Typesetting the "Begriffsschrift" by Gottlob Frege in plain TEX

Udo Wermuth

Abstract

A macro package, gfnotation, is described that can be used to typeset the monograph "Begriffsschrift" published by Gottlob Frege in the year 1879. The package contains two methods to input the unusual notation invented by Frege. The "symbolic representation" allows complete control about the elements and their positions and the "short form" simplifies the complexity to enter the formulas. The package includes macros to build chains of inferences and avoid problems with page breaks. It has been successfully applied to typeset the "Begriffsschrift".

1 Introduction

A well-known strength of TEX is its capability to typeset mathematics. In the long history of printing mathematical formulas, some notations have appeared which are no longer in use. To typeset such notations TEX sometimes does not provide an easy solution. A book with such an outdated notation, which probably only its inventor ever used, is the "Begriffsschrift" [4] published by Gottlob Frege in 1879. Figure 1 shows a page from the book.

I own a facsimile reprint of the "Begriffsschrift", and asked myself how it can be typeset with plain TEX. Of course, I realized that this would require much macro programming. My first goal was to produce a layout that comes as close as possible to the one used in the original printing of the "Begriffsschrift". A second goal was to create a useful set of macros to typeset the whole book without great difficulty and not just a single formula. The output of my macros [25] for Fig. 1 is shown in Fig. 2.

In this article the macros that I developed to typeset the whole book are sketched, several examples of their output are given, and my approach to the problem is discussed. But first, in the next subsection, I briefly introduce the author. Then I discuss the contents of the "Begriffsschrift" and describe the importance of this monograph. The focus of the next subsections is on the notation and the challenge of typesetting it. In sections 2 and 3 I explain the two macro packages (a symbolic representation and a short form) that I wrote to allow a practical handling of Frege's notation in plain TEX. Finally, in the last section I describe the changes to the format of Frege's notation that occurs in Frege's main work [7] of 1893.

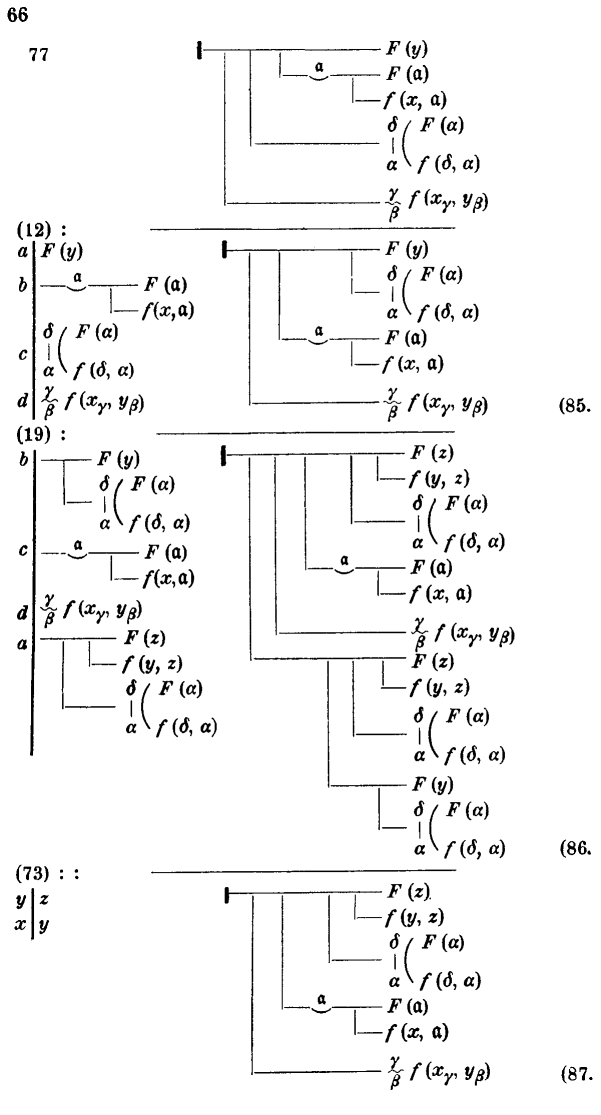


Figure 1: Page 66 of the Begriffsschrift [4] (approx. 62% of original area)

The author Gottlob Frege. Friedrich Ludwig Gottlob Frege (1848–1925) was a German mathematician and according to his own words partly a philosopher [3]: "Every good mathematician is at least half a philosopher, and every good philosopher at least half a mathematician." Several of his articles treat topics in the borderland between mathematics and philosophy. He was interested in an exact and rigorous foundation of mathematics and is one of the founders of the mathematical school called *logicism,* whose ultimate goal is to derive all of mathematics from logic [26]. To reach his goal Frege needed to capture imprecise linguistic phrases by exact and unambiguous statements. And he had to develop an automated system that can transform such statements without using their meaning or ac-

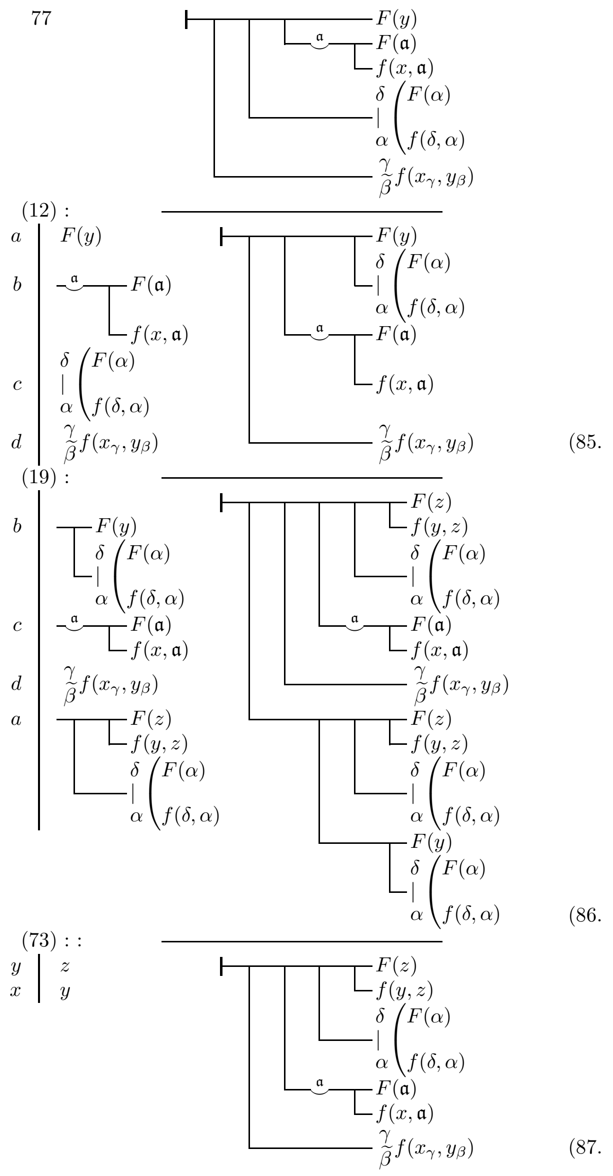


Figure 2: Page of Fig. 1 in plain TeX (approx. 58% of original area)

tual content. This is why he invented a new formal syntax that is introduced in the book "Begriffsschrift". More about Gottlob Frege and his work can be found, for example, online in [27].

W. Quine wrote in 1955 (see [19, p. 158]) "All of modern logic owes an incalculable debt to Frege. If anyone can be singled out as the founder of mathematical logic, it is by all odds he." But during his lifetime the work of Frege was not appreciated and mostly ignored. As a consequence of unfavorable reviews of his work, which show that the contemporary reviewers did not understand the important points, Frege's academic career was severely hampered. He worked as an underpaid Honorary Professor in Jena until his retirement in 1918. In [18] Bertrand Russell is cited with the words: "In spite of the epoch-making nature of [Frege's] discoveries, he remained wholly without recognition until I drew attention to him in 1903." This neglect of his work by other researchers hit Frege hard and filled him with bitterness (see [2]). Maybe even harder was the setback for the scientist through the discovery of a contradiction in his main work. A year before Frege published the second volume of his main work "Grundgesetze der Arithmetik" [7] (Basic Laws of Arithmetic [8]) Bertrand Russell wrote him a letter and pointed out that a contradiction can be constructed from his axioms in the first volume (see Russell's letter [23] and Frege's response [6]). Frege wrote an epilogue for the second volume of the "Grundgesetze" [7, pp. 253–265], in which he explained the problem Russell found. He tried — unsuccessfully, as is known today (see [19]) — to solve it. Russell used for the contradiction an axiom of Frege about which Frege wrote in the preface of the first volume of the "Grundgesetze" that it might cause controversy [7, p. vii].

