The Layers of Logiweb

K. Grue[1]

DIKU, Universitetsparken 1, DK-2100 Copenhagen
`grue@diku.dk`, http://logiweb.eu/, http://www.diku.dk/˜grue/

Abstract. Logiweb is an open source, distributed system for publication of machine checked mathematics. It covers all aspects of electronic publishing: high typographical quality, archival, handling of references to previously published results, and publication of refereed volumes. The present paper is itself produced using Logiweb; and the paper is formally correct in the sense that it has been verified by Logiweb. The paper describes the implementation layers of the Logiweb system as seen by the user: the programming layer, the metalogic layer, the tactic layer, and the object proof layer.

1 Introduction

Logiweb [8,9] is a system for authoring, storing, distributing, indexing, checking, and rendering of "Logiweb pages". Logiweb pages may contain mathematical definitions, conjectures, lemmas, proofs, theories, journal papers, computer programs, and proof checkers.

The main features of Logiweb are that Logiweb offers authors unlimited notational freedom, offers unlimited choice of what kind of logic authors may use, allows authors to reference definitions and lemmas web-published by other authors, and still is able to verify the formal correctness of papers.

From the point of view of an author, Logiweb resembles Mizar [19,16] and TEX [12]: One prepares a source text `mypaper.pyk` with a text editor and runs a command like `pyk mypaper` to get it checked, rendered, and published.

One difference from Mizar is that, on Logiweb, one has to see the result in a web browser (even during the writing of papers where one typically does not web-publish each iteration). Papers are typically rendered in PDF using TEX. Another difference is that one may reference any Logiweb paper published on the Internet. A third difference is that Logiweb has no central authority: anybody can publish anything on the system as easily as one can publish pages on the World Wide Web. A fourth difference is that Logiweb is more resource demanding than Mizar and requires a modern PC to run smoothly.

Yet another difference from Mizar is that Mizar defines a syntax for source files whereas Logiweb leaves the definition of that to the user. This allows authors to write in a spoken math style like "limit as x tends to infinity of 1 over x" as in EzMath [17]. Actually, the name "pyk" of the authoring tool was constructed by removing "Vola" from "Volapyk" where "Vola" means "world" and "pyk" means "speak". Spoken mathematics like "limit as x tends to

infinity of 1 over x" certainly does look like Volapyk to most people. Furthermore, the extension ".pyk" was not used by anybody. The liberty to define the source language also allows an author to choose e.g. `Lim x -> infty : 1 / x` for denoting a limit. The source language of Logiweb has been used for proof checking at DIKU since 1985.
Having mentioned the differences with Mizar it should also be noted that Mizar .miz and Logiweb .pyk files are not all that different, and the possibility of a miz2pyk filter is being investigated.

1.1 Internet problems

If one author proves Zorns lemma in ZFC and another proves Hahn-Banachs theorem from Zorns lemma, then the correctness of the second proof depends on the correctness of the former. If the proofs were published as papers on Logiweb, then the paper proving Hahn-Banach would reference the one proving Zorn. When Logiweb checks the Hahn-Banach paper, it fetches the Zorn paper from the Internet and verifies that as well.
Now consider what happens if the author of the Zorn paper makes a slight change of the formulation of Zorns lemma. In that case, the latter paper may end up being incorrect.
Logiweb solves that using *immutability*. Immutability is normal in pure functional programming: if one computes $2::3$ then one gets a pair whose head is 2 and whose tail is 3. After that, one cannot change the head and tail of the pair. The pair remains immutable until it is garbage collected. If one wants a pair whose tail is 4 one has to compute $2::4$ which gives rise to a new pair without affecting the $2::3$ pair.
Logiweb pages are also immutable: once published, they cannot change. So now the author of the Hahn-Banach paper can be sure that even if a new version of the Zorn paper is produced, the Hahn-Banach paper will still reference the old Zorn paper.
But there still is a problem, since the author of the Zorn paper may garbage collect the old version. To counter for that, the author of the Hahn-Banach paper may mirror the Zorn paper. The Hahn-Banach author does so by writing `pyk HahnBanach -mirror` which makes the authoring tool resubmit all transitively referenced papers. Resubmitted papers are stored on the computer of the author, out of reach of other peoples garbage collectors.
The Logiweb software suite contains a demon named "logiweb" which must run on all sites that publish Logiweb papers. All logiweb demons in the world cooperate on indexing all Logiweb papers in the world, and no-one will notice them as long as they do what they are supposed to do. The Logiweb demon is the heart of the system. The pyk compiler is just one possible authoring tool (a wysiwyg authoring tool has been made but is not currently maintained).
The Logiweb demons ensure that, as long as at least one copy of the old Zorn paper remains accessible, no-one will notice that the author of the paper deleted the original.

1.2 Notational problems

If the Zorn author for some reason uses $x \oplus y$ for the disjoint union of x and y, and if the Hahn-Banach author wants to use $x \oplus y$ instead, then the Hahn-Banach author may define $[x \uplus y \stackrel{\text{macro}}{\to} \lambda x.\text{expand}\ (\ \lceil [x \uplus y \ddot{=} x \oplus y] \rceil\ ,\ x\ )]$ which makes Logiweb macro expand $x \oplus y$ to $x \oplus y$. This allows the Hahn-Banach author to write $x \oplus y$, but Logiweb will macro expand each $x \oplus y$ to $x \oplus y$ before proof checking.
If the Hahn-Banach author then wants to use $x \oplus y$ for some other operation, then the Hahn-Banach author may define his own $x \oplus y$ and define it to his liking. To Logiweb, each operator is identified by two natural numbers, a *reference* and an *index*. The reference is a big number which identifies the home paper of the operator (the paper which introduced the construct) and the index is a small number which distinguishes operators introduced by the same paper. Logiweb will not care about the two $\oplus$ operators having the same rendering. An author may fool himself and others by mistaking the two operators, but to Logiweb the two $\oplus$ operators are as different as any two other operators.
A malicious author could abuse notation to make readers think he has a checked proof of some lemma, but the fraud would be detected the moment somebody else try to use the lemma for further work.

