Peano Commutativity in [S′]

Martin Røpcke

1. juli 2005

Indhold

1 Introduction

This report is turned in for credit for the course “Logic” at DIKU summer 2005. In short, the assignment is to prove the commutativity of addition within peano arithmetic using the Logiweb system developed by Klaus Grue. Logiweb will be introduced shortly. That is, in this report we shall carry out a proof of the proposition that for all natural number the following holds:

$$\mathrm{t} + \mathrm{r} = \mathrm{r} + \mathrm{t}, \tag{1}$$

and we shall use Klaus Grue’s Logiweb system to check our proof. Cf. [?]. However, we use [$\dot{\mathsf{x}}$] and [$\dot{\mathsf{y}}$] as our variables.
During the course we have been introduced to mathematical logic using the text book *Introduction to Mathematical Logic* by Elliot Mendelson, cf. [?]. In order to get the full of this report we thus assume an elementary knowledge of logic as it is introduced in this book. However, we do give short introductions to some basic topics—Logiweb as well as logic—making this report selfcontained—at least to a graduate student of computer science at DIKU. The report has been written from June 23rd to July 3rd. Due to this short period of time the proof is carried out as simpel as possible discarding any fancy solutions even before they have been fully conceived of. The commutativity of addition is part of Proposition 3.2 in [?], p. 156, which is proved immediately afterwards on pp. 157f. We follow this proof backchaining from the actual proof of (1) through other results of the proposition and necessary lemmas and theorems contained in the book. Besides Mendelson’s propositions we only make use of two tautologies and one lemma in the final proof.
We hope our intentions have been reached and that this report carries just a little of the fun it has been working with Logiweb. As Klaus always invites the Logiweb user to we do also: “Have fun”.

2 The Logiweb System

In this section we give a short introduction to the Logiweb system, a web-based system for distribution of mathematical definitions, lemmas, and proofs writing web-pages. Cf. [?]. For further details and elaboration on the myriad of intricacies and topics concerning the system we refer to Klaus Grue’s introduction to Logiweb, namely the check page. Alternatively, we recommend the peano page introducing peano arithmetic, which is displays well the system “at work”. In order to obtain a thorough knowledge of the system one is adviced to follow the course “Logic” or read the base page (cf. [?]), but probably the best choice would be to devour the exercise of writing a page proving some proposition of choice chosen from [?] using Logiweb. After such an exercise any logiweb work and the hardship of doing so will most likely depend on the level of mathematics one is working with.

2.1 References

As hinted at above Logiweb is a system in which users world wide can distribute their own mathematical work and reuse the work of others. Mathematics in Logiweb is publicized by submitting a *logiweb page* to the Logiweb system, that is, to a *Logiweb server*. Abstractly speaking, in a logiweb page one can define a mathematical theory or system[1] and a language of this system within which one can then define, state, and prove mathematical lemmas, theorems, propositions, etc. A theory must defined in one page. The proofs within a theory will be proof checked upon submission (actually they will be proof checked when compiling the page locally) to the Logiweb system and one then knows if the contents of a page are correct within the theory. A page can refer to other logiweb pages in which a needed theory, results, functionalities etc. are defined. Thus, Logiweb works much like the world wide web where pages refer to each other using hyperlinks. The mesh of Logiweb servers translate Logiweb references to http urls making sure a page can find its references and that they together form a directed acyclic graph from top (the base) to bottom (a logiweb page).
Essentially, the Logiweb reference of a page is global hash key computed on the basis of the contents of the specific page. This ensures that each page has a unique key, that is, reference. The reference of this page in the special *kana*[2] format is:

```
nani
sika nuta tita nate kesa  suke nesu nute tese kaku
nuki teni sike kute nenu  keti kanu sane suke suta
sasu saki seni keni kuke  sati sika nasa natu
```

`Kana` is a format developed for Logiweb making it possible to speak or pronounce a reference. This is done because Logiweb has been developed with the possibility of using a microphone as input tool in mind.
Also, one can refer to this page using the well-known http url:

http://www.diku.dk/ grue/logiweb/20050502/home/
mrmr/peano-commutativity/latest/vector/page.lgw

Two last ways to refer to a page is using the reference expressed in so-called *mixed endian hexadecimal* or using a decimal reference. Cf. [**?**] and the `Reference` section in the source or `pyk` code in a Logiweb page. The reference section—known as the `BIBLIOGRAPHY`—of a Logiweb page will be introduced shortly.