It seems that Frege was, or more likely became, a man with a difficult personality. In some of his works he attacked other scientists, and he wrote polemical texts (see [2, pp. 46–47] and [3]).

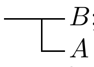
The book "Begriffsschrift". Frege published his first major work in 1879 under the title "Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens" (Begriffsschrift, a formula language, modeled upon that of arithmetic, for pure thought) [4]. This long title is always shortened to "Begriffsschrift" (*Concept Notation*; I use the translation of the German terms as they appear in [5]). In this monograph Frege presented his automated system in the framework of a formal syntax; it was the preparation for his subsequent works where he considered the topics "number" and "quantity". The Begriffsschrift consists of three parts. In the first part the formal system is introduced. The second part shows how to express in this system judgments of pure thought (for example, syllogisms and tautologies like "If a or b takes place, then b or a takes place."). In the last part Frege applied the system to the mathematical theory of sequences.

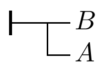
The Begriffsschrift is a short book of less than a hundred pages but great importance is attached to it. The introduction to the translated text in [5, p. 1] contains the words "... it is perhaps the most important single work ever written in logic." And on page 53 of [2] one can find the statement: "It was also the first example of a formal artificial language constructed with a precise syntax. From this point of view, the Begriffsschrift was the ancestor of all programming languages in common use today."

From a typographic point of view the Begriffsschrift is special because of Frege's notation for his formal syntax. This notation did not become an accepted standard and therefore the book is not easy to read. The opposition to the notation included the waste of space and the vertical writing. Frege answered that mathematical formulas are written in a sequence of lines to obtain the advantage of the two dimensions that paper offers [9, p. 8].

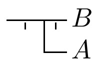
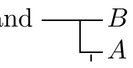
Later in the first volume of the Grundgesetze [7] Frege repeats the definitions of his formal system, but with some small changes to the notation. He wrote [8, p. 5]: "My Begriffsschrift (Halle a. S. 1879) no longer corresponds entirely to my present standpoint; it is therefore to be consulted as an elucidation of what is presented here only with caution." (See [13] for a discussion of the notation and the symbols of [7].)

Frege's Notation. Let's look at the notation of the Begriffsschrift in more detail as this is the main topic of the article. Greek letters are used for terminal strings: A, B, ... (uppercase Alpha, Beta, etc.). Gothic type (i.e., Fraktur) is used for a construction called *concavity* — Frege called them German letters. The following notation is used by Frege:

- *Content stroke* written as $—\!\!—A$; it generates an idea of A; i.e., it is a statement. The truth of A has not yet been judged.
- *Judgment*: $\vdash\!\!—A$; it confirms the truth of A.
- *Negation*: $\neg A$; it states the opposite of A.
- *Affirmation* written as double negation: $\neg\neg A$.
- *Condition*: ; it represents a conclusion.

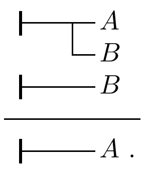
  Frege used a truth table to define the meaning of the condition:  excludes the case in which A is true but B is false, i.e., $A \Rightarrow B$.
- *Generality*: $\overset{\mathfrak{a}}{\smile}\Phi(\mathfrak{a})$; it formulates a statement for all possible values of $\mathfrak{a}$, i.e., it is a "for all" quantification.
- *Identity of content*: $\vdash\!\!—(A \equiv B)$; it establishes the same content for A and B, i.e., A and B are interchangeable.

Frege allowed as variables in the generality not only elements like $\mathfrak{a}$ but also functions, i.e. $\mathfrak{F}$. A quantification with a function appears, for example, in formula 76 of the Begriffsschrift (see Fig. 5(b)).

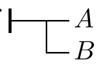
As Frege defined only one binary relation, the condition, some well-known operations lack the symmetrical form in Frege's syntax as they have in modern notation. For example, Frege analyzed words like "and" and "or" and described them in his notation as  for "and" and  for "or".

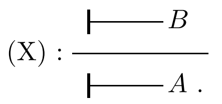
Besides the notation for formulas he introduced a notation for rules, i.e., *inferences.* At first he used only one inference and stated that later applications of the system shall define more modes of inference (see [4, p. vii]). Frege did this in [7]; his notation is described in more detail in section 4.

An inference creates from two formulas a third:

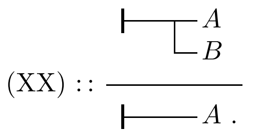


It means: If it is true that A is a conclusion of B and the truth of B is known, then A must be true.

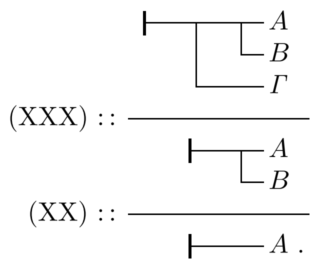
Although there is only one inference three different notations are used in the Begriffsschrift to avoid the repetition of a formula. One of the two input formulas might be referenced by its number instead of being listed again. For example, if  was established and if this formula is called X then the above inference is abbreviated:



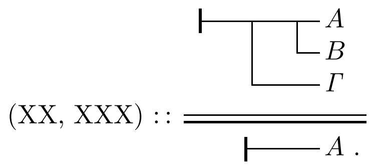
On the other hand, if $\vdash\!\!—B$ is known and called XX then Frege put two colons after the reference number:



And in some places more than two formulas are involved in an inference. If the formula $\vdash\!\!—\Gamma$ is called XXX then the chain of inferences



can be written:



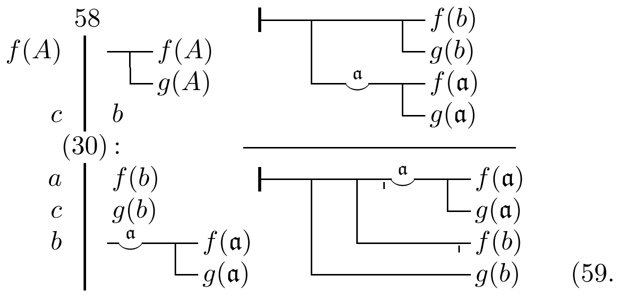
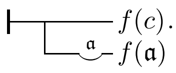
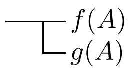
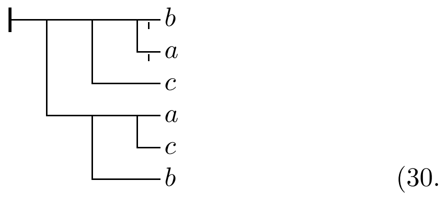


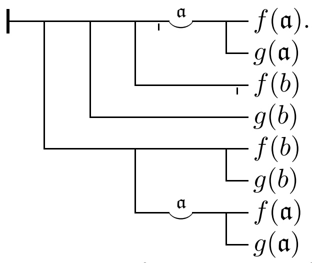
Figure 3: The inference for formula 59 of [4]

In Fig. 1, the second kind of abbreviation is used in the last inference with formula 73. The first two inferences with formulas 12 and 19 are examples of the first kind of abbreviation.

Another important notation is used for substitutions. In a previously derived formula substitutions with other formulas are possible as long as the substitution is done consistently. Frege used a vertical line to the left of a formula and wrote to the left of that line the statement that is replaced and to the right, the replacement. The formula in which the substitution shall be performed is referenced by its number over the top of the line. Figure 3 shows an example.

The first formula has the number 58 and reads . The formula is printed above the horizontal line with two changes: $f(A)$ is replaced by  and c is replaced by b. Formula 30 is used in the inference but it is not shown. And in this formula substitutions are performed too: The three substitutions are listed below the number 30. To allow the reader to perform the inference of formula 59 himself, here is formula 30:



And here, the abovementioned substitutions have been applied to it: . If this formula is placed at the top of Fig. 3, the inference for formula 59 is clear.