1.3 Foundational problems

Logiweb was constructed for supporting Map Theory (MT) [5,1] which is alien to "normal" logic: MT builds on lambda calculus [2], has ZFC-power, unlimited lambda abstraction, and equality, and it can e.g. well-order any set by recursive use of Hilberts epsilon operator. But it does not have logical connectives; they have to be defined. A large MT proof [7] has been formalized in Isabelle [18], but existing systems are not very good at supporting MT. Instead of making a taylor-made system for MT, Logiweb was constructed such that it can support any theory equally well. Furthermore, Logiweb was constructed with human readability in mind. As an example, the text book [6] for teaching logic in computer science was written in a "Logiweb style" before Logiweb itself was designed. That allowed to collect the requirements needed to formalize a complete math book without sacrificing readability.
But the ability to support different kinds of logic raises new problems. Different authors are likely to use different axiomatic systems, and a single paper may use more than one axiomatic system. As an example, the present paper defines both propositional calculus Prop and first order logic FOL. To avoid conflicts, each lemma must state which axiomatic system it is relative to. It may be non-trivial to use lemmas proved in one axiomatic system in proofs that are relative to another system. Using Prop lemmas in FOL is easy since, as we shall see, FOL builds on top of Prop. It is more cumbersome to prove ZFC lemmas in MT even though every ZFC lemma is provable in MT. If one has to use ZFC lemmas in MT, Logiweb offers two possibilities: The easy one is to state as an axiom scheme that any proved ZFC lemma automatically

becomes an axiom of MT. The more cumbersome one is to define a proof tactic which, given a ZFC proof, translates the proof to an MT proof.

1.4 Rendering problems

Logiweb renders pages using TEX and PDF. From August 11 to November 23, 2004, Logiweb also had support for MathML [15]. Support was removed, however, for several reasons: The automatically generated MathML could easily bring the users browser to the knees, TEX had lots of facilities for bibliographies, indexes, and tables of contents, and the very purpose of MathML is not particularly compatible with Logiweb. MathML provides a nice and easy way to get mathematics on the web and it is easy to write and modify by hand. But Logiweb has no need for a human friendly intermediate format here. Furthermore, PDF and TEX source can be fed directly from Logiweb into EasyChair or even a publishing house. The latter was done in [9].

It should be noted that the rendering machinery of Logiweb is not tied to TEX and PDF. TEX and PDF are only supported in the sense that TEX, BIBTEX, makeindex, and dvipdfm are external programs which Logiweb provides access to during rendering. All other rendering must be done by Logiwebs own machinery. As an example, Logiweb also generates html, but that is done internally.

So any user who wants Logiweb to render pages in MathML may define their own "renderer" and publish it as a Logiweb paper. That would allow anybody else to reference that renderer and themselves write papers which were rendered in MathML.

The OMDoc format [13] is not supported either. But, again, one may add support without opening the code of Logiweb itself. Like MathML, OMDoc seems best suited for information in flux, i.e. information which the author may change now and then without a need for keeping the old version. OMDoc is suited as an output format for Logiweb, i.e. as a format in which papers are rendered. But OMDoc invites inclusion of URLs inside documents. And one should avoid relying on URLs in Logiweb papers since the immutability of Logiweb papers prevents authors from repairing broken links.

Like OMDoc, Logiweb supports linear, tree-like, and DAG-like proofs. At the inter-proof level, DAG-like structures are supported by the ability to use lemmas in a non-circular manner. At the intra-proof level, DAG-like proofs are supported by the ability to use arbitrary elements from the set of premises for further reasoning. Tree-like proofs are supported by the cut sequent operation, and linear proofs are supported by lists formed using the cut sequent operation to bind lines together.

Contrary to OMDoc, Logiweb proofs are either formal (meaning they are checked mechanically) or informal (meaning they are not checked). A partly formal proof of a lemma L, however, may be represented defining a construct P to denote the conjunction of the unproved statements used in the proof and then stating and proving $P \vdash L$ instead of L. The proof of $P \vdash L$ can then be

checked formally, and L will be easy to prove if the proof obligation P is proved later on on another Logiweb page.
One particular kind of rendering is rendering of executables. As an example, one could define the Unix "ls" command in a Logiweb paper. Rendering of the paper would then result in an executable named "ls" which does what Unix "ls" is supposed to do provided the definition of "ls" in the paper is correct.