2.2 The Base Page

Now, in the previous section we mentioned *the base* of the mesh of Logiweb pages. This is a certain part of Logiweb, which implements the basic and

[1]We shall use *theory* and *system* interchangeably, as is the case with *Logiweb system* and *Logiweb*.

[2]For details on kana we refer to `http://yoa.dk/logiweb/doc/misc/kana.html`.

necessary functionalities of the mathematical part of the Logiweb system. That is, the *base page*—as it is denoted—defines the proof checker and most of what makes the mathematics of Logiweb possible. Actually, there is possibly more than one base page, but the one Klaus Grue has written will probably be *the* base page to begin with. But one is free to write a base page making up a different (or a similar) system. A base page is a page without references to other pages and thus form the base of a mathematical system within Logiweb. For more on this we refer to the base page, Section 1.4 in particular.
We now know the necessary basics of the Logiweb system. Next, we take a closer look at the structure of a Logiweb page.

3 A Logiweb Page

In this section we shortly describe the most important details of a Logiweb page. That is, we introduce the Logiweb page, the language used in Logiweb, and the predefined structure all pages must be written according to. This introduction is adequate for getting started with Logiweb, but is in no manner satisfactory to the serious user of the Logiweb system.

3.1 "Main Menu"

In the Logiweb system each page has a so-called *main menu*, e.g., the main menu of this page is at:

`http://www.diku.dk/~grue/logiweb/20050502/home/mrmr/peano-commutativity/latest/`

From the main menu one can browse and find numerous kinds of information about the page. The most important of which to begin with is the page in `pdf` format and (for Logiweb newbie) the source code.
A Logiweb page is written in mixture of LaTeX and the language `pyk` developed by Klaus Grue. `pyk` is then the language in which the mathematics to be checked by a computer of a Logiweb page is written. The best way to get a feeling for this mixture is to read the source code of a Logiweb page. This is found under `Source` → `Actual Source` from the Logiweb main menu of a Logiweb page, e.g., the check page.
One can also read the `pyk` text of a page, which is found under `Body` → `Pyk` also at the Logiweb main menu. It is also under `Body` that one finds the TeX and pdf files of the page.[3]

3.2 The Pyk Language

As mentioned in the previous section the `pyk` language has been developed for easy pronounciation or in generel for "spoken mathematics", cf. the base page,

[3]The TeX files are not available through a browser due to safety precautions.

Section 1.1. Thus, a mathematical expression as we are well-acquaintated with such as:

$$(x + y)^2 = x^2 + 2xy + y^2$$

may be the equivalent of the following `pyk` expression:

> parenthesis var x plus var y end parenthesis square equals var x square plus two times var x times var y plus var y square[4]

Thus, in time one can "tell" one's mathematics to a computer and it will generate a file written in `pyk`, which will then be the source for precisely one Logiweb page. This is also the reason why `pyk` is case sensitive, that is, the `pyk` code cannot be in capital letters.
We thus see that `pyk` is much like any other programming language. However, in `pyk` one writes everything out in words instead of using symbols, e.g., "parenthesis" and "end parenthesis" instead of "(" and ")". In pdf format the textual representations will be replaced by their symbolic representations defined by the author of a Logiweb page. We shall return to this *aspect* of Logiweb in Section **??**.

3.3 The Structure of a Logiweb Page

A Logiweb page has to be written according to certain standards. Thus, a specific structural sketch for a Logiweb page has to used, which defines the specific and necessary parts of a Logiweb source file, that is, a `pyk` file. A `pyk` file has exactly one of each of these parts or sections, which will be described below. We note that the section names must be in capital letters. Like the TEX code and normal text written this is not part of `pyk` and does not need to be in small letters.

`PAGE` A Logiweb page has a name which is given in the `PAGE` section of the `pyk` file. For instance, in the source code for this page one will find the PAGE section at the very top where it is seen that this page is named "peano commutativity".

`BIBLIOGRAPHY` The references of a Logiweb page are given in the `BIBLIOGRAPHY` section of the `pyk` file. A page can refer to every possible Logiweb page, also if one of these pages refer to once referenced pages. The Logiweb server will make sure no cyclic references occur.

`ASSOCIATIVITY` In the `ASSOCIATIVITY` section of the `pyk` file the constructs of a logiweb page and their associativity—either POST- or PRE- are the options— are declared. The order of the constructs furthermore define their priority. Constructs declared with comma separation after the same associativity statement have same priority. (In a sense the *constructs* of a

[4] This example has been taken from [?].