Typesetting Frege's notation. Of course, Frege was aware that the typesetting of his notation is very difficult. He argued [10, p. 364] that his two-dimensional form was easier to read and understand than the usual compression of all formulas into a single line. And he was not willing to change his layout to make the work of the typesetters easier. Using one half of a verse by Friedrich Schiller, he wrote (loosely translated): "The ease of typesetting is not, however, the supreme good."

In the next section I present my symbolic coding to reproduce Frege's notation as closely as possible. Then a recursive notation is introduced that allows writing the input for a formula on a single line. Of course, the output is the two-dimensional structure defined by Frege and the lengths of content strokes are automatically calculated to have all terminal strings aligned.

Other people have worked on the problem to typeset Frege's notation in TeX. J. Parsons developed a package *begriff.sty* for LaTeX [21]. It doesn't fulfill my above mentioned goals; for example, the overall look of the notation of the Begriffsschrift is not reached and the alignment of judgment strokes or terminal strings is not easy, as the lengths of the content strokes are set manually. The look of the output is improved in [20]. The article [17] describes a GUI to support the manual work and outputs formulas in the coding of *begriff.sty*; it was used in a translation project for Frege's Grundgesetze (see [1]). The team created its own package *grundgesetze.sty* [22] to typeset the English translation [8] of Frege's main work [7]. The package is based on *begriff.sty* and inherits some of its weak points. The style of Frege's notation in his main work differs from the style of the Begriffsschrift (see section 4).

2 Symbolic representation

It is easy to understand that Frege's notation can be applied in a longer text only if macros are available to support the input. My first decision: Greek and Fraktur letters are entered using control words of length 2; the letter that must be typeset is preceded by either a 'g' for uppercase Greek, or a 'k' for lowercase Greek, or a 'd' for Fraktur. The assignments of characters to Greek letters follows [16, p. 20]. For example, `\gA` produces an uppercase Greek Alpha (A) and `\da` gives $\mathfrak{a}$. The Fraktur font is taken from AMS-TeX's *Euler* family of fonts. (Not all combinations of two letters are allowed. For example, the control word `\dp` is already used by TeX.)

First, I thought to use `\halign` for the formulas, but this approach has some major drawbacks:

- Page breaks in long inferences are not possible when they are typeset as a single alignment. Frege never broke a formula; page breaks occur inside an inference only after the inference line.
- The templates and the horizontal positions of formulas have to be carefully aligned if each formula is an `\halign` itself.
- Laborious counting of &-signs inside a formula is needed to position the parts correctly.

As the notation requires control words for the line segments anyway, the following simple approach seems to be possible:

1. All symbols (and the empty space) of the notation are defined in a uniform length.
2. Normal text lines are used (which might have to be typeset without interline spacing).

I observed that the symbols consist of three parts. This can be seen most easily in the condition. Therefore I defined a macro `\*` with three parameters. (I saved the plain TEX meaning of `\*` as `\discretionarytimes`.)

For example, I write the *triple* `\*---` for the content stroke (———), which consists of three equal parts. `\*-~-` is the single negation (—╷—). `\*-:-` gives the then-part (—┬—) of a condition, which finds in `\*_'-` its partner (└—) in the following line. A `\*_!_` (│) is used to connect a then- and its if-part over several lines. And `\*.a.` builds the concavity (–$\overset{\mathfrak{a}}{\smile}$–) with the letter $\mathfrak{a}$. Although one might think that the colon appears only in the middle of a triple one should consider a triple like `\*:-:` (┬—┬) to get a more compact variant to code two then-parts. (This *compact form* is required in the Grundgesetze [7], which has a two-column page layout. Figure 7 shows the compact form of the formulas of Fig. 1.) In order to connect the symbols in this coding all elements must have the same length. And of course, all horizontal lines must have the same thickness and the same position above the baseline.

First, I identified 19 different parts without the concavity. The flexibility in defining symbols from these parts is more than the Begriffsschrift requires: It uses 8860 triples in the whole text but only 73 different ones; 21 triples occur just once and only 10 more than 100 times.

Here is the list of the seven basic parts:

- `_` is the empty space;
- `-` is a horizontal line for the content stroke and the inference rule;
- `~` is a horizontal line with centered negation;
- `+` is a horizontal line with two centered negation indicators (affirmation);
- `:` is the then-part of the condition;
- `'` is the if-part of the condition;
- `!` is a vertical line (part of the so-called *condition stroke*) that connects an if-part with the corresponding then-part over several lines.

Four symbols are used for special situations:

- `[` is a vertical line placed left and starts a horizontal line (a judgment with a content stroke);
- `|` is the vertical line for the substitution;
- `=` is two horizontal lines used for an inference that involves more than two formulas;
- `3` is two vertical lines used for *definitions* (see Fig. 5).

And finally there are eight symbols to fine-tune the output to match the original in certain situations:

- `"` a skip, that is, no output at all;
- `(` negation, with the indication of negation moved to the left;
- `)` negation, indicator moved to the right;
- `<` negation, indicator placed at the left end;
- `>` negation, indicator placed at the right end;
- `^` affirmation, with indicators moved to the left;
- `/` affirmation, indicators moved to the right;
- `]` a vertical line placed right, that is, only the judgment stroke is printed.

Second, the definition of the concavity requires typesetting of letters. The width of such a letter is considered to be counted as two elements of a triple. So special symbols for shorter elements are defined, which must be used in pairs. Three additional basic symbols are needed:

- `a` is the letter for the concavity (`a` can be replaced by other letters);
- `.` is a short vertical line (i.e., a content stroke);
- `,` is a short vertical line with a centered negation indicator;
- `;` is a short vertical line with double negation.

And two more are needed for fine-tuning:

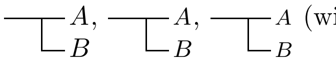
- `@` signals (without any output) that a letter for the concavity follows;
- `` ` `` represents negation but the indicator is moved to the left;

A triple for the concavity must be started with one of the following symbols: ``.,`;@``. If it is not `@`, then the symbol after the letter must be one of ``.,`;``.

I call the definition of formulas in this encoding the *symbolic representation* of Frege's notation. Of course, the production of the input needs a lot of keystrokes. But it allows the creation of any formula, even those not obeying the rules, so that I could typeset the errors in the Begriffsschrift. And using an editor with a monospaced font means that the formula can be read in the input file.

```
      \nlp1\ci{\hss\thinspace58\hss}\rep3\*___\*_[-\*-:-\*---\*-:-\ce{$f(b)$}
\nlpc1{$f(A)$}\*_|_\*-:-\ci{$f(A)$}\rep2\*___\*_!_\*___\*_'-\ce{$g(b)$}
        \nlp1\*_|_\*_'-\ci{$g(A)$}\rep2\*___\*_'-\*.a.\*-:-\ce{$f(\da)$}
   \nlpc1{$c$}\*_|_\ci{$b$}        \rep3\*___\*___\*___\*_'-\ce{$g(\da)$}
       \bcc1/m30:.\rep3\*___\null          \rep6\*---\ecc
   \nlpc1{$a$}\*_|_\ci{$f(b)$}     \rep2\*___\*_[-\*-:-\*-:-\*,a.\*-:-\ce{$f(\da)$}
   \nlpc1{$c$}\*_|_\ci{$g(b)$}     \rep2\*___\*___\*_!_\*_!_\*___\*_'-\ce{$g(\da)$}
   \nlpc1{$b$}\*_|_\*.a.\*-:-\ci{$f(\da)$}   \*___\*_!_\*_'-\*---\*--(\ce{$f(b)$}
        \nlp1\*_|_\*___\*_'-\ci{$g(\da)$}   \*___\*_'-\*---\*---\*---\ce{$g(b)$}
\fono{59}
```

Figure 4: Symbolic representation for Fig. 3

The output scales with the size of the font if size-changing macros are set up similarly to p. 414 of [14]. For example, here is a formula in sizes 10 pt, 9 pt, and 8 pt:  (with constant baselineskip here; that can be changed too).

A detailed example. As an example, the input of Fig. 3 with formula 59 is given here in detail; see Fig. 4. So it is a real world example, containing more than the code for a single formula. Several macros must be introduced to handle the whole structure of the original text.