1.5 Availability

Logiweb is available from http://logiweb.eu/. First beta release (version 0.1.1) was released on December 27, 2006. The current version is 0.1.5. Starting from version 0.1.1 it is the intension to make all updates backward compatible. All of Logiweb except its logo is available under the GNU public license.
At the time of writing, Logiweb runs at http://logiweb.eu, http://logiweb.imm.dtu.dk, and http://topps.diku.dk/logiweb. The two first sites support a Logiwiki each, allowing anyone to publish on Logiweb without installing local software. All three sites provide a tutorial.
At the time of writing, the present paper is available at http://logiweb.eu/grue/pages/Logiweb+layers/latest/. Note that a Logiweb paper has one and only one Logiweb reference but may be available under many URLs. This may happen if a paper is mirrored. Furthermore, the URLs of a paper may change over time, but the Logiweb reference remains fixed. The Logiweb reference of the present paper is BUpCgdix9lwGZKohkESIwWkv dwwIhPa08-fhoigBB and is based on a RIPEMD-160 hash key [3]. It is the task of the afore-mentioned Logiweb demons to keep track of the relationship between Logiweb references and URLs. Logiweb demons listen at particular URLs such as http://logiweb.eu/logiweb/server/relay/ called "relays". To look up a paper with a given Logiweb reference, append the URL of an arbitrary relay with the string "64/" if the Logiweb reference is expressed base 64, the Logiweb reference itself, and, optionally, a relative address such as "/2/body/tex/page.pdf" in case one wants a rendering rather than raw bytes.
At the time of writing, both http://logiweb.eu/grue/pages/Logiweb+layers/latest/ and http://logiweb.eu/logiweb/server/relay/64/BUpCgdix9lwGZKohkESIwWkvdwwIhPa08-fhoigBB/2/ point to the "front page" of the present version of the present paper. But the former may point to some other version in the future while the latter will continue to point to the present version as long as at least one copy of the present version remains on Logiweb. The 2/ at the end of the latter reference means "two levels above the raw bytes", i.e. at address ../../ relative to the raw bytes of the page.

1.6 Overview

The following sections present the programming, metalogic, tactic, and proof layers of Logiweb, respectively.

2 Programming layer

2.1 Combinations

As an example of the programming and rendering facilities of Logiweb, we define the binomial coefficient

$$\binom{n}{k} = \frac{n!}{k!n-k!}$$

A recursive definition reads:

$$[\binom{n}{k} \overset{\text{val}}{\to} \textbf{if } k = 0 \textbf{ then } 1 \textbf{ else } \binom{n-1}{k-1} \cdot n \text{ div } k]$$

As an example, we have $[\binom{4}{2} = 6]^{\cdot}$.

2.2 Syntax

The source text of Section 2.1 reads:

```
\subsection{Combinations}\label{sec:Combinations}

As an example of the programming and
rendering facilities of Logiweb, we
define the binomial coefficient

"[[[ (( n , k )) = (( n factorial /
k factorial,( n - k ) factorial )) ]]]"

\noindent A recursive definition reads:

"[[[ value define (( n , k )) as
if k = 0 then 1 else (( n - 1 , k - 1 ))
* n div k end define ]]]"

\noindent As an example, we have
"[[ ttst (( 4 , 2 )) = 6 end test ]]".
```

Section 2.1 contains an informal formula $((\mathrm{n}, \mathrm{k})) = \ldots$, a value definition (value define ...), and a test case (ttst ...). The formulas are mixed with TEX source with formulas delimited by `"[...]"` (double and triple brackets are like single brackets but also change to TEX math and display math mode, respectively). The value definition is what actually defines the binomial coefficient to Logiweb. Logiweb uses that definition when verifying the test case. Logiweb simply ignores the informal formula. The test case in Section 2.1 reveals to thorough readers that it is formally checked by an almost invisible dot superscript.

2.3 Aspects

The present paper has an electronic appendix [10] which contains the following:

```
"[[ tex define (( n , k )) as "
\left( \begin{array}{l} "[ n ]"
\\ "[ k ]"
\end{array}\right)" end define ]]"
```

That defines how to render the binomial coefficient. Logiweb uses named parameters `"[ n ]"` and `"[ k ]"` where TEX uses positional parameters #1 and #2.

The two definitions of the binomial coefficient stated so far define the value and tex aspects of the coefficient, respectively. Logiweb allows to assign an arbitrary number of aspects to any construct and allows users to invent new aspects.

2.4 Headers

The source text of the present paper starts thus:

```
PAGE Logiweb layers

BIBLIOGRAPHY
"check" "http:check/latest/vector/page.lgw",
"base" "http:base/latest/vector/page.lgw"

PREASSOCIATIVE
"check" check
"base" base
"" (( " , " ))
"" (( " / " ))

PREASSOCIATIVE
"base" +"
```

The first line gives the page a local name. The next three lines reference two previously published Logiweb pages (the references given are relative to a "current directory", one may also give full URLs or Logiweb references). The bibliography provides access to all constructs defined on the two mentioned pages plus all their transitively referenced pages.

The next five lines import the "check" construct from the check page, the "base" construct from the base page, and defines two new constructs named $((*, *))$ and $((*/*))$. The new constructs become constructs of the Logiweb layers page. Lines that begin with the empty string `""` introduce new constructs. Lines beginning with a page name import a construct from that page. Inside defined and imported constructs, the double quote `"` servers as a parameter placeholder.

