Logiweb sequent calculus, Appendix

Klaus Grue

GRD-2006-03-06.UTC:13:37:57.803308

Contents

1 Introduction

The present appendix is part of the *Logiweb page* available at
http://www.diku.dk/cgi-bin/cginetd/grue/relay/go/
01827245F0F167F3F10B82632D74D1466DB6371642ECC689ADACF5A00806/2/
body/tex/page.pdf.

The hex number which contains a RIPEMD-160 [DBP96] hash key and a time stamp identifies the present paper uniquely on Logiweb. The part before the hex number is the address of a *Logiweb relay* which redirects the users browser to the paper. That part can be replaced by the address of any Logiweb

relay. The part after the hex number indicates the address of the body of the paper relative to the official, binary representation (2/ is shorthand for ../../).

Actually, the present page is rendered as three pdf files present at 2/body/tex/page.pdf, 2/body/tex/appendix.pdf, and 2/body/tex/chores.pdf where appendix.pdf is the present appendix. Many proofs and definitions are deferred to the present appendix to spare the reader of the main paper in the page.pdf file. The chores.pdf file defines how to render each construct in TEX. A good place to start browsing is 2/index.html.

Each Logiweb page has a Logiweb bibliography which lists previously published pages which the present page relies on. The present Logiweb page references the base page at
http://www.diku.dk/cgi-bin/cginetd/grue/relay/go/018040352BB22FB1B6433E6EBEAC52191C8D2C270588A4F6A5EEDBA00806/2/body/tex/page.pdf.
which defines the proof system used to verify the present page (c.f. 2/bibliography/tex/page.pdf).

As mentioned in the main paper, the user friendliness of the Logiweb sequent calculus is achieved by using the Turing complete macro expansion facility of Logiweb. To see the proofs present in the main paper after macro expansion, consult 2/expansion/tex/page.pdf. To see the proofs of the present appendix after macro expansion, consult 2/expansion/tex/appendix.pdf.

2 Notation

2.1 Hiding construct

$[\mathsf{x}^{\mathrm{hide}} \bowtie \text{“hide”}]$ proclaims $\mathsf{x}^{\mathrm{hide}}$ to denote hiding. We introduce it here to have a visible hiding construct instead of the invisible ‘unicode start of text’ operator normally used for hiding, c.f. the base page. The use of hiding in $[\mathsf{x} \doteqdot \mathsf{y}]^{\mathrm{hide}}$ later in this appendix is play for the gallery since the x is introduced on the base page and since macro definitions only have effect when they occur on the home page of the construct being macro defined.

2.2 Metaquantification

The following macro definition makes e.g. $\Pi\mathcal{A}, \mathcal{B}, \mathcal{C} : \mathcal{D}$ expand into $\forall\mathcal{A} : \forall\mathcal{B} : \forall\mathcal{C} : \mathcal{D}$. Note that $\forall\mathsf{x} : \mathsf{y}$ is meta quantification as defined on the base page which, unfortunately, is rendered exactly like object quantification in the present document. Such notational clashes are unavoidable in mathematics in general but becomes extremely visible in a system like Logiweb where referenced papers are only a click away. To avoid confusion between the two, identically looking quantifiers, we only use the meta quantifier from the base page in the macro definition below and otherwise use $\forall\mathsf{x} : \mathsf{y}$ to denote object quantification. The $(\mathsf{x})^{\mathbf{p}}$ operator used in the definition protects against macro expansion. The left hand side of the

macro definition is protected against macro expansion as is the 'recursive' call of the macro that occurs in the right hand side.

$[(\Pi \mathsf{x}\colon \mathsf{y})^{\mathbf{p}} \overset{\text{macro}}{\rightarrow} \lambda \mathsf{t}.\lambda \mathsf{s}.\lambda \mathsf{c}.\tilde{\mathcal{M}}($
if $\neg \mathsf{t}^1 \overset{\text{r}}{=} \lceil \mathsf{x}, \mathsf{y} \rceil$ **then**
$\tilde{\mathcal{Q}}(\mathsf{t}, \lceil \forall \mathsf{x}\colon \mathsf{y} \rceil, \langle \lceil \mathsf{x} \rceil :: \mathsf{t}^1, \lceil \mathsf{y} \rceil :: \mathsf{t}^2 \rangle)$ **else**
$\tilde{\mathcal{Q}}(\mathsf{t}, \lceil \forall \mathsf{x}\colon (\Pi \mathsf{y}\colon \mathsf{z})^{\mathbf{p}} \rceil, \langle \lceil \mathsf{x} \rceil :: \mathsf{t}^{11}, \lceil \mathsf{y} \rceil :: \mathsf{t}^{12}, \lceil \mathsf{z} \rceil :: \mathsf{t}^2 \rangle), \mathsf{s}, \mathsf{c})]$

