# A mechanised environment for Frege's *Begriffsschrift* notation

Rob MacInnis[1], James McKinna[2], Josh Parsons[3] and Roy Dyckhoff[4]

[1]`rfm3@st-andrews.ac.uk`
[2]`james.mckinna@st-andrews.ac.uk`
[4]`rd@st-andrews.ac.uk`
School of Computer Science,
University of St Andrews
[3]`jp30@st-andrews.ac.uk`
*Arché* AHRB Research Centre for the
Study of Logic, Language, Mathematics and Mind,
University of St Andrews

**Abstract.** Frege's *Begriffschrift* [3] introduced new levels of sophistication and complexity in logical syntax and its representation; its two-dimensional nature has proved a stumbling block for those seeking to understand Frege's ideas and his system. It is, however, merely a form of abstract syntax for a higher-order predicate logic, with proofs represented linearly and formulae two-dimensionally. Our work concerns the development of a Java and XML-based GUI for the interactive construction of formulae (and, in due course, of proofs) of this system, with output in concrete form such as LaTeX. It is intended to make Frege's notation more easily used and understood, and to illustrate XML techniques on a seriously challenging and unusual problem. End users will be Frege scholars; we plan to make the system available in due course as a web-based application either publicly or in association with a publisher.

## 1 Introduction

Frege's *Begriffsschrift* [3] introduced new levels of sophistication and complexity in logical syntax and its representation, which have proved a stumbling block to subsequent generations of philosophers, mathematicians and logicians seeking to understand Frege's ideas.

The starting point for our work was the preparation of a new translation of the *Grundgesetze der Arithmetik* [4], with the intention of typesetting Frege's complex formalism using LaTeX, building on an initial style file produced by the third author [5]. There rapidly emerged the desirability of (semi-)automated support for the construction of the (LaTeX source for the) figures in Frege's notation, some 1000–1500 in all, rather than an attempt to typeset them by hand.

The first author was therefore employed to build a tool to support the graphical construction of the figures; a natural design choice was to factor the production of LaTeX source via an intermediate layer of XML. This paper documents some of the work done in producing a Java GUI for direct manipulation of Frege's notation, together with the XML and LaTeX back-end processing. Much of the latter is coming to seem routine, especially since the emergence of the MathML [9], OpenMath [1] and MKM [8] communities. Our work is distinguished in the first instance by not committing to the use of existing tools, but instead focusing on the unusual and challenging aspects of Frege's unusual and challenging notation.

## 2 Frege's notation

Frege's notation is merely a form of abstract syntax for a higher-order predicate logic, with proofs represented linearly and formulae two-dimensionally (in contrast partial or total to modern practice).

The notation in Frege's logic [7] contains, *inter alia*, all the expressive power of the modern predicate calculus. There are, however, two critical differences between them. Firstly, Frege's calculus is higher-order, allowing quantification over concepts (i.e. predicates or properties) rather than just over individuals, with predicates moreover represented as propositional functions. Secondly, and with more severe consequences for producers and consumers of the notation, Frege's system fails to use an abbreviation mechanisms for definition of logical connectives (although such a mechanism is used for other purposes). Table 1, adapted from Zalta [10], illustrates how the connectives are represented in Frege's notation—given the basic tree structure of an implication, and the negation decoration, the other connectives simply follow their standard definitions in terms of implication, negation and universal quantification. The lack of abbreviation, coupled with the two-dimensional layout, renders the notation essentially unreadable for all but simple formulae.

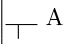
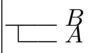
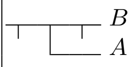
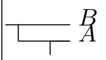
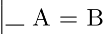
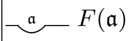
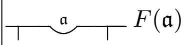
| Modern Notation | Frege's Notation |
|---|---|
| $\neg A$ |  |
| $A \supset B$ |  |
| $A \wedge B$ |  |
| $A \vee B$ |  |
| $A \equiv B$ |  |
| $\forall x.F(x)$ |  |
| $\exists x.F(x)$ |  |

**Table 1.** Logical connectives

Though the graphical part of Frege's notation is conceptually just abstract syntax, one may observe that the second dimension allows us to observe structural relationships that are only visible in modern notation via parenthesisation.