First, since spaces after the `\*` macro count, I use a `\null` to ignore spaces and to align the substitution and the formulas for the inference in this example.

The control words `\ci` (Character Inside) and `\ce` (Character at the End) place their arguments in an `\hbox`. In the case of `\ci` the box has the width of the `\*` macro and possibly following spaces are ignored. The macro `\fono` (FOrmula NO.) outputs the number of the formula at the right of the text line with an opening parenthesis and a period. And `\rep` (REPeat) is a macro with two arguments. The second argument must be a `\*` sequence. The output looks as if this sequence was entered as many times as the first argument states. For example, `\rep3\*___` produces exactly the same output as `\*___\*___\*___`.

The three main macros are: (a) `\nlp` (NewLine and Position), (b) `\nlpc` (NLP with Character), and (c) `\bcc #1/#2#3:#4#5\ecc` (Begin/End ConClusion). In the five parameters of (c) the third is the number of the used formula that is placed above the substitution rule. The number of colons, which shows the type of inference, is specified by the fourth parameter. The fifth parameter, which is ended by the `\ecc` marker, draws the line or lines required in the notation of the inference. The second parameter positions the colon(s). In the above-displayed case an 'm' (middle) is used. The other possibilities are 'l' (left) and 'r' (right). (In fact, only the 'm' is required by the description given above. But in order to reproduce the original text at one place an 'l' is needed.) The first parameter has the same meaning as the parameter of the `\nlp` macro: it gives the indent from the left as a multiple of the width of the `\*` macro. But the `\nlp` macro does more than just the indentation. It finishes the previous line and uses a strut to specify the height and depth of the new line. This macro is discussed later in more detail. The last macro `\nlpc` is an abbreviation: `\def\nlpc#1#2{\nlp{#1}\llap{#2}}`. It does the work of the `\nlp` macro and places its second argument in the empty space created by the indentation.

My first goal is achieved with this output. It is very close to the text as it is printed in the original. The second goal, to have a set of macros to make the typesetting of the whole book "easy", is not completely achieved. The amount of typing is huge and some counting for the positioning of symbols is required. So the output is acceptable, but the input has to be improved.

Parameters. The macros for the symbolic representation were written to take some parameters for changing the appearance. This is necessary as Frege made some changes in this area for the Grundgesetze [7]. All the internal macro names start with the prefix `\gfbs` and most of them have a German name after this prefix. For example, the dimen register `\gfbsstrichdicke` sets the thickness of the content stroke. And `\gfbsraise` gives the height of the content stroke above the baseline.

Here is a list of a few dimen registers:

- `\gfbsstrichdicke` for horizontal lines; default is 0.5 pt.
- `\gfbsurteildicke` for the judgment bar; default is 1 pt.
- `\gfbsersetzungdicke` for the line that is used in the substitution part; default is 0.8 pt.
- `\gfbsschlussdicke` for the thick inference line; default is 0.8 pt.
- `\gfbsraise` for the height of horizontal lines; default is 0.5 ex.
- `\gfbsneg` for the height of the negation indicator; default is 0.25 ex.

- `\gfbsuht` for the height of the judgment stroke; default is 1.5 ex.
- `\gfbsudp` for the depth of this stroke; default is 0.5 ex.
- `\gfbsschlussabstand` for the distance between the two lines in an inference; default is 2.5 pt.
- `\gfbsvolleeinheit` for the width of a single part in the `\*` macro; default is 0.57 em.

These parameters are used to calculate the values of other dimen registers; for example, the register `\gfbselementdimen` contains the width of a complete `\*` macro and is roughly 3`\gfbsvolleeinheit`. (The units overlap a little bit to make sure that no gaps appear between the line parts.) The above dimen parameters are not used directly in the main macros of the code so that values can be adjusted for different output formats. For example, the following dimensions are used in the Begriffsschrift:

- `\gfbshoehe` equals `\gfbsraise`;
- `\gfbsnegdp` equals `\gfbsneg`;
- `\gfbssdicke` equals `\gfbsstrichdicke`;
- `\gfbsht` is the sum of the value `\gfbshoehe` and the value `\gfbssdicke`;
- `\gfbseinheit` is the width of one part of the `\*` macro minus the overlap;
- `\gfbszweiheit` is $2 \times$ `\gfbsvolleeinheit` minus the overlap.

Besides the listed dimens two flags are defined:

- `\gfbsnegdirekt` controls whether the negation indicator and content stroke touch or leave a small gap; the gap occurs in the Begriffsschrift, but not in the Grundgesetze. The default is `\gfbsnegdirektfalse`, so the gap is present.
- `\gfbsfonoohnepunkt` controls whether the closing period in a formula number is omitted. The default is `\gfbsfonoohnepunktfalse`, i.e., the period is printed.

Macros. In this subsection a few aspects of the macro definitions needed for the symbolic representation are discussed. Let us look at the definition of `\nlp`:

```
\def\nlp#1{\hfil\break\gfbsstrut
   \hskip#1\gfbselementdimen\relax}
```

The `\hskip` sets the current position to a multiple of the width of the `\*` macro.

The control word `\gfbsstrut` provides a strut (see [14, p. 82]) whose height and depth can be set inside the text. Such a strut is needed to get acceptable page breaks. (See pages 79 and 80 of [5] for bad breaks that can occur.) The macro package provides two commands to change the height and depth of the strut either by a specified percentage (`\gfbsreduziereabstandum`) or by an explicit value (`\gfbssetzeabstand`). The control sequence `\gfbsabstandzuruecksetzen` resets the strut to its original height and depth.

The main macro of the symbolic representation is simply a nested `\if` sequence. Well, at its end are 36 `\fi`s, so it might not look very simple. The macro `\gfbsteilelement` processes one parameter of the `\*` macro.

Let's look at a few (simplified) examples:

```
\def\gfbsteilelement#1{%
   \if...
   \else\ifx #1~% negated content stroke
      \hbox to \gfbsvolleeinheit{%
         \gfbsrulefill}%
      \hskip-0.5\gfbseinheit
      \hskip-0.5\gfbssdicke
      \ifgfbsnegdirekt
         \vrule width  \gfbssdicke
                height \gfbshoehe
                depth  \gfbsnegdp
      \else
         \vrule width  \gfbssdicke
                height .8\gfbshoehe
                depth  \gfbsnegdp
      \fi
      \hskip-0.5\gfbssdicke
      \hskip 0.5\gfbseinheit
   \else\ifx...
   \else\ifx #1:% then-connection
      \hbox to \gfbsvolleeinheit{%
         \gfbsrulefill}%
      \hskip-0.5\gfbseinheit
      \hskip-0.5\gfbssdicke
      \vrule width  \gfbssdicke
             height \gfbshoehe
      \hskip-0.5\gfbssdicke
      \hskip 0.5\gfbseinheit
   \else\if...
   \fi\fi...\fi\fi...\fi}
```

`\gfbsrulefill` defines a macro for horizontal rules that have a distance of `\gfbshoehe` above the baseline. It acts like `\hrulefill` (see [14, p. 357]).

```
\def\gfbsrulefill{%
   \leaders\hrule height \gfbsht
                  depth -\gfbshoehe
   \hfill}
```

The concavity is built with the symbol `\smile`: $\smile$ (I don't use [11]). It is placed so that the horizontal lines are attached at its ends. At the right and left the $\smile$ has a little bit more than 1 u empty space. The symbol itself has a total width of 18 u (see [15, p. 441]). This data is used to calculate the seamless junction with the horizontal lines. For very thick lines the thicker $\smile$ of $\mathcal{A}\mathcal{M}\mathcal{S}$-TeX is used. The Fraktur letter is placed above this symbol with the macro `\buildrel` (see [14, p. 437]).

```
\ifdim\gfbssdicke<0.59 pt
   \setbox0=% plain \TeX \smile
      \hbox{$\mathchar"015E$}%
\else
   \setbox0=% \AmSTeX
      \hbox{$\mathchar"0\hexno\cmmibfam5E$}%
\fi
\dimen0=\gfbseinheit \advance\dimen0 by-.5\wd0
\dimen2=\wd0 \divide\dimen2 by 18 % approx. 1u
\advance\dimen0 by 1.5\dimen2
\hbox to \gfbszweiheit{%
   \hbox to\dimen0{\gfbsrulefill}%
   \kern-1.2\dimen2
   \raise\gfbshoehe\hbox{\lower\ht0\hbox{%
        $\buildrel{{\frak #1}}\over{\box0}$}}%
   \kern-1.5\dimen2
   \gfbsrulefill}%
```