The definition above is a general macro definition as opposed to more restricted macro definitions like [Arbitrary $\gg$ i; $\mathsf{p} \doteqdot \Pi \mathsf{i}\colon \mathsf{p}$] which just define a left hand side to expand into a right hand side. The $[\mathsf{x} \doteqdot \mathsf{y}]^{\text{hide}}$ construct itself is defined by a general macro definition and happens to expand into a general macro definition, so the $[\mathsf{x} \doteqdot \mathsf{y}]^{\text{hide}}$ construct is just syntactic sugar.

A general macro definition assigns a 'macro' aspect to the principal operator of the right hand side. The definition ignores the parameters of the principal operator. Instead, the right hand side of the general macro definition must take three arguments, t, s, and c. When a macro aspect is invoked, it is applied to the term t to be expanded, a 'macro state' s, and a 'Logiweb cache' c.

The macro state should itself be a function of three parameters t, s, and c and it is the function to be invoked when continuing the macro expansion. A macro like the macro protection macro $(\mathsf{x})^{\mathbf{p}}$ protects its argument x against macro expansion by returning it instead of calling s on it. The construct above calls the general macro expansion invokation function $\tilde{\mathcal{M}}(\mathsf{t}, \mathsf{s}, \mathsf{c})$ which simply applies s to t, s, and c. So the definition above does something to t and then continues macro expansion.

The Logiweb cache c is the cache of the home page of t, i.e. the page on which t occurs. The cache contains, among other, all definitions present on the home page and all pages transitively referenced by the home page. It also contains other things like compiled versions of all value defined constructs. This is badly needed for efficiency reasons because macro expansion is done not by the Logiweb computing engine itself but by a self-interpretter which is run by the Logiweb engine and simulates the Logiweb engine. This self-interpretter, which resides inside the macro state s, fetches compiled versions of constructs from the cache c for efficiency reasons.

The term t has the structure described in Section 3.1. t^2 is the second subtree of t and t^{12} is the second subtree of the first subtree of t. $\mathsf{t} \overset{\text{r}}{=} \mathsf{t}'$ is true when t and t' have the same principal operator.

The $\tilde{\mathcal{Q}}(\mathsf{t}, \mathsf{p}, \mathsf{a})$ construct expands the pattern p according to the association list a. Some day, when the author has some spare time, a new version of $\tilde{\mathcal{Q}}(\mathsf{t}, \mathsf{p}, \mathsf{a})$ will be implemented in which the association list is replaced by a facility like the 'comma' in Lisp backquotes.

The construct $\tilde{\mathcal{Q}}(\mathsf{t}, \mathsf{p}, \mathsf{a})$ not only expands the pattern p according to the association list a, it also extracts the 'debugging information' of the principal operator of t and installs that debugging information in all operators that come from the pattern p. The debugging information indicates the exact location of the construct before macro expansion.

The $\lceil \mathsf{t} \rceil$ construct evaluates to the Logiweb representation of the term t.

2.3 The 'Arbitrary' proof constructor

The 'Arbitrary' proof constructor defined on the base page does not take advantage of the meta quantification macro above. For that reason we introduce a new construct for introducing arbitrary meta variables:

[Arbitrary $\gg$ i; p $\doteq$ Πi: p]

Actually, the definition above was used as an example of a simple macro in Section 2.2. When the same aspect of the same construct is defined more than once (e.g. the macro aspect of Arbitrary $\gg$ i; p), only the first definition has effect. But the Logiweb compiler prints a warning if the same aspect of the same construct have definitions that contradict each other. The repeated definition of Arbitrary $\gg$ i; p above gives no warnings since the two definitions have identical right hand sides.

