... </mi>
                  <mo> &InvisibleTimes; </mo>
                  <mrow>
                     <msub><mi>p</mi><mi>r</mi></msub>
                  </mrow>
                  <mo> + </mo>
                  <mn> 1 </mn>
           </mrow>
         </math>
      </expression>
      und
      <expression form="simple" type="predicate">
         <math xmlns="http://www.w3.org/1998/Math/MathML">
           <mrow><mi>p</mi></mrow>
         </math>
         ein Primteiler von
         <math xmlns="http://www.w3.org/1998/Math/MathML">
           <mrow><mi>n</mi></mrow>
         </math>
```

```
            </expression>
         </expression>
      </scope>
   .
   </expression>
</premise>
<consequence>
   <proof type="tollens">
      <demonstrandum id="dem1">
         Wir sehen, dass
         <expression form="quantor" type="proposition">
            <scope>
                  <math xmlns="http://www.w3.org/1998/Math/MathML">
                    <mrow><mi>p</mi></mrow>
                  </math>
                  von
                  <det type="all" number="plural"
                        pluraltype="distributive">
                     allen
                     <math
                         xmlns="http://www.w3.org/1998/Math/MathML">
                       <mrow><msub>
                          <mi>p</mi>
                          <mi><proofml:qvar>i</proofml:qvar></mi>
                       </msub></mrow>
                  </math>
               </det>
                  verschieden ist,
         </scope>
      </expression>
   </demonstrandum>
   da
   <premise idref="dem1" reftype="neg">
      sonst
   </premise>
   <consequence>
      <expression type="proposition" idref="dem2">
          <math xmlns="http://www.w3.org/1998/Math/MathML">
            <mrow><mi>p</mi></mrow>
          </math>
          <expression type="object" form="conj"
                   conjtype="distributive" id="dem3">
             sowohl
             <expression type="object">
                die Zahl
                <math xmlns="http://www.w3.org/1998/Math/MathML">
                   <mrow><mi>n</mi></mrow>
                </math>
             </expression>
             als auch
             <expression type="object">
```

```
                das Produkt
                <math xmlns="http://www.w3.org/1998/Math/MathML">
                 <mrow>
                    <mrow>
                       <msub><mi>p</mi><mn>1</mn></msub>
                    </mrow>
                    <mo> &InvisibleTimes; </mo>
                    <mrow>
                       <msub><mi>p</mi><mn>2</mn></msub>
                    </mrow>
                    <mo> &InvisibleTimes; </mo>
                    <mi> ... </mi>
                    <mo> &InvisibleTimes; </mo>
                    <mrow>
                       <msub><mi>p</mi><mi>r</mi></msub>
                    </mrow>
                 </mrow>
                </math>
             </expression>
          </expression>
          teilen w&uuml;rde,
      </expression>
   </consequence>
   <consequence idref="dem2" reftype="parallel">
      somit auch die
      <expression idref="dem3">
         <math xmlns="http://www.w3.org/1998/Math/MathML">
          <mrow><mn>1</mn></mrow>
         </math>
      </expression>
   </consequence>
   ,
   <consequence>
      <expression type="contradictio">
         was nicht sein kann
      </expression>
   </consequence>
</proof>
      .
</consequence>
<consequence><demonstrandum>
   <expression type="proposition" form="neg">
      <expression type="proposition" form="quantor">
         <det type="ex" number="singular">
            Eine
         </det>
         <restr>
            endliche Menge
            <math xmlns="http://www.w3.org/1998/Math/MathML">
             <mrow>
                <mo> { </mo>
```

```
                  <mrow>
                     <mrow> <msub><mi>p</mi><mn>1</mn></msub> </mrow>
                     <mo> , </mo>
                     <mi> ... </mi>
                     <mo> , </mo>
                     <mi> <msub><mi>p</mi><mi>r</mi></msub> </mi>
                   </mrow>
                  <mo> } </mo>
               </mrow>
            </math>
         </restr>
         kann also niemals
         <scope>
            <expression type="predicate" form="quantor">
               <scope>
                  die Menge
                  <det type="all" number="plural"
                        pluraltype="collective">
                     <em>aller</em>
                  </det>
                  <restr>Primzahlen</restr>
                  sein.
               </scope>
            </expression>
         </scope>
      </expression>
   </expression>
</demonstrandum></consequence>
</proof>
```

Literatur

Aigner, M. and Ziegler, G. M. (2002). Das BUCH der Beweise. Springer, Berlin; Heidelberg.
Barwise, J. and Cooper, R. (1980). Generalized quantifiers and natural language. Linguistics and Philosophy, 4: 159-218.
Gamut, L. T. F. (1991). Logic, Language, and Meaning, volume 1: Introduction to Logic. The University of Chicago Press, Chicago; London.
Gödel, K. (1930). Die Vollständigkeit der Axiome des logischen Funktionenkalküls. Monatshefte f. ur Mathematik und Physik, 37.
Groenendijk, J. and Stockhof, M. (1991). Dynamic predicate logic. Linguistics and Philosophy, 14: 39-100.
Rudnicki, P. (1992). An overview of the Mizar project. In Proceedings of the 1992 Workshop on Types for Proofs and Programs, Bastad. Chalmers University of Technology.
Zinn, C. (1999).Understanding mathematical discourse. In: Proceedings of the Workshop on the Semantics and Pragmatics of Dialogue, Amsterdam University.
Zinn, C. (2000). Towards the mechanical verification of textbook proofs. 7th Workshop on Logic, Language, Information and Computation (WOLLIC-2000), Natal, Brasilien.

Internetquellen

Mizar: http://mizar.uwb.edu.pl/
MathML: http://www.w3c.org/Math