As mentioned earlier, the concavity needs twice the width of a single unit and the two other elements of a triple take only a half width each. So the symbols `.,‘;` must appear as a pair.

```
\def\gfbsteilelement#1{%
   \if...
   \else\ifx #1.% half
      \hbox to 0.5\gfbsvolleeinheit{%
         \gfbsrulefill}%
      \ifgfbszweitehaelfte
         \gfbszweitehaelftefalse
      \else
         \gfbshoehlungtrue
         \gfbszweitehaelftetrue
      \fi
   \else\ifx #1, % half and negated
   ...
   \else\ifx #1;% half, double negation
   ...
   \else
      \ifgfbszweitehaelfte
         \errhelp=\gfbshaelftehilfe
         \errmessage{Unexpected ...}%
         \gfbszweitehaelftefalse
   \fi\fi...\fi...\fi...\fi}
```

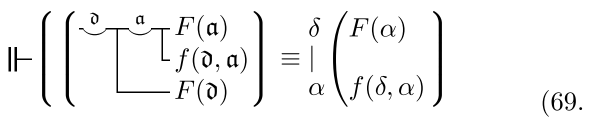
If the flag `\gfbshoehlungtrue` is set then the next symbol is used as the letter for the concavity. The other flag `\gfbszweitehaelfte` signals that a symbol of the set `.,‘;` is used instead of the `@`.

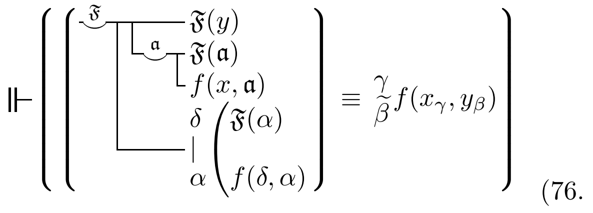
This code snippet shows that error conditions are caught. Using `\newlinechar=‘\^^J`, the following help message is shown if the pairing didn't work.

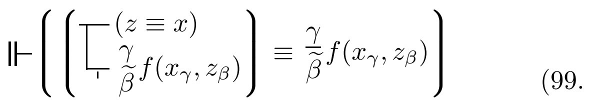
```
\newhelp\gfbshaelftehilfe{After a signaled %
‘‘for all’’ one of (1) \string., (2) \string,, %
(3) \string;^^J or (4) \string‘ must directly %
follow the variable. These symbols must be %
paired^^J in any combination around the %
character that is used as the^^J variable in %
the ‘‘for all’’.}
```

In total more than 20 error messages with associated help messages are coded in the package.

Special symbols. Figure 1 shows that the terminal strings are another idiosyncratic aspect of the work,


(a) Property F is hereditary in the f-sequence.


(b) y follows x in the f-sequence.


(c) z belongs to the f-sequence beginning with x.

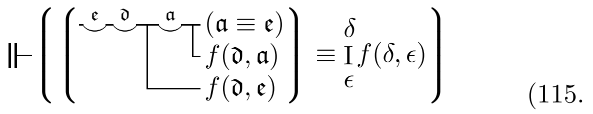

(d) The procedure f is single-valued.

Figure 5: Special symbols of [4] and their meaning

in addition to the Frege notation. These symbols are introduced in the third part of the Begriffsschrift, in which the formalism is applied to the theory of sequences. So these symbols are not part of the formal syntax of the notation, but nevertheless they are needed to typeset the book.

With the use of the abovementioned *definition symbol* (“`\*_3-`” outputs “ $\Vdash$”), Frege introduces four special symbols. All of them can be built with available symbols of *Computer Modern*. The symbol for the definition is followed by an equivalence $((\mathcal{A}) \equiv \mathcal{B})$. $\mathcal{A}$ is a formula and $\mathcal{B}$ is defined as the abbreviation for this formula. Just to show that these definitions can be typeset with the macros the definitions are given in Fig. 5. Because of limitations in space the compact form is used although it is not used in the Begriffsschrift.

The right-hand sides of the definitions are coded as “normal” macros. The first one has one parameter, the second and third two and the last none. The right-hand sides in Fig. 5 are named `\1F`, `\2xy`, `\4xz`, and `\5`, resp. These macros are also used in Fig. 6. (I use digits to build control symbols: The `\3` is defined to be an abbreviation for a frequently used subformula, which is written as its replacement text; see line 1 of Fig. 6. But this is not a “normal” macro and to use it “expansion” must be called first; see the exclamation mark in front of every use of `\3` in

```
\def\3#1#2{*.a.{..{{#1(\da)}}
                  .{{f(#2,\da)}}}}
\gfbskompakttrue % use compact form
\outof p0,0"77"with\thatis
\formula p5|{..{..{..{F(y)}
                    .{!\3Fx}}
                  .{\1F}}
                .{\2xy}}
\followswith p0"12"a.p4s7
\substituting p0 a:{F(y)}
           b:!\3Fx
           c:\1F
           d:\2xy
\whichgives
\formula p5|{..{..{..{F(y)}
                    .{\1F}}
                  .{!\3Fx}}
                .{\2xy}}
\named "85"
\followswith p0"19"a.p4s7
\substituting p0 \ :{\ }
            b:{..{F(y)}
                .{\1F}}
            c:!\3Fx
            d:\2xy
            a:{..{..{F(z)}
                    .{{f(y,z)}}}
                 .{\1F}}
\whichgives
\formula p5|{..{..{..{..{..{F(z)}
                            .{{f(y,z)}}}
                          .{\1F}}
                        .{!\3Fx}}
                      .{\2xy}}
                .{..{..{..{F(z)}
                          .{{f(y,z)}}}
                        .{\1F}}
                      .{..{F(y)}
                        .{\1F}}}}
\named "86"
\followswith p0"73"a:p4s7
\substituting p0 y:z
          x:y
\whichgives
\formula p5|{..{..{..{..{F(z)}
                          .{{f(y,z)}}}
                        .{\1F}}
                      .{!\3Fx}}
                .{\2xy}}
\named "87"
```

Figure 6: The input for Fig. 7 in short form

Fig. 6. The details are discussed in the next section, which introduces the short form.)

3 A recursive short form

Of course, the symbolic representation is not easy to code. It would be much better to reduce the complexity of typing the symbolic representation with additional macros. So I developed a short form that TeX transforms into the symbolic representation to reach the desired quality for the output. Figure 6 shows the input for Fig. 7 that corresponds to Fig. 2 but uses the abovementioned *compact form* for the output (see line 3 in Fig. 6).

The short form has three types of commands:

1. place a content stroke (with or without negation or affirmation indicators, i.e., signs) in front of a formula;
2. place a concavity and optional signs in front of a formula;
3. combine two formulas into a condition construction again with optional signs in front of the two formulas and for the overall construction.

If the terminal strings are called a formula too then every formula in Frege's notation can be build recursively with these commands. The optional judgment stroke is attached after the formula is created.

I decided to use signs in the above-listed cases 2 and 3 and not only in case 1. In this way case 1 is only needed when the content stroke stands alone.

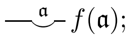
The structures of the commands are as follows:

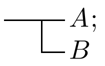
1. `=`⟨*sign*⟩`{`⟨*formula*⟩`}`
2. `*`⟨*sign*⟩⟨*character*⟩⟨*sign*⟩`{`⟨*formula*⟩`}`
3. ⟨*sign*⟩⟨*sign*⟩`{`⟨*formula*⟩`}`⟨*sign*⟩`{`⟨*formula*⟩`}`

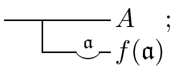
⟨*formula*⟩ is a previously generated formula or a terminal string, which is typeset in math mode. ⟨*sign*⟩ is an optional sign and has one of three values:

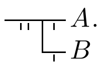
`.` represents no sign;
`-` represents the negation;
`+` represents the affirmation.