Figure 1 (on the following page) shows how the following formula in Peano-style notation appears in *Begriffsschrift*:

$$[\forall a(f(x,a) \supset F(a)) \supset (f(x,y) \supset F(y))] \supset$$
$$[\forall b[F(b) \supset \forall a(f(b,a) \supset F(a))] \supset [F(x) \supset (f(x,y) \supset F(y))]]$$

## 3 The Mechanised Environment

The system is a graphical editor for Frege's two-dimensional tree notation. Its initial purpose was to enable the creation of LaTeX source to represent the formulae and proofs in the new translation of *Grundgesetze der Arithmetik*[4] mentioned above. Subsequently, an intention to make the tool available to Frege scholars has influenced the system's development. It is written entirely in Java and includes three distinct modules: input interface, XML converter, and XML

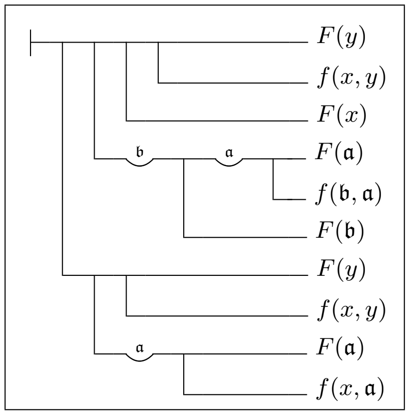


**Fig. 1.** Theorem 71 from *Begriffsschrift* (p. 59)

to LaTeX converter. As outlined in the following sections, this division leaves the system open to future development while fulfilling its current purpose in a timely manner.

A delicate issue in the eventual typesetting of *Begriffsschrift* is that leaf formulae are required to align on their initial symbols—see section 3.2 below. Output (as in Fig. 4 below) generated using the third author's initial set of LaTeX macros has the defect that formulae do not always align correctly. This appears to be an intrinsic limitation of the one-pass nature of LaTeX processing, influencing both the system architecture and the detailed design choices within each component.

## 3.1 User Interface

As an alternative to typesetting the LaTeX by hand, the user interface[1] provides all of the necessary tools to build and manipulate formulae and proofs in Frege's *Begriffsschrift* notation. The task of translation, which requires well over a thousand formulae to be constructed, has driven the design considerations for the GUI, as have typical HCI goals and expectations. Shortcut keys are included for experienced users, tool tips for beginners, and thoughtful colour selection aims to provide equal sensory feedback to colour-blind users.

[1] Functionality discussed has all been built but at the time of publication is still undergoing developmental changes.

Users will interact with the editor through a combination of mouse and keyboard commands. Though most editing actions can be performed with the mouse, more experienced users should find the shortcut keys to be a more fluid approach to construction. For mouse-based interaction, several visual cues are provided to aid in the construction process. When a user drags the pointer over an object in the tree it becomes highlighted, the insertion point for new tree objects is identified by a vertical line, and the *sub-tree in scope* (that part of the tree that will be affected by any changes) is highlighted as well. These features can be seen in the figures below.

Users can perform various actions upon the tree structure. In order to insert a negation, implication, or quantification, the user simply selects the corresponding tool, drags the pointer to the desired location and clicks. The tree is automatically realigned and ready for another action.

*Delete* actions can be applied to two types of object: a single tree and an entire sub-tree. The currently selected tree is highlighted in a colour different from that of the sub-tree in scope, making it very clear to the user what will be affected by a *delete* action. For deletion of a single tree the user simply selects the tree and presses 'delete' (or 'backspace'); for deletion of a sub-tree the user must either right-click on a tree and select 'delete sub-tree' or, alternatively, select the vertical implication line and press 'delete' as above[2].

Figures 2 and 3 show the *insert* and *delete* actions — note the tree realignment and the propagation of Gothic characters from the inserted universal quantifier through to the terminal nodes. (Abstraction of a formula $F(a)$ w.r.t. a free variable $a$ forces the conversion of $a$ to a Gothic font.) For example, the final formula in the two cases is representable in (fully parenthesised) modern notation (and using sets to represent concepts and membership thereof to represent 'falling under' a concept) as

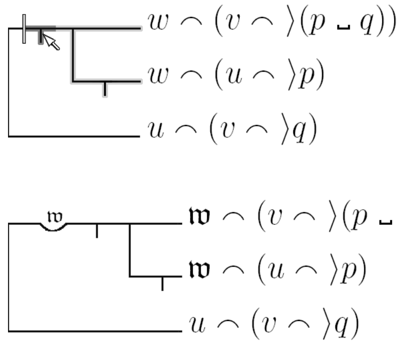
$$(\forall x.((x \epsilon u) \supset \forall y.((y \epsilon u) \supset (x = y)))) \supset (Nu = 1)$$

or even as

[2] The latter mechanism is provided as a faster way to prune trees and should not be done by accident if visual cues are followed. Accidents, of course, do happen, and an undo mechanism is being developed.

$$\forall x, y \in u.(x = y) \supset Nu = 1$$

expressing the idea (in modern terminology) that a set $u$ has cardinality 1 if any two of its elements are equal. (Whether this is true is another matter.)



