The De Bruijn Factor

Freek Wiedijk
<freek@cs.kun.nl>

Department of Computer Science
Nijmegen University
P.O. Box 9010
6500 GL Nijmegen
The Netherlands

Abstract. We study de Bruijn's 'loss factor' between the size of an ordinary mathematical exposition and its full formal translation inside a computer. This factor is determined by a combination of the amount of detail present in the original text and the expressivity of the system used to do the formalization. For three specific examples this factor turns out to be approximately equal to *four*.

1 Loss Factor

In 'A survey of the project Automath' de Bruijn wrote (p. 160 in section A.5 of [9] which is a reprint from [1]):

> *A very important thing that can be concluded from all writing experiments is the* constancy of the loss factor. *The loss factor expresses what we loose in shortness when translating very meticulous 'ordinary' mathematics into Automath. This factor may be quite big, something like 10 or 20, but it is constant: it does not increase if we go further in the book. It would not be too hard to push the constant factor down by efficient abbreviations.*

Here* we briefly study this loss factor, which we call the *de Bruijn factor*.

When writing a 'formal proof' (a proof that is entered in a computer in full detail in such a way that the computer can check the correctness) there are basically two approaches:

- One takes an existing, non-formal, mathematical text and translates it – more or less faithfully – into a computer representation.
- One 'programs' the proof directly into the formal system, without first creating a 'natural language' counterpart.

The first method has the advantage that the formalization automatically will be well documented, and also it generally seems to be easier to translate a pre-existing text than to think about the proof and about the mechanics of the formalization at the same time. The de Bruijn factor of course only can be objectively measured for a formalization of the first kind.

The de Bruijn factor of a formalization depends on two aspects. On the one hand there is the level of detail in the original, which depends on the 'character' of the text that is being translated. There exist a wide range of mathematical styles, which each have their own level of precision at which the proofs are elaborated. For instance there are:

- Books that give a detailed development of a subject for foundational purposes, like Whitehead and Russell's *Principia Mathematica* [10].

* The full Automath, Mizar and TEX files that are discussed here can be found on the World Wide Web at the address <http://www.cs.kun.nl/~freek/factor/>.

– Ancient mathematics, like Euclid's *Elements* [4].
– Textbooks for education.
– Handbooks about specific mathematical subjects.
– Papers in computer science that have a strong mathematical flavor.
– Mathematical research papers.

The three cases studied in this note are respectively of the first, fifth and fourth kinds.

On the other hand there is the system that is being used. Some systems have more automation and more (as de Bruijn called it) 'efficient abbreviations' than others. So the de Bruijn factor measures how efficient a system is. One might imagine a 'benchmark' for proof assistants consisting of a number of mathematical texts in various styles to be represented. However, the technology currently seems not to be well-developed enough to already put together such a benchmark.

2 Apparent and Intrinsic De Bruijn Factors

The size in bytes of the files of a formalization is not a very meaningful measure. It depends on such matters as the choice of variable names and the use of whitespace: these factors don't seem to mean much for the contents of the files. For instance if indentation is done with tab characters that part of the file will be eight times as small as when it's done with space characters: but the file will look the same in both cases. As another example, the TEX macro name '`\Leftrightarrow`' for the '⇔' symbol uses 15 characters, while an encoding like '`<=>`' uses only 3.

To compensate for these effects it seems natural to 'squeeze out' trivial redundancy by compressing the files before calculating the ratios of their sizes. In fact, use of the tab character can be seen as a crude way of compressing long runs of spaces.

We will call the ratio of the uncompressed file sizes the *apparent* de Bruijn factor, and that of the compressed file sizes the *intrinsic* de Bruijn factor.

If one uses the intrinsic de Bruijn factor, it isn't useful any more to remove the white space from the computer representation of a proof to get a better ratio, because this kind of optimization only has a minor effect on the size of the compressed file.

Surprisingly it turns out that generally both the 'natural language' and the 'computer' versions of a proof compress similarly well. This means that the apparent and intrinsic de Bruijn factors turn out to be approximately the same.

3 Arithmetic in Automath

The first example for which we will calculate the de Bruijn factor is Jutting's classic Automath translation (see section D.2 of [9], a reprint from [6]) of Landau's *Grundlagen der Analysis* [7], a cute little book about the basic laws of arithmetic up to the complex numbers.