The next two lines import a "gluing" prefix plus. It is gluing in the sense that no space is allowed between the plus sign and its argument. A gluing plus is used for expressing numerals like $^{+}117$. Importing the gluing plus implicitly imports all constructs with the same priority as the gluing plus, which happens to include a gluing minus. For that reason, one may write $^{-}117$ even though there is no explicit mention of a gluing minus in the header (for completeness: the base page also defines a unary minus -117 which changes the sign of its argument; the gluing minus only works with numerals and is handled at macro expansion time whereas the unary minus can be applied to anything and is handled at run time).
Constructs mentioned in pre- and postassociativity sections are left and right associative, respectively, when used in text written left to right. Constructs in early associativity sections have higher priority than those in later sections. Constructs in the same associativity section have the same priority. As an example,

```
PREASSOCIATIVE
"check" all " : "
POSTASSOCIATIVE
"check" " imply "
```

makes `y = 0 imply all x : x >= 0 imply all x : x >= y` have the following tacit parentheses:

$$y = 0 \Rightarrow \text{allfunc } \lambda x.x \geq 0 \Rightarrow \text{allfunc } \lambda x.x \geq y$$

The input syntax is up to the user. As an example, the present author likes to write `imply` rather than `=>` for implication, but other authors may have other preferences. Logiweb supports Unicode, so one may even use a ⇒-character for implication.

3 Metalogic layer

3.1 Sequent calculus

A proof checker is no more than a big, recursive function when expressed in Logiwebs programming language. The check page referenced by the present paper defines one, particular proof checker which we shall refer to as the Logiweb sequent checker. All Logiweb pages which reference the check page as their first reference and which do not define their own proof checker are checked by the sequent checker.
Any user can define their own proof checker, but the sequent checker is general enough that there should be no need for doing so. The sequent checker implements a variant of sequent calculus [4,11]. The calculus allows to formulate axiomatic theories, lemmas, and proofs.

3.2 Terms

The following axioms of FOL [14] illustrate what a term of the Logiweb sequent calculus may look like:

[A1 $\stackrel{\text{stmt}}{\rightarrow}$ $\Pi \mathsf{x}\colon \Pi \mathsf{y}\colon \mathsf{x} \Rightarrow \mathsf{y} \Rightarrow \mathsf{x}$][A1 $\stackrel{\text{unitac}}{\rightarrow}$ $\lambda u.$unitac-rule (u)]

[A2 $\stackrel{\text{stmt}}{\rightarrow}$ $\Pi \mathsf{x}\colon \Pi \mathsf{y}\colon \Pi \mathsf{z}\colon \mathsf{x} \Rightarrow \mathsf{y} \Rightarrow \mathsf{z} \Rightarrow \mathsf{x} \Rightarrow \mathsf{y} \Rightarrow \mathsf{x} \Rightarrow \mathsf{z}$][A2 $\stackrel{\text{unitac}}{\rightarrow}$ $\lambda u.$unitac-rule (u)]

[A3 $\stackrel{\text{stmt}}{\rightarrow}$ $\Pi \mathsf{x}\colon \Pi \mathsf{y}\colon \neg \mathsf{y} \Rightarrow \neg \mathsf{x} \Rightarrow \neg \mathsf{y} \Rightarrow \mathsf{x} \Rightarrow \mathsf{y}$][A3 $\stackrel{\text{unitac}}{\rightarrow}$ $\lambda u.$unitac-rule (u)]

[A4 $\stackrel{\text{stmt}}{\rightarrow}$ $\Pi \mathsf{t}\colon \Pi \mathsf{x}\colon \Pi \mathsf{a}\colon \Pi \mathsf{b}\colon \lambda c.$sub0 ($\lceil \mathsf{b} \rceil$, $\lceil \mathsf{a} \rceil$, $\lceil \mathsf{x} \rceil$, $\lceil \mathsf{t} \rceil$, c) $\Vdash$ allfunc $\lambda \mathsf{x}.\mathsf{a} \Rightarrow \mathsf{b}$][A4 $\stackrel{\text{unitac}}{\rightarrow}$ $\lambda u.$unitac-rule (u)]

[A5 $\stackrel{\text{stmt}}{\rightarrow}$ $\Pi \mathsf{x}\colon \Pi \mathsf{a}\colon \Pi \mathsf{b}\colon \lambda c.\lceil \mathsf{x} \rceil$ objectavoid (c) $\lceil \mathsf{a} \rceil \Vdash$ allfunc $\lambda \mathsf{x}.\mathsf{a} \Rightarrow \mathsf{b} \Rightarrow \mathsf{a} \Rightarrow$ allfunc $\lambda \mathsf{x}.\mathsf{b}$][A5 $\stackrel{\text{unitac}}{\rightarrow}$ $\lambda u.$unitac-rule (u)]

[MP $\stackrel{\text{stmt}}{\rightarrow}$ $\Pi \mathsf{x}\colon \Pi \mathsf{y}\colon \mathsf{x} \Rightarrow \mathsf{y} \vdash \mathsf{x} \vdash \mathsf{y}$][MP $\stackrel{\text{unitac}}{\rightarrow}$ $\lambda u.$unitac-rule (u)]

[Gen $\stackrel{\text{stmt}}{\rightarrow}$ $\Pi \mathsf{u}\colon \Pi \mathsf{x}\colon \mathsf{x} \vdash$ allfunc $\lambda \mathsf{u}.\mathsf{x}$][Gen $\stackrel{\text{unitac}}{\rightarrow}$ $\lambda u.$unitac-rule (u)]

[Prop $\stackrel{\text{stmt}}{\rightarrow}$ MP $\oplus$ A1 $\oplus$ A2 $\oplus$ A3]

[FOL $\stackrel{\text{stmt}}{\rightarrow}$ Prop $\oplus$ Gen $\oplus$ A4 $\oplus$ A5]

Terms of the Logiweb sequent calculus are built up from $\Pi \mathsf{x}\colon \mathsf{y}$ (y is provable for all terms x), $\mathsf{x} \vdash \mathsf{y}$ (y is provable if x is provable), $\mathsf{x} \Vdash \mathsf{y}$ (y is provable if x evaluates to T), and $\mathsf{x} \oplus \mathsf{y}$ (both x and y are provable).