Note: The braces around ⟨*formula*⟩ are not required if it is a terminal string of one token. Double braces are required if the string has more than 5 tokens. Here are some examples:

1. `=.a` outputs ——— a;
2. `*.a.{f(\da)}` outputs 

3. `..\gA.\gB` outputs 

4. `..\gA.{*.a.{f(\da)}}` is 

5. `+-\gA-\gB` outputs 

The fourth example shows how the formulas are nested: In the formula of the third example the B is replaced by the formula of example 2. The construction with `=` that is shown in the first example is not often used in nested formulas, as the length of the content strokes in a condition is calculated automatically to have a proper alignment.

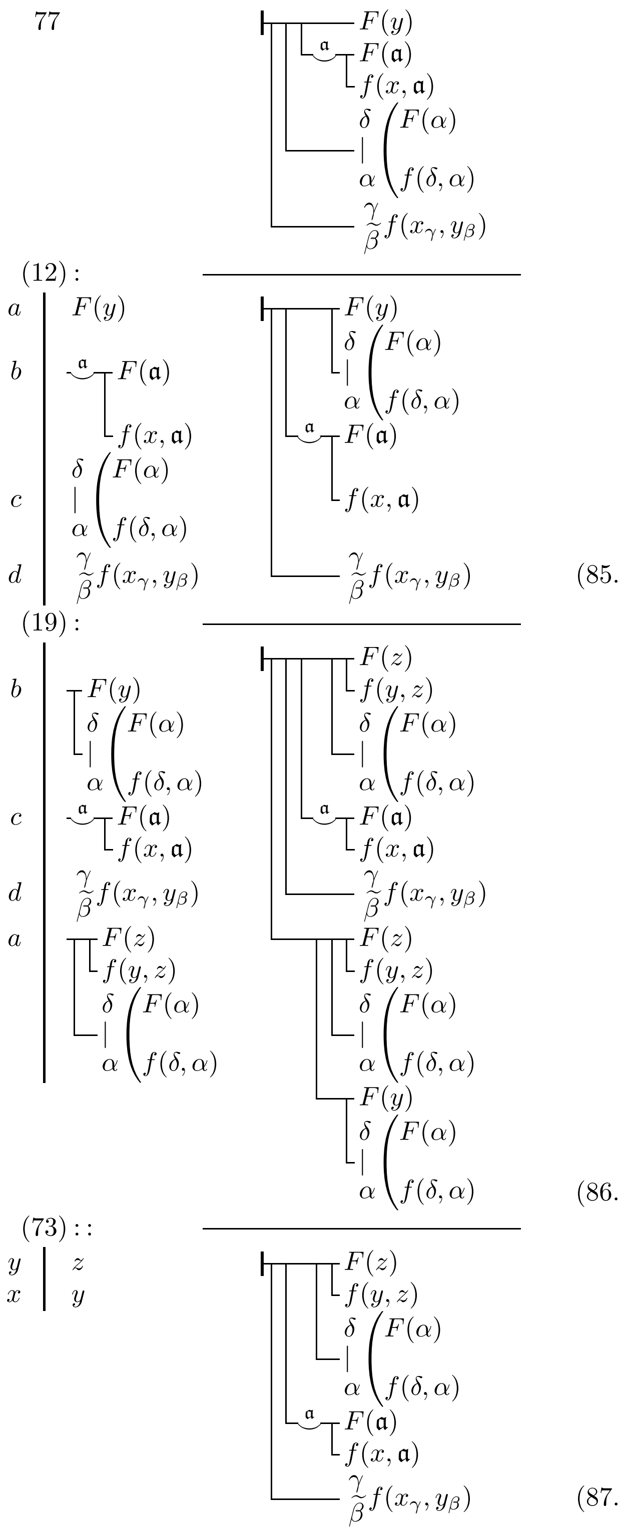
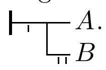
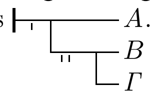


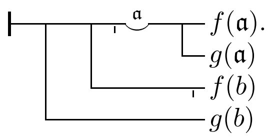
Figure 7: Typesetting Fig. 1 in compact form; typed into this article (Fig. 6 shows the source)

A single formula is started with the command `\frege`, which has two parameters. The first parameter codes the decision if the formula is a judgment; it is a judgment if the first parameter is '`|`' instead of '`.`'. The second parameter is the formula in the short form.

Again a few examples: `\frege|{-.\gA+\gB}`. The generated formula is a condition with an overall negation, an unsigned A, and an affirmed B. This is the output:  The replacement of `\gB` by `{..\gB.\gG}` gives `\frege|{-.\gA+{..\gB.\gG}}` and this code outputs 

Let's go back to formula 59 of Fig. 3. The short form for this formula is

```
\frege|{..{..{*-a.{..{f(\da)}.{g(\da)}}}
         -{f(b)}}
       .{g(b)}}
```

with the expected output 

But remember: in the text, Formula 59 does not stand alone. We need the complete chain of inference with a correct placement of several formulas. Such an inference is not coded with the use of `\frege`, which is only used for formulas in running text. (There are small differences between a formula in running text and in an inference.) In inferences it is replaced by `\formula`, which has an additional parameter to determine the position of the formula. The substitutions are coded as pairs of formulas separated by a colon. Figure 8 shows the complete code for Fig. 3. It uses a control symbol `\0` to make it easier to add the sub-formula with the concavity as explained below.

The parameter type "p*n*" is used to position the formula, in units of the `\*` macro, as before (see the `\nlp` macro above). The four-parameter macro `\outof p#1,#2"#3"with#4\thatis` starts a block of formulas. The first parameter is the position. The third parameter is the number of the formula that follows. The second parameter is special: It is used to place the formula number several lines lower. Usually it is 0 to get the same baseline. But in the Begriffsschrift a few cases appear that have no substitutions and the number is placed one line or two lines lower. (Recall that one of my goals was to reproduce the original text as closely as possible.) The last parameter is a sequence of formula pairs separated by colons; this is the list of substitutions. Such a list is also used as the second parameter of the macro `\substituting p#1 #2\whichgives`.

The macro `\followswith p#1"#2"a#3p#4s#5` is used to typeset the lines indicating the type of

```
\def\0#1{*#1a.{..{f(\da)}.{g(\da)}}}
\outof p1,0"58"with
   f(A):{..{f(A)}.{g(A)}}
       c:b
\thatis
\formula p5|{..{..{f(b)}.{g(b)}}.{!\0.}}
\followswith p1"30"a.p5s6
\substituting p1 a:{f(b)}
            c:{g(b)}
            b:{!\0.}
\whichgives
\formula p5|{..{..{!\0-}-{f(b)}}.{g(b)}}
\named "59"
```

Figure 8: Short form notation for Fig. 3

inference. The first parameter is the position of the used formula number that is given as the second parameter. The fourth parameter is the position of the lines for the inference and their length is given by the last parameter. The type of inference is given by the third parameter. The whole macro stands for the following (invalid TEX) statement in the notation of the above-explained `\bcc`...`\ecc` macro: `\bcc{#1}/m{#2}:#3\rep{#4-#1-1}\*___\rep{#5}\*---\ecc`.

Finally, the macro `\named` assigns a number to the inferred formula.

Figure 3 of the inference for formula 59 shows that the sub-formula `*.a.{..{f(\da)}.{g(\da)}}` has to be entered two times. Besides the three basic commands I defined a fourth one: macro expansion. The token after a `!` is expanded as a *formula macro*, which might have up to three parameters. In Fig. 8 the macro `\0` is defined as the abbreviation for the above expression. To make it more flexible, a parameter for the first sign was added to the macro.

The output of the code is shown in Fig. 9, which should be identical to Fig. 3.

Parameters. The number of lines in a formula is defined by the parameter `\gfbsmaxanzahlzeilen`. The default is 25 lines, which is sufficient to typeset the Begriffsschrift. For each line two pairs of `\toks` and `\skip` registers are created: the first pair codes a single line of the Frege formula, the second a substitution in front of that line. A few flags are available in the macros. The first one is a flag that is not defined as an `\if`:

- `\nosubst` must be given before the macro package is loaded. The value '`t`' indicates that no registers for substitutions are needed, so the number of registers is reduced by 50%. (As a result, several commands are no longer usable, for example, `\substituting` is "turned off" and the command `\outof` becomes `\use`.)