$w \frown (v \frown \rangle(p ␣ q))$
$w \frown (u \frown \rangle p)$
$u \frown (v \frown \rangle q)$
$\mathfrak{w}$
$\mathfrak{w} \frown (v \frown \rangle(p ␣$
$\mathfrak{w} \frown (u \frown \rangle p)$
$u \frown (v \frown \rangle q)$

**Fig. 2.** Before and after an *insert* action.

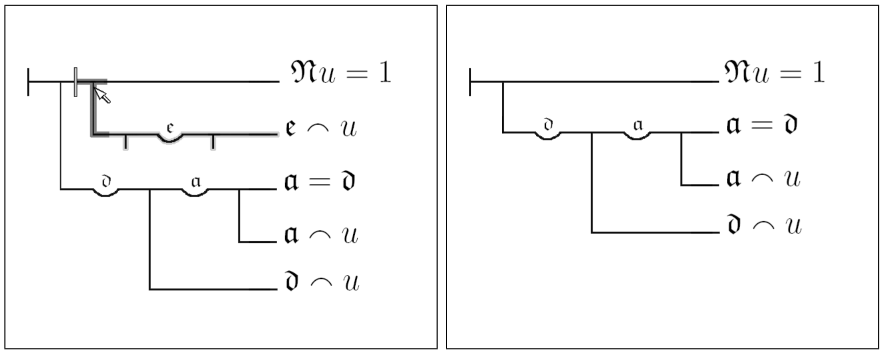


**Fig. 3.** Before and after a *delete* action.

The ability to use multiple editing windows was added after the initial development. This is particularly useful when making use of the copy-and-paste functionality, which saves construction time by allowing similar tree sub-structures to be reused. In order to copy a user must simply drag over the desired sub-tree, right-click, and select ‘copy.’ Paste always creates a new implication line and is similar to insertion: point, right-click, select ‘paste,’ and the tree will appear at the insertion point.

## 3.2 Alignment Algorithm

Correct alignment of the formulae is a primary concern for the GUI and its generated output, and a key requirement which the system attempts to fulfil. As alluded to above regarding the one-ass nature of LaTeX processing, there appeared no simple solution to left-align the leading symbols of the statements of a formula as required in Frege’s notation.

The original alignment produced mis-aligned trees like Fig. 4, whereas they should be aligned as in Fig. 1. While the improperly aligned figure may, in this case, be more informative as regards scope, it is simply not “official” *Begriffsschrift*.

This has the further consequence that the notation is particularly ill-suited to dynamic editing: any changes to the tree affect (the alignment of) all branches *every time*. Because of this it was imperative to develop an algorithm that efficiently restructured trees of any size. The final algorithm—O($n$) for both time & space complexity[3]—quickly and accurately aligns the tree after every edit.

Every branch includes a spacer of initial length zero. Starting at the root (the ‘$\vdash$’ symbol) and working top to bottom, the algorithm calculates the distance from the beginning of the branch to the beginning of the statement. This value is passed as the horizontal ‘guess’, along with the vertical height of its statement, to the next(rightmost) branch[4]. This branch calculates its own length, compares it to the ‘guess’, and passes on the greater of the two (note that it does not resize its spacer at this point). Upon reaching the bottom of the tree, each branch knows its exact height location but only the bottom-most branch knows the maximum width. A quick pass back upwards through the tree informs all branches of their maximum width; the tricky bit here, however, is that each branch doesn’t

[3] $n$ being the number of branches in the formula
[4] Note that this is not the abstract *next* branch in logical terms.

know the horizontal distance between itself and the previous branch. This is where the ‘guess’ comes in: after each branch resizes it simply returns the difference between its current length and the ‘guess’ value, effectively telling the previous branch the only information it needs: how big to make its spacer. The widest branch receives a ‘correction’ of zero.

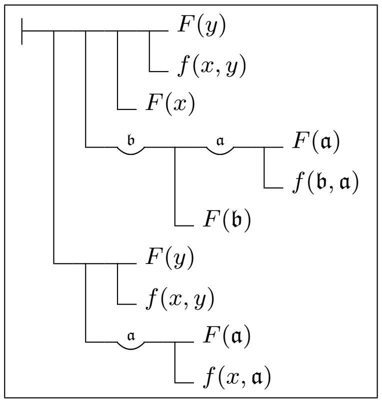


**Fig. 4.** Misaligned theorem 71 from *Begriffsschrift* (p. 59)

### 3.3 XML Layer

The intermediate layer of XML is a simple, structured method of abstractly representing Frege’s formulae and proofs. Saving graphical objects in this XML format enables simple batch conversion into LaTeX while leaving open the opportunity to create converters for other formats such as MathML. XML offers genericity, interoperability and a rich powerful toolset.

The underlying XML format is a direct representation of Frege’s tree structure, together a number of relative spacing measurements—the spacing data produced by the GUI’s alignment algorithm is simply exported along with abstract tree structure in order to generate properly aligned LaTeX. This represents the middle ground between purely abstract representation and a concrete rendering format, improving upon previous solutions to the problem.