Constructs like $x \Rightarrow y$ and $\neg x$ live at the object level and do not mean anything to the Logiweb sequent calculus. The construct $\lambda c.\lceil x \rceil$ objectavoid (c) $\lceil y \rceil$ is defined such that it evaluates to T (truth) if x is an object variable not free in y and $\lambda c.$sub0 ($\lceil b \rceil$, $\lceil a \rceil$, $\lceil x \rceil$, $\lceil t \rceil$, c) is T if b is alpha-equivalent to a where x is replaced by t (see the check page for details).

A statement like

[A1 $\stackrel{\text{stmt}}{\rightarrow}$ $\Pi \mathsf{x}\colon \Pi \mathsf{y}\colon \mathsf{x} \Rightarrow \mathsf{y} \Rightarrow \mathsf{x}$][A1 $\stackrel{\text{unitac}}{\rightarrow}$ $\lambda u.$unitac-rule (u)]

defines the "statement" aspect of A1 to be $\Pi \mathsf{x}\colon \Pi \mathsf{y}\colon \mathsf{x} \Rightarrow \mathsf{y} \Rightarrow \mathsf{x}$. The Logiweb sequent calculus does not distinguish between axioms, rules, and theories, so all of the declarations leading up to the declaration of FOL above just macro expand to statement definitions. Even lemmas translate into statement definitions. What distinguishes lemmas from other statements is that they also have a "proof" aspect which proves the lemma. Axioms, rules, and theories are statements that do not have proofs.

3.3 Sequents

A Logiweb sequent is a triple $p :: s :: c :: \langle \rangle$ where c is a sequent term and p and s are sets of sequent terms. We shall refer to c as the conclusion, to elements of p as premises, and to elements of s as side conditions. A Logiweb sequent represents the statement that if all elements of p are true and all elements of s evaluate to T then c is true.

3.4 Sequent operations

Logiweb sequent calculus has thirteen sequent operations:

a^{I}	$\rightarrow \{a\} :: \emptyset :: a :: \langle\,\rangle$	Init
$a \vdash p :: s :: c :: \langle\,\rangle$	$\rightarrow p \setminus \{a\}, s, a \vdash c :: \langle\,\rangle$	Infer
$a \Vdash p :: s :: c :: \langle\,\rangle$	$\rightarrow p, s \setminus \{a\}, a \Vdash c :: \langle\,\rangle$	Endorse
$\Pi x\colon p :: s :: c :: \langle\,\rangle$	$\rightarrow p :: s :: \Pi x\colon c :: \langle\,\rangle^1$	Generalize
$p, s, a \vdash c :: \langle\,\rangle^{\rhd}$	$\rightarrow p \cup \{a\} :: s :: c :: \langle\,\rangle$	Ponens
$p, s, a \Vdash c :: \langle\,\rangle^{\unrhd}$	$\rightarrow p :: s \cup \{a\} :: c :: \langle\,\rangle$	Probans
$p :: s :: \Pi x\colon c :: \langle\,\rangle$ @ a	$\rightarrow p :: s :: \langle \mathrm{c} \vert \mathrm{x} := \mathrm{a}\rangle :: \langle\,\rangle^2$	Instantiate
$p, s, a \Vdash c :: \langle\,\rangle^*$	$\rightarrow p :: s :: c :: \langle\,\rangle^3$	Verify
$p, s, a \vdash b \vdash c :: \langle\,\rangle^{\mathrm{U}}$	$\rightarrow p, s, a \oplus b \vdash c :: \langle\,\rangle$	Uncurry
$p, s, a \oplus b \vdash c :: \langle\,\rangle^{\mathrm{C}}$	$\rightarrow p, s, a \vdash b \vdash c :: \langle\,\rangle$	Curry
$p :: s :: c :: \langle\,\rangle$ ie n	$\rightarrow p :: s :: n :: \langle\,\rangle^4$	Reference
$p :: s :: n :: \langle\,\rangle^{\mathrm{D}}$	$\rightarrow p :: s :: c :: \langle\,\rangle^4$	Dereference
$p :: s :: c :: \langle\,\rangle; p' :: s' :: c' :: \langle\,\rangle$	$\rightarrow p' \setminus \{c\} \cup p :: s \cup s' :: c' :: \langle\,\rangle$	Cut

As an example, $\mathrm{A1} \vdash \mathrm{A1}^{\mathrm{ID}}$ @ $x = 0$ @ $y = 0$ sequent evaluates as follows:

$$
\begin{array}{ll}
\mathrm{A1} \vdash \mathrm{A1}^{\mathrm{ID}} \mathbin{@} x = 0 \mathbin{@} y = 0 & \rightarrow \\
\mathrm{A1} \vdash \{\mathrm{A1}\} :: \emptyset :: \mathrm{A1} :: \langle\,\rangle^{\mathrm{D}} \mathbin{@} x = 0 \mathbin{@} y = 0 & \rightarrow \\
\mathrm{A1} \vdash \{\mathrm{A1}\} :: \emptyset :: \Pi \mathsf{x}\colon \Pi \mathsf{y}\colon \mathsf{x} \Rightarrow \mathsf{y} \Rightarrow \mathsf{x} :: \langle\,\rangle \mathbin{@} x = 0 \mathbin{@} y = 0 & \rightarrow \\
\mathrm{A1} \vdash \{\mathrm{A1}\} :: \emptyset :: \Pi \mathsf{y}\colon x = 0 \Rightarrow \mathsf{y} \Rightarrow x = 0 :: \langle\,\rangle \mathbin{@} y = 0 & \rightarrow \\
\mathrm{A1} \vdash \{\mathrm{A1}\} :: \emptyset :: \Pi \mathsf{y}\colon x = 0 \Rightarrow y = 0 \Rightarrow x = 0 :: \langle\,\rangle & \rightarrow \\
\emptyset, \emptyset, \mathrm{A1} \vdash x = 0 \Rightarrow y = 0 \Rightarrow x = 0 :: \langle\,\rangle &
\end{array}
$$