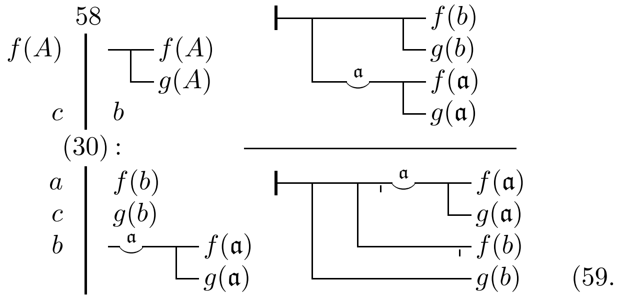


Figure 9: Output of the source of Fig. 8

- `\gfbslognotation` controls if the result of the short form is written to TEX's log file in an extended symbolic representation; details are in the next subsection. The default is *false*.
- `\gfbszeigestats` controls the output of some statistics about the maximum number of lines used in the formulas. The output is written to the log file and to the screen. The default is `\gfbszeigestatsfalse`.

Macros. The macros for the short form are much more complicated than those for the symbolic representation. They cannot be explained in greater detail in this article, but a few aspects are described; some of them might be well-known patterns.

The symbolic representation that is produced from the short form uses one `\toks` register for each line to store the coding. The registers carry the line number of the formula in their names. A static part of the name for the token registers is extended by the line number as a roman numeral (see [14, ex. 7.10]):

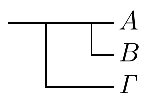
```
\csname
   gfbstoks\romannumeral\gfbszeile
\endcsname
```

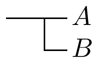
This code accesses a token register whose name is created with a `\csname` and `\endcsname` construction. If the counter `\gfbszeile` has the value 4 then the above code equals `\gfbstoksiv`. To add the element `\gfbselement` to the left of a token register the following method is used (see [14, p. 378]):

```
\edef\gfbstoken{\the\gfbselement
   \the\csname gfbstoks\romannumeral\gfbszeile
   \endcsname}
\global\csname gfbstoks\romannumeral\gfbszeile
\endcsname=\expandafter{\gfbstoken}%
```

For each line that is produced from the short form three values are stored:

a) the length of the line counted as those parts of the macro `\*` that contribute to the output;
b) the number of the line to which the current line is connected upwards;
c) the maximum line number that is connected directly or indirectly to the current line.

An example best shows how these numbers are used to build a formula. For example, 

is to be formed from the two formulas  and Γ. The first formula occupies lines 1 and 2, the second line 3. Then line 1 gets the three values $(2, 0, 2)$, line 2 $(2, 1, 2)$, and line 3 $(1, 0, 3)$. First the length of lines 1 and 3 are compared and the shorter line and all its "descendants" are filled with the abovementioned technique to add something to the left of a token register. Then the two lines are connected. So to join lines 1 and 3 with a condition the length of line 3 (the first value of the triple) must be increased to match the length of line 1 using one `\*---`. Next, at the left side of line 1 `\*-:-` is added. And all its dependent lines, i.e., line 2, get a `\*_!_`. Line 3 receives the matching `\*_'-`. The data for the lines is updated to the values $(3, 0, 3)$, $(3, 1, 2)$, and $(3, 1, 3)$, resp.

I had the (crazy?) idea to store the numbers in a single `\skip` register. The length is the normal part of the skip, the other two parameters are defined as the stretch and the shrink value.

```
\def\ggobble#1#2{\relax}
\def\setskip#1#2#3#4{%
  % #1: line no.; #2: no. of parts;
  % #3: line no. up; #4: max line no. down
  \global\csname
     gfbsskip\romannumeral #1%
  \endcsname=#2pt plus #3pt minus #4pt}
\def\getskip#1.#2 #3 #4.#5 #6 #7.#8{%
   \global\gfbsanzahl=#1% no. of parts
   \global\gfbshaengtan=#4% line no. up
   \global\gfbsgehtbis=#7% max no. down
   \ggobble}
```

The macro `\setskip` is called with the register number, i.e., the line number, and the three values to be stored. `\getskip` assigns the stored values of the `\skip` register to three named counters when called in this way:

```
\expandafter\getskip\the\csname
   gfbsskip\romannumeral\gfbszeile
\endcsname
```

The output of `\the` creates characters of the category "other" except for spaces. So a simple "pt" cannot be used in the parameter text (see [14, p. 375]). To delete the last "pt" the macro `\ggobble` is called. It deletes the next two tokens. In order to understand `\getskip`, the output of a well-known quantity, for example, `\bigskipamount`, should be studied.

Outputting the created symbolic representation in the log file of TeX is easy, as only the token registers must be written:

```
\wlog{\the\csname
      gfbstoks\romannumeral\gfbszeile
   \endcsname}
```

For other commands `\string` is used, for example:

```
\wlog{\string\bcc\string{#1\string}/m%
      \string{#2\string}:#3\string%
      \rep\string{\the\gfbsgrundeinzug%
      \string}\string\*___\string\rep%
      \string{#5\string}\string\*---%
      \string\ecc}
```

Finally, I want to write a few words about the size of the implementation. The macro package for the notation contains more than 1800 lines of code. It is produced from a WEB-like coding system. To make sure that all the macros produce the desired output 330 test cases have been coded.

The macros grew in several steps and not all problems might have the best solution. Nevertheless I hope I have accomplished my goals and the system is quite usable.

4 Extensions for the Grundgesetze

To reproduce the formulas used in Frege's Grundgesetze [7] some adaptation of the macros is required. Besides the abovementioned compact form, which is used throughout the Grundgesetze, the most obvious changes were the use of lines that have a uniform thickness and an increased set of inferences. Figure 10 shows an example of the style used in the Grundgesetze.

This is not the place to describe the necessary changes in detail, but a few comments are worthwhile. The command `\toggleGGstyle` switches between the style of the Begriffsschrift and the style of the Grundgesetze. It uses the flag `\gfbsuseGGstyle` (with the default setting `\gfbsuseGGstylefalse`) to activate the line thickness of the Grundgesetze.

As explained above, several `\dimen` registers are used to change the output of the symbolic representation. To change these registers into the style of the Grundgesetze a few values are defined. Now the control words start with `\gfgg`:

- `\gfggstrichdicke` represents the uniform line thickness; default is 0.58 pt.
- `\gfggraise` is the height of the content stroke; default is 0.14 ex.
- `\gfggneg` is the height of the negation indicator; default is 0.47 ex.
- `\gfgguht` is the height of the judgment stroke; default is 1.4 ex.
- `\gfggudp` is the depth of the judgment stroke; default is 0.9 ex.

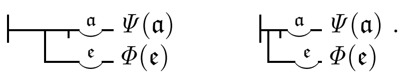
Compact form. Frege used a more compact form for the formulas in the Grundgesetze. (Figure 7 shows the compact form for the formulas of Fig. 2.) The short form is able to produce this form because it creates a kind of extended symbolic representation. The "extension" is the use of symbols that change their output. The selection process is complex and involves the `\*` macros to the right of the current position. Eleven more symbols are defined and nine of them can change. In the following list two forms are given for the new symbols: one for the normal output and one for the compact form.

- `0` represents negation that is either moved to the left ('`(`') or moved to the right ('`)`'): `0=()`;
- `1` represents affirmation: `1=^/`
- `2` again affirmation: `2=^+`;
- `4` represents optional content stroke: `4=-"`;
- `5` represents optional empty space: `5=_"`;
- `6` represents judgment strokes: `6=[]`;
- `7` represents a negated concavity: `7=',`;
- `8` again negation: `8=<~`;
- `9` represents a content stroke that has only one third of its usual length;
- `*` is a skip whose length is multiplied by 5/3;
- `$` is used only with a concavity and represents either a short content stroke or an empty space.

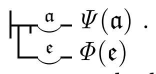
`\gfbskompakt` switches to the compact form. The default for the flag is *false*. The compact form does not always give perfect results, i.e., it doesn't match the original. Therefore some flags are defined that have to be invoked in a problematic formula.

- `\gfbskeinekompaktehoehlung` controls the use of the compact form for concavities. Its default value is false.
- `\gfbsaussagesichtbar` controls whether a content stroke can disappear completely in front of a statement. Its default value is also false.