### 3.4 LaTeX Generator

The main purpose of the system is to achieve figures in processed LaTeX identical to those constructed in the GUI. The XML to LaTeX converter provides a simple mechanism for converting our Frege XML format into LaTeX source for the suitably modified version[5] of the original *Begriffsschrift* style file [5]. All of the correctly aligned sample figures in this paper have been constructed using the system, thus providing a first proof of concept before work begins in earnest on the Grundgesetze.

While the macros in the *Begriffsschrift* style file are quite complex, the actual conversion between the XML and LaTeX is relatively straight-forward. Initially a decision had to be made about whether to use a DOM or SAX parser, and this raised some deeper issues about the language itself and how to appropriately implement rules for data. Several attempts to produce a correct schema were put on hold for various reasons. Given the evolving nature of the current system design, a generic DOM tree implementation has been employed.

The XML is parsed by the DOM parser and the Document object is created. This object is passed to the tree-walker which uses a very simple look-up table to turn each element of the XML into the corresponding source for that that macro instance. This is very clean and very efficient, and also very simple. It is easily extensible because the look-up table is loaded from a text file at run-time, so any new changes in the XML that are ready to be used by the style file can be translated without updating the software. The same technique is used for converting symbols specific to Frege's notation into their respective macros.

A particularly unusual feature, one which ties the software directly to its task of producing LaTeX, is the ability for expert users to type macros directly into terminal nodes rather than using the mouse or shortcut keys. This functionality was desired by the translation team, and actually has required no additional code. Because there is no proof-checking involved in the construction process, users can put whatever they like at the terminal nodes. Frege-specific symbols are encoded in XML and converted to their specific macros later,

[5] `begriffnew.sty`, available from the authors

but any surrounding text is simply carried into the LaTeX source and compiled like any other document.

## 4 Conclusions and Further Work

The system has been designed to a still-evolving specification; whether it is indeed easier than typesetting the proofs by hand has yet to be seen. There are many directions in which this project has room to grow and our aims have certainly gone beyond those originally conceived by the translators of the *Grundgesetze*.

Compatibility with other representations such as SVG (*Scalable Vector Graphics*) [2] and MathML [9] is particularly desirable and is the most likely next step. As suggested by a referee, the work of Padovani and Solmi [6] may be of interest. For researchers and students eager to learn by experimentation, the ability to enter formulae and have them built in Frege's notation automatically may also be useful. Authors of papers on Frege will hopefully find this package useful for creation of any necessary formulae, rather than attempting a reconstruction from scratch.

Certainly there are many other possible applications for both this system as a whole and also its constituent parts. Where they will fit into the broader mathematical community and which parts will be developed further remain to be seen. The initial translation task has given rise to a complex tree editing problem which this system addresses, and its architecture leaves open the possibility for future development in a variety of directions.

## 5 Acknowledgments

We acknowledge the helpful comments of two referees, the advice of Roy Cook, Philip Ebert and Marcus Rossberg on the work and notation of Frege, Peter Sullivan for a preprint of [7] and all of the University of St Andrews, the EPSRC and the *Arché* project (and thus indirectly the Arts and Humanities Research Board) for financial support.

## References

1. The OpenMath Consortium. OpenMath 2.0. `http://www.openmath.org/cocoon/openmath/standard/om20/index.html`.
2. The W3C Consortium. Scalable vector graphics (svg). `http://www.w3.org/Graphics/SVG/`.
3. Gottlob Frege. *Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens.* Halle: L. Nebert, 1879.
4. Gottlob Frege. *Grundgesetze der Arithmetik, begriffsschriftlich abgeleitet.* Pohle: Jena, 1892.
5. Josh Parsons. `begriff.sty` a LaTeX2e package for typesetting Begriffsschrift. `http://arche-wiki.st-and.ac.uk/~{}ahwiki/bin/view/Main/BegriffsschriftLaTeX`, 2003. Released under the GNU General Public License.
6. Luca Padovani & Riccardo Solmi. An investigation on the dynamics of direct-manipulation editors for mathematics. In *Third International Conference on Mathematical Knowledge Mnagement, LNCS 3119.* Springer, 2004.
7. Peter Sullivan. Frege's logic. In Dov M. Gabbay and John Woods, editors, *Handbook of the History of Logic*, pages 659–750. Elsevier BV, 2004.
8. The MKMNET Consortium. `http://monet.nag.co.uk/mkm/consortium.html`.
9. W3C. MathML Technical Recommendation. `http://www.w3.org/TR/MathML2/`.
10. Edward N. Zalta. Gottlob Frege. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy.* 2004.