The resulting sequent has no premises, no side conditions, and conclusion $\mathrm{A1} \vdash x = 0 \Rightarrow y = 0 \Rightarrow x = 0$. For that reason, $\mathrm{A1} \vdash \mathrm{A1}^{\mathrm{ID}}$ @ $x = 0$ @ $y = 0$ is said to prove $\mathrm{A1} \vdash x = 0 \Rightarrow y = 0 \Rightarrow x = 0$.

Statements that can be proved in the Logiweb sequent calculus are intuitionistically valid. As an example, if ZFC is Zermelo-Fraenkel set theory with the Axiom of Choice and Zorn is Zorns lemma, then ZFC $\vdash$ Zorn is the intuitionistically valid statement that Zorn follows from ZFC. With suitable definitions of ZFC and Zorn, that statement would be provable in the sequent calculus.

4 Tactic layer

4.1 Proof construction

It is difficult to express proofs directly in the Logiweb sequent calculus. For that reason, the proof checker defined on the check page defines a number of

[1] If x is not free in any element of p or s

[2] If a is free for x in c

[3] If a evaluates to T

[4] If the statement aspect of n is c

macros and "tactics" which perform some of the trivial work in constructing proofs.

A Logiweb page has a "body" which is the parse tree of the source text. Logiweb pages are rendered on basis of the body using tex definitions.

In many situations it is convenient to use shorthand notation. Logiweb supports that by having a macro expansion facility. Among other, the macro expansion facility is responsible for translating axiom declarations into statement definitions. Evaluation of test cases and proof checking is done after macro expansion.

As a further convenience, proofs are "tactic" expanded before they are checked. In principle, tactic expansion is much like macro expansion. But tactic expansion is only applied to proofs, not to test cases. Furthermore, the sequent proof that results from tactic expansion is discarded as soon as it is verified whereas the outcome of macro expansion is kept in the memory of the proof checker. Hence, when the Logiweb proof checker verified the present paper it had access to the macro expanded version of the check page but not to the tactic expanded version. Furthermore, the tactic expanded version of each proof is discarded before the proof checker goes on to the next proof.

4.2 Definitions of tactics

A tactic is defined by defining the "tactic" aspect of a construct, and the tactic is used by including that construct in a proof. When the proof checker verifies a proof, it invokes the tactics and then sequent evaluates the outcome.

When the proof checker invokes a tactic, it applies the tactic to a "cache". Each page has a "cache" which is a functional array of information. Among other, the cache contains all definitions on the page itself and on all transitively referenced pages. Hence, the tactics have access to all definitions. Since lemmas and proofs macro expand into definitions, one would be able to write a tactic which searches all transitively referenced pages for suitable lemmas. The tactics defined on the check page are more modest, however.

The cache also contains a lot of other information such as the compiled versions of all value definitions, the bodies and macro expanded versions of all transitively referenced pages, and "diagnoses" of all referenced pages. Logiweb allows users to publish incorrect pages. And it may even make sense to reference incorrect pages. But incorrect pages are easy to distinguish from other pages in that they have a non-empty diagnose. That diagnose, when rendered, is supposed to tell what is wrong with the given page.

4.3 Tactic levels

The system of tactics defined on the check page defines three levels of tactics: A top level tactic which expands entire proofs, medium level tactics which take care of proof constructors, and low level tactics which take care of individual lemmas and rules.

The top level tactic is an evaluator which invokes all the medium level tactics in a proof. The top level tactic is installed in the root of each proof during macro expansion.

Most of the medium level tactics collect information and pass it on. As an example, the sequent operator $x \vdash y$ has an associated tactic which ensures that tactics inside y are aware that they can assume that x holds.

The check page also defines a somewhat more complex "unification" tactic $x \gg y$ which is a medium level tactic. The unification tactic takes two arguments x and y where x is an incomplete proof and y is the desired conclusion. The unification tactic tries to adapt x such that x proves y. It mainly does so by adding instantiation operators u @ v where it guesses v using unification.

Low level tactics may occur inside the x argument of $x \gg y$. Among other, there is a low level tactic for proving axioms like A1. From a human point of view, axioms are assumed rather than proved. But in the sequent calculus, axioms do have to be proved, and the proof depends on context. As an example, if the proof is relative to Prop then the tactic expansion of A1 reads $\mathrm{Prop} \vdash \cdots$ and A1 is proved by dereferencing Prop into $\mathrm{MP} \oplus \mathrm{A1} \oplus \mathrm{A2} \oplus \mathrm{A3}$ and then picking the second element using suitable sequent operations. If the proof is relative to FOL then a proof of A1 is slightly different.