To give an impression of the text and its translation, here is a small fragment of a proof (of 'Satz 27' on p. 37 of [7]) in the *Grundlagen*:

> *1 gehört zu* $\mathfrak{M}$ *nach Satz 24. Nicht jedes* x *gehört zu* $\mathfrak{M}$*; denn für jedes* y *aus* $\mathfrak{N}$ *gehört* $y+1$ *nicht zu* $\mathfrak{M}$*, wegen*
>
> $$y + 1 > y.$$

together with its rendering in the AUT-QE dialect of the Automath language:

```
s@[n:nat]
t1:=[x:<n>p]satz24a(n):lbprop(1,n)
s@t2:=[x:nat]t1(x):lb(1)
[l:[x:nat]lb(x)][y:nat][yp:<y>p]
```

```
t3:=satz18(y,1):more(pl(y,1),y)
t4:=satz10g(pl(y,1),y,t3):not(lessis(pl(y,1),y))
t5:=th4"l.imp"(<y>p,lessis(pl(y,1),y),yp,t4):not(lbprop(pl(y,1),y))
t6:=th1"l.all"(nat,[x:nat]lbprop(pl(y,1),x),y,t5):not(lb(pl(y,1)))
t7:=mp(lb(pl(y,1)),con,<pl(y,1)>l,t6):con
l@t8:=someapp(nat,p,s,con,[x:nat][y:<x>p]t7(x,y)):con
```

Note the reference to 'Satz 24' at the start of both versions of the fragment.

The sizes of both the TEX and Automath versions of this book, both uncompressed and compressed (using the Unix `gzip` utility), and the corresponding de Bruijn factors are given in the following table:

	informal	formal	*de Bruijn factor*	
uncompressed	189 K	736 K	apparent	3.9
compressed	42 K	155 K	intrinsic	3.7

Apparently the de Bruijn factor of Automath for this kind of text is slightly less than *four.*

4 Computer Science in Mizar

The second example that we will calculate the de Bruijn factor for, is a section from a paper about 'finite topology' (pp. 12–17 of [8]), really a mathematical development of the theory of digital filtering of one-bit images. The translation [5] is in Mizar, which is a more accessible and more high level system than Automath.

As an example, here's a fragment of the proof of 'Theorem 2.1' (p. 16 of [8]):

> *Let B and C be non-void subsets of A such that $B \cap C = \emptyset$ and $B^b \cap C = \emptyset$. Then, there exists an element x in B, and we can construct a set P_n as a procedure described previously and $P_{n+1} = P_n$.*

with as Mizar translation:

```
given B, C being Subset of the carrier of FT such that
  A26:A = B U C and
  A27:B <> ∅ and
  A28:C <> ∅ and
  A29:B ∩ C = ∅ and
  A30:B^b ∩ C = ∅;
A31: B c= B^b by Th18 ;
A32: B^b ∩ A = B^b ∩ B U ∅ by A26,A30, BOOLE:70
           .= B^b ∩ B by BOOLE:60
           .= B by BOOLE:42, A31 ;
consider x being Element of B ;
x ∈ A by A26, BOOLE:def 2,A27 ;
then consider S being FinSequence of bool the carrier of FT such that
  A33:len S > 0 and
  A34:π(S,1) = {x} and
  A35:for i being Nat st i > 0 & i < len S holds π(S,i+1) = π(S,i)^b ∩ A and
  A36:A c= π(S,len S) by  A25 ;
```

Note the close syntactical similarity of the requirements on B and C in both versions of the text.

Here are the statistics of this example:

	informal	formal	*de Bruijn factor*	
uncompressed	7.7 K	35.3 K	apparent	4.6
compressed	2.6 K	8.0 K	intrinsic	3.1

So the de Bruijn factor of this text is not much better than that of the previous one. An explanation might be that the paper that is translated contains much less detail than the *Grundlagen* book, so the extra power of Mizar is compensated for by the more loose style of the informal article. For instance, a number of statements at the end are not proved, but instead it is just stated that:

> *The following facts are easily derived.*

5 Mathematics in Mizar

The final example for which we calculate the de Bruijn factor, is part of an ongoing effort at the Mizar project to translate a complete mathematical book [3] into the Mizar language. For the book a handbook from mathematical logic was chosen, which presents the theory of 'continuous lattices.' The translation of this book (which currently is halfway finished) consists of a large number of Mizar articles with names starting with '`YELLOW`' and '`WAYBEL`'.