Again examples might be useful to demonstrate the effects of the flags. On the left the non-compact form is given and on the right the same formula with `\gfbskompakttrue` is output:



The sign in the first line gives an asymmetrical result on the right. Next, here is the same formula with `\gfbskeinekompaktehoehlungtrue` as well as `\gfbskompakttrue`:



As a second example, we can look at the output for `\frege.{=.a}` and `\frege.{=.{=.a}}`. It is identical in the compact form: —a and —a. The activation of `\gfbsaussagesichtbartrue` shows a difference: –a —a.

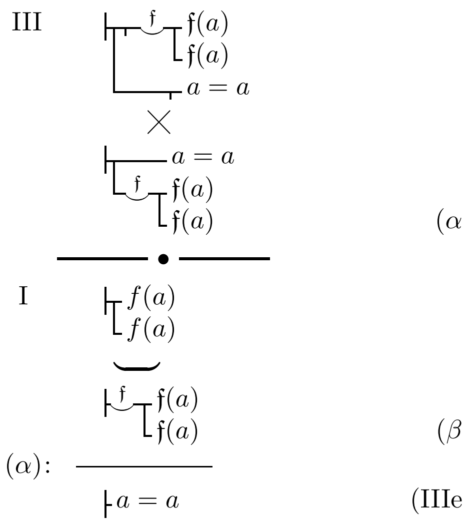


Figure 10: Inferences from the Grundgesetze [7], p. 66

New inferences. A flag is defined to control the use of the new inferences: `\gfggschlussweisetrue` allows the following construction in the formulas. If the first symbol in the `\*` macro is a question mark then the next two symbols are used to define the building blocks of an inference in the style of the *Grundgesetze*. Note that the thick line with the centered circle in Fig. 10 is only a separator for inference chains. It is coded as a normal macro.

The width of each of the two symbols after '`?`' is 50% larger then the usual width of a part in the macro `\*`. The following symbols are defined:

- `?` (must be the first in the triple) signals that the next two symbols build an inference line;
- `-` is a single horizontal stroke;
- `.` is a single (centered) period;
- `=` is a double stroke;
- `*` is a single stroke in the height of the upper line of the double stroke;
- `_` is an empty space;
- `"` is a skip;
- `>` has no output; the next symbol is a *transition-sign* (now the translation of technical terms follows [8]);
- `x` (must follow a `>`) represents the *contraposition*;
- `u` (must follow a `>`) represents a "quantification".

Now the eight transition-signs of the Grundgesetze can be typeset:

a) `\*?>x` gives $\times$;
b) `\*?>u` gives $\smile$;
c) `\*?--\*?--\*?--` gives ———— ;

d) `\*?==\*?==\*?==` gives ═══════ ;
e) `\*?-_\*?-_\*?-_` gives — — — ;
f) `\*?=_\*?=_\*?=_` gives ═ ═ ═ ;
g) `\*?=*\*?=*\*?=*` gives ═─═─═─ ;
h) `\*?.-\*?.-\*?.-` gives ·—·—·— .

Although I have not yet finished my copy of [7] I assume that this is the core that is required to typeset the formulas of the Grundgesetze. Of course, this work has special strings too. Its 29 symbols are different from those used in the Begriffsschrift; they all stay on a single line. Some symbols are available in a special font created by J. J. Green (see [12]).

References

[1] The Arché *Grundgesetze* Translation Project `http://www.st-andrews.ac.uk/~arche/projects/grundgesetze/grundgesetze.shtml` (accessed: 2014-11-29)

[2] Martin Davis, "Frege: From Breakthrough to Despair," in *The Universal Computer — The Road from Leibniz to Turing* (New York: W. W. Norton & Company, 2000), 41–58

[3] Encyclopædia Britannica, "Gottlob Frege" in Encyclopædia Britannica online, primary contributor: Michael A. E. Dummett. `http://www.britannica.com/EBchecked/topic/218763/Gottlob-Frege` (accessed: 2014-11-29)

[4] Gottlob Frege, *Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens* (Halle an der Saale: Louis Nebert, 1879)

[5] Gottlob Frege, "Begriffsschrift, a formula language, modeled upon that of arithmetic, for pure thought," in [24], 1–82; this translation of [4] was done by S. Bauer-Mengelberg

[6] Gottlob Frege, "Letter to Russell (1902)," in [24], 126–128; the translation was done by B. Woodward

[7] Gottlob Frege, *Grundgesetze der Arithmetik — begriffsschriftlich abgeleitet* (Jena: Hermann Pohle; Volume 1, 1893 Volume 2, 1903)

[8] Gottlob Frege, *Basic Laws of Arithmetic* (Oxford: Oxford Univ. Press, 2013); translation of [7] by Philip A. Ebert and Marcus Rossberg `http://www.frege.info/index.html` (accessed: 2014-11-29)

[9] Gottlob Frege, "Über den Zweck der Begriffsschrift", Suppl. zur "Jenaischen Zeitschrift für Naturwissenschaft" **16** (1882/83), 1–10

[10] Gottlob Frege, "Über die Begriffsschift des Herrn Peano und meine eigene", Berichte über die Verhandlungen der Königlich Sächsischen Gesellschaft der Wissenschaften zu Leipzig, Mathematisch-Physische Classe **48** (1896), 361–378

[11] J. J. Green, `bguq`, `http://ctan.org/pkg/bguq` (accessed: 2014-11-29)

[12] J. J. Green, `fge`, `http://ctan.org/pkg/fge` (accessed: 2014-11-29)

[13] J. J. Green, Marcus Rossberg and Philip A. Ebert, "The Convenience of the Typesetter; Notation and Typography in Frege's *Grundgesetze der Arithmetik*," Bull. Symbolic Logic, **21**:1 (2015), 15–30

[14] Donald E. Knuth, *The TEXbook*, Volume A of *Computers & Typesetting* (Boston: Addison-Wesley, 1984)

[15] Donald E. Knuth, *Computer Modern Typefaces*, Volume E of *Computers & Typesetting* (Boston: Addison-Wesley, 1986)

[16] Silvio Levy, "Using Greek Fonts with TEX," *TUGboat* **9**:1 (1988), 20–24 `http://tug.org/TUGboat/tb09-1/tb20levy.pdf` (accessed: 2014-11-29)

[17] R. MacInnis, J. McKinna, J. Parsons, and R. Dyckhoff, "A mechanised environment for Frege's Begriffsschrift notation," Workshop Mathematical User Interfaces 2004, Bialowieza (Poland)

[18] The MacTutor History of Mathematics archive. `http://www-history.mcs.st-andrews.ac.uk/Biographies/Frege.html` (accessed: 2014-11-29)

[19] W. V. Quine, "On Frege's Way Out," Mind New Series, Vol. 64, No. 254 (1955), pp. 145–159.

[20] Quirin Pamp, `frege.sty`, `http://ctan.org/pkg/frege` (accessed: 2014-11-29)

[21] Josh Parsons, `begriff.sty`, `http://ctan.org/pkg/begriff` (accessed: 2014-11-29)

[22] Marcus Rossberg, `grundgesetze.sty`, `http://ctan.org/pkg/grundgesetze` (accessed: 2014-11-29)

[23] Bertrand Russell, "Letter to Frege (1902)," in [24], 124–125; the translation was done by B. Woodward

[24] Jean van Heijenoort (ed.), *From Frege to Gödel: A Sourcebook in Mathematical Logic, 1879–1931*, (Cambridge, MA: Harvard University Press, 1967)

[25] Udo Wermuth, `GFnotation.tex`, `http://ctan.org/pkg/gfnotation` (accessed: 2015-09-22)

[26] Edward N. Zalta, "Gottlob Frege," The Stanford Encyclopedia of Philosophy (Fall 2014 Edition) `http://plato.stanford.edu/archives/fall2014/entries/frege` (accessed: 2014-11-29)

[27] Edward N. Zalta, "Frege's Theorem and Foundations for Arithmetic," The Stanford Encyclopedia of Philosophy (Summer 2014 Edition) `http://plato.stanford.edu/archives/sum2014/entries/frege-theorem` (accessed: 2014-11-29)

◇ Udo Wermuth
Babenhäuser Straße 6
63128 Dietzenbach
Germany
`u dot wermuth (at) icloud dot com`