5 Proof layer

5.1 A lemma and a proof

As a simple example, consider the following lemma and proof:

$[\mathrm{Taut} \overset{\mathrm{stmt}}{\rightarrow} \mathrm{Prop} \vdash \Pi\mathsf{x}\colon \mathsf{x} \Rightarrow \mathsf{x}][\mathrm{Taut} \overset{\mathrm{unitac}}{\rightarrow} \lambda u.\mathrm{unitac\text{-}lemma}\ (\ u\ )]$

$[\mathrm{Taut} \overset{\mathrm{proof}}{\rightarrow} \lambda p.\lambda c.\mathrm{taceval}\ ($

$\lceil \mathrm{Prop} \vdash \Pi\mathsf{x}\colon \mathrm{A2} \gg \mathsf{x} \Rightarrow \mathsf{y} \Rightarrow \mathsf{x} \Rightarrow \mathsf{x} \Rightarrow \mathsf{x} \Rightarrow \mathsf{y} \Rightarrow \mathsf{x} \Rightarrow \mathsf{x} \Rightarrow \mathsf{x}; \mathrm{A1} \gg \mathsf{x} \Rightarrow \mathsf{y} \Rightarrow \mathsf{x} \Rightarrow \mathsf{x}; \mathrm{MP} \rhd \mathsf{x} \Rightarrow \mathsf{y} \Rightarrow \mathsf{x} \Rightarrow \mathsf{x} \Rightarrow \mathsf{x} \Rightarrow \mathsf{y} \Rightarrow \mathsf{x} \Rightarrow \mathsf{x} \Rightarrow \mathsf{x}^{\mathrm{I}} \rhd \mathsf{x} \Rightarrow \mathsf{y} \Rightarrow \mathsf{x} \Rightarrow \mathsf{x}^{\mathrm{I}} \gg \mathsf{x} \Rightarrow \mathsf{y} \Rightarrow \mathsf{x} \Rightarrow \mathsf{x} \Rightarrow \mathsf{x}; \mathrm{A1} \gg \mathsf{x} \Rightarrow \mathsf{y} \Rightarrow \mathsf{x}; \mathrm{MP} \rhd \mathsf{x} \Rightarrow \mathsf{y} \Rightarrow \mathsf{x} \Rightarrow \mathsf{x} \Rightarrow \mathsf{x}^{\mathrm{I}} \rhd \mathsf{x} \Rightarrow \mathsf{y} \Rightarrow \mathsf{x}^{\mathrm{I}} \gg \mathsf{x} \Rightarrow \mathsf{x} \rceil$

$,\ \mathrm{tactic\text{-}push}\ (\ \mathrm{premises}\ ,\ \lceil \mathrm{Prop} \rceil\ ,\ \mathrm{tacstate0}\ )\ ,\ c\ )]$

The first line above (the one before line L01) defines the proof aspect of Taut to be $\mathrm{Prop} \vdash \cdots$. Effectively that makes all axioms and inference rules of propositional calculus available.

Line L01 translate into $\Pi\mathsf{x}\colon \cdots$. That produces the Π in the lemma.

Line L02 macro expands into a local macro definition which defines L02 as shorthand for $\mathsf{x} \Rightarrow \mathsf{y} \Rightarrow \mathsf{x} \Rightarrow \mathsf{x} \Rightarrow \mathsf{x} \Rightarrow \mathsf{y} \Rightarrow \mathsf{x} \Rightarrow \mathsf{x} \Rightarrow \mathsf{x}$ in the lines following Line L02. That is utilized in Line L04. Furthermore, Line L02 macro expands into a cut operation whose left argument is
$\mathrm{A2} \gg \mathsf{x} \Rightarrow \mathsf{y} \Rightarrow \mathsf{x} \Rightarrow \mathsf{x} \Rightarrow \mathsf{x} \Rightarrow \mathsf{y} \Rightarrow \mathsf{x} \Rightarrow \mathsf{x} \Rightarrow \mathsf{x}$ and whose right argument is the rest of the proof.

Later on, A2 is tactic expanded to a proof P of A2 by a low level tactic, and the unification tactic massages $P \gg \mathsf{x} \Rightarrow \mathsf{y} \Rightarrow \mathsf{x} \Rightarrow \mathsf{x} \Rightarrow \mathsf{x} \Rightarrow \mathsf{y} \Rightarrow \mathsf{x} \Rightarrow \mathsf{x} \Rightarrow \mathsf{x}$ into P @ x @ $\mathsf{y} \Rightarrow \mathsf{x}$ @ x.

After macro and tactic expansion, the proof has the form of a rather unreadable sequent proof which is verified and then discarded.

5.2 Source

The source of Section 5.1 starts thus:

```
As a simple example, consider the following lemma and proof:

"[ Prop lemma Taut : All #x : #x imply #x end lemma ]"

"[ Prop proof of Taut :

line L01 : Arbitrary >> #x ;

line L02 : A2 >>
  ( #x imply ( #y imply #x ) imply #x ) imply newline
  ( #x imply #y imply #x ) imply #x imply #x ;

line L03 : A1 >> #x imply ( #y imply #x ) imply #x ;

line L04 : MP ponens L02 ponens L03 >>
  ( #x imply #y imply #x ) imply #x imply #x ;

line L05 : A1 >> #x imply #y imply #x ;

line L06 : MP ponens L04 ponens L05 >> #x imply #x qed ]"
```

In the source, the present author has chosen to use `#x` and `#y` for meta variables and to use `All` for the meta quantifier as opposed to `all` for an object quantifier. The `newline` in Line L02 tells TEX that the given place is a good place for breaking the line. The blank line between lemma and proof is important to TEX as it puts TEX in vertical mode before the proof. The blank lines inside the pyk source of the proof are merely aesthetic.