The article that we analyze here [2] is the translation of four pages of the book. Again we give an example of the style of both the original and the translation. The statement of 'Corollary 1.13' (on p. 106 of [3]) is:

> *If L is a continuous lattice, then $(L, \sigma(L))$ is a quasicompact and locally quasicompact sober space. In particular, $(L, \sigma(L))$ is a Baire space.*

which gets translated into Mizar as:

```
L is continuous implies L is compact locally-compact sober Baire
```

Then the proof of this 'Corollary' starts with the following reasoning:

> *We have to show that a point $x \in L$ has a basis of quasicompact neighborhoods. By 1.10 the sets $\uparrow y$ with $y \ll x$ form a basis for the neighborhoods of the point. But as we know, if $x \in U \in \sigma(L)$, then actually we have a $y \in U$ with $y \ll x$; hence, $\uparrow y \subseteq U$, and so the sets $\uparrow y$ can be used as neighborhoods.*

to which corresponds the following fragment of the Mizar proof:

```
 thus A5: L is locally-compact
 proof let x be Point of L, X be Subset of L such that
A6: x ∈ X and
A7: X is open;
   reconsider x' = x as Element of L by STRUCT_0:def 2;
   set bas = { wayabove q where q is Element of L: q << x' };
A8: bas is basis of x by A1, WAYBEL11:44;
   consider y being Element of L such that
A9: y << x' & y ∈ X by A1, A6, A7, WAYBEL11:43;
   X is upper by A7, WAYBEL11:def 4; then
A10: uparrow y c= X by A9, WAYBEL11:42;
   set Y = uparrow y;
 take Y;
```

```
    wayabove y ∈ bas by A9; then
A11: wayabove y is open & x ∈ wayabove y by A8, YELLOW_8:21;
    wayabove y c= Y by WAYBEL_3:11; then
    wayabove y c= Int Y by A11, TOPS_1:56;
 hence x ∈ Int Y by A11;
 thus Y c= X by A10;
```

Because this book is 'more mathematical' and hence reasons at a higher level than the previous two examples, the de Bruijn factor is a bit higher:

	informal	formal	*de Bruijn factor*	
uncompressed	11.7 K	78.4 K	apparent	6.7
compressed	4.0 K	16.3 K	intrinsic	4.1

6 The De Bruijn Threshold

It seems plausible that there is a certain value for the de Bruijn factor such that, when proof checkers become sufficiently powerful that their factor drops below it, people will start using them for serious work (like verifying the correctness of their mathematics and communicating the precise details of their work to others). We suggest to call this value the *de Bruijn threshold.* Like the de Bruijn factor of a system, it probably depends on the kind of mathematics.

As it probably is not possible to get a formalization to be as short as its informal version, it is to be hoped that the de Bruijn threshold of something interesting won't be less than *one.*

References

[1] N.G. de Bruijn. A survey of the project Automath. In J.P. Seldin and J.R. Hindley, editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 579–606. Academic Press, New York, London, 1980.

[2] C. Bylinski and P. Rudnicki. The Scott topology, Part II. *Journal of Formalized Mathematics*, 9, 1997. MML Identifier: `WAYBEL14`.

[3] G. Gierz, K.H. Hofmann, K. Keimel, J.D. Lawson, M. Mislove, and D.S. Scott. *A Compendium of Continuous Lattices.* Springer-Verlag, Berlin, Heidelberg, New York, 1980.

[4] Sir Th.L. Heath. *The Thirteen Books of Euclid's Elements.* Dover Publications, New York, dover edition, 1956. First edition 1908, second edition 1925.

[5] H. Imura and M. Eguchi. Finite Topological Spaces. *Journal of Formalized Mathematics*, 4, 1992. MML Identifier: `FIN_TOPO`.

[6] L.S. van Benthem Jutting. *Checking Landau's "Grundlagen" in the Automath system.* Number 83 in Mathematical Centre Tracts. Mathematisch Centrum, Amsterdam, 1979.

[7] E. Landau. *Grundlagen der Analysis.* Chelsea Publishing Company, New York, fourth edition, 1965. First edition 1930.

[8] Y. Nakamura, Y. Fuwa, and H. Imura. A Theory of Finite Topology and Image Processing. *Journal of the Faculty of Engineering, Shinshu University*, 69:11–24, 1991.

[9] R.P. Nederpelt, J.H. Geuvers, and R.C. de Vrijer. *Selected Papers on Automath*, volume 133 of *Studies in Logic and the Foundations of Mathematics.* Elsevier Science, Amsterdam, etc., 1994.

[10] A.N. Whitehead and B. Russell. *Principia Mathematica.* Cambridge University Press, Cambridge, paperback edition, 1962. First edition 1910, second edition 1927.