5.3 Further examples

As a simple example of a macro, one may define

$$[x \trianglerighteq y \overset{\text{macro}}{\rightarrow} \lambda x.\text{expand}\ (\ \lceil [x \trianglerighteq y \mathrel{\ddot{=}} \text{MP} \rhd x \rhd y] \rceil\ ,\ x\ )]$$

and then replace e.g. MP $\rhd$ L02 $\rhd$ L03 by MP $\rhd$ L02 $\rhd$ L03 in line L04 of the proof above. This illustrates the simplest kind of macro definition available in Logiweb. The general macro facility is Turing complete.
Further proofs may be found at http://logiweb.eu/.
One topic which has not been touched upon is the handling of the deduction theorem. Deduction can be handled by macro expansion or tactic expansion or

by including the deduction rule as an inference rule of FOL. At the time of writing, deduction is treated as an inference rule, but a solution using tactics is being implemented.

6 Conclusion and further work

The macro, tactic, and rendering facilities of Logiweb together with the Logiweb sequent calculus and the proof checker makes it easy to define axiomatic theories and state lemmas and proofs in a readable style close to that of e.g. [14]. The rendering and programming facilities also make it easy to define and render constructs like the binomial coefficient in a style close to that of ordinary mathematics. The present paper only presents very few examples, and it is left to the reader to extrapolate. Larger proofs may be found at logiweb.eu. The reader may construct proofs more complex the that given here in less than half an hour by following the tutorial at logiweb.eu.
A particularly important problem not covered here is the handling of deduction. Deduction is currently being ported from being an inference rule to being a tactic. Proofs that use a deduction inference rule may be found at logiweb.eu and in [9]. Deduction is different in FOL and MT, so it has to be implemented twice. Furthermore, it is planned to investigate the possibility of translating Mizar .miz files to Logiweb .pyk files, and to verify [7] and [6] using Logiweb. Finally, it is the intension to continue doing bug fixes and backward compatible enhancements of the Logiweb system.

References

1. C. Berline and K. Grue. A κ-denotational semantics for Map Theory in ZFC+SI. *TCS*, 179(1–2):137–202, jun 1997.
2. A. Church. *The Calculi of Lambda-Conversion*. Princeton University Press, 1941.
3. Hans Dobbertin, Antoon Bosselaers, and Bart Preneel. RIPEMD-160: A strengthened version of RIPEMD. In *Fast Software Encryption*, pages 71–82, 1996. `http://citeseer.nj.nec.com/dobbertin96ripemd.html`.
4. G. Genzen. *The Collected Papers of Gerhard Gentzen*. North-Holland, 1969. M. E. Szabo (Ed.).
5. K. Grue. Map theory. *Theoretical Computer Science*, 102(1):1–133, jul 1992.
6. K. Grue. *Mathematics and Computation*, volume 1–3. DIKU, Universitetsparken 1, DK-2100 Copenhagen, 7 edition, 2001.
7. K. Grue. Map theory with classical maps. Technical Report 02/21, DIKU, 2002. `http://www.diku.dk/publikationer/tekniske.rapporter/2002/`.
8. K. Grue. Logiweb. In Fairouz Kamareddine, editor, *Mathematical Knowledge Management Symposium 2003*, volume 93 of *Electronic Notes in Theoretical Computer Science*, pages 70–101. Elsevier, Feb 2004.
9. K. Grue. Logiweb - a system for web publication of mathematics. In *Mathematical Software - ICMS 2006*, volume 4151/2006 of *Lecture Notes in Computer Science*, pages 343–353. Springer Berlin / Heidelberg, 2006.

10. K. Grue. The layers of logiweb - appendix. Technical report, Logiweb, 2007. http://logiweb.eu/logiweb/server/relay/64/BUpCgdix9lwGZKohkESIwWkvdwwIhPa08-fhoigBB/2/body/tex/appendix.pdf.
11. S.C. Kleene. *Introduction to Metamathematics*, volume 1 of *Bibliotheca Mathematica*. N-H, 1964.
12. D. Knuth. *The TeXbook*. Addison Wesley, 1983.
13. Michael Kohlhase. OMDoc: An open markup format for mathematical documents (version 1.2), March 16. 2005. http://www.mathweb.org/omdoc/index.html.
14. E. Mendelson. *Introduction to Mathematical Logic*. Wadsworth and Brooks, 3. edition, 1987.
15. Robert Miner and Jeff Schaeffer. A gentle introduction to MathML, 2001. `http://www.dessci.com/en/support/tutorials/mathml/default.htm`.
16. Michał Muzalewski. *An Outline of PC Mizar*. Foundation of Logic, Mathematics and Informatics, Mizar User Group, Brussels, 1993.
17. Dave Raggett and Davy Batsalle. Adding math to Web pages with EzMath. *j-COMP-NET-ISDN*, 30(1–7):679–681, apr 1998.
18. Sebastian C. Skalberg. *An Interactive Proof System for Map Theory*. PhD thesis, University of Copenhagen, oct 2002. `http://www.mangust.dk/skalberg/phd/`.
19. Andrzej Trybulec and Howard Blair. Computer assisted reasoning with MIZAR. In Aravind Joshi, editor, *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pages 26–28, Los Angeles, CA, aug 1985. Morgan Kaufmann. `http://www.mizar.org/`.