2.4 Macro indentation

The following construct increases the left margin. Should be used in inline math mode only. Should be replaced by automatic indentation inside blocks as soon as possible.

[MacroIndent(x) $\doteq$ x]

2.5 Protected macro expansion

The following construct is a general macro definition construct which both protects its left hand side against macro expansion and renders its left hand side using the tex name aspect. To see the definitions of the tex and tex name aspects of $[\mathsf{x} \overset{\circ}{=} \mathsf{y}]^{\mathrm{hide}}$ consult the chores part of the present Logiweb page.

$[[\mathsf{x} \overset{\circ}{=} \mathsf{y}] \doteq [(\mathsf{x})^{\mathbf{p}} \overset{\mathrm{macro}}{\rightarrow} \mathsf{y}]]$

2.6 Object quantification

The following is a repetion of Section 2.2 but concerns object quantification. It allows to write $\forall \mathsf{x}, \mathsf{y}, \mathsf{z}\colon \mathsf{u}$ for $\forall \mathsf{x}\colon \forall \mathsf{y}\colon \forall \mathsf{z}\colon \mathsf{u}$.

$[(\forall \mathsf{x}\colon \mathsf{y})^{\mathbf{p}} \overset{\mathrm{macro}}{\rightarrow} \lambda \mathsf{t}.\lambda \mathsf{s}.\lambda \mathsf{c}.\tilde{\mathcal{M}}($
if $\neg \mathsf{t}^1 \overset{\mathrm{r}}{=} \lceil \mathsf{x}, \mathsf{y} \rceil$ **then**
$\tilde{\mathcal{Q}}(\mathsf{t}, \lceil \forall_{\mathrm{obj}} \mathsf{x}\colon \mathsf{y} \rceil, \langle \lceil \mathsf{x} \rceil :: \mathsf{t}^1, \lceil \mathsf{y} \rceil :: \mathsf{t}^2 \rangle)$ **else**
$\tilde{\mathcal{Q}}(\mathsf{t}, \lceil \forall_{\mathrm{obj}} \mathsf{x}\colon (\forall \mathsf{y}\colon \mathsf{z})^{\mathbf{p}} \rceil, \langle \lceil \mathsf{x} \rceil :: \mathsf{t}^{11}, \lceil \mathsf{y} \rceil :: \mathsf{t}^{12}, \lceil \mathsf{z} \rceil :: \mathsf{t}^2 \rangle), \mathsf{s}, \mathsf{c})]$

2.7 Proof constructs

The Begin b; l : End; p construct puts a parenthesis around the block b and places a cut operator between b and p. The construct is complicated by the line number l which should be macro defined locally to denote the conclusion of the block b. This is non-trivial since the conclusion is the last line prefixed by all premises, side-conditions, and meta-quantifiers introduced in earlier lines.

$[\mathrm{Begin}\,\mathsf{b};\mathsf{l}:\mathrm{End};\mathsf{p} \stackrel{\circ}{=} \lambda\mathsf{t}.\lambda\mathsf{s}.\lambda\mathsf{c}.\mathrm{Block}_1(\mathsf{t},\mathsf{s},\mathsf{c})]$

$[\mathrm{Block}_1(\mathsf{t},\mathsf{s},\mathsf{c}) \doteq \mathsf{t}!\mathsf{s}!\mathsf{c}!$
$\mathbf{let}\ \mathsf{b} = \tilde{\mathcal{M}}(\mathsf{t}^1,\mathsf{s},\mathsf{c})\ \mathbf{in}$
$\mathbf{let}\ \mathsf{x} = \mathrm{Block}_2(\mathsf{b})\ \mathbf{in}$
$\mathbf{let}\ \mathsf{q} = \tilde{\mathcal{Q}}(\mathsf{t},\lceil(\mathbf{let}\ \mathsf{l} \ddot{=} \mathsf{x}\ \ \mathrm{in}\ \ \mathsf{p})^{\mathsf{p}}\rceil,\langle\lceil\mathsf{l}\rceil :: \mathsf{t}^2,\lceil\mathsf{p}\rceil :: \mathsf{t}^3,\lceil\mathsf{x}\rceil :: \mathsf{x}\rangle)\ \mathbf{in}$
$\mathbf{let}\ \mathsf{q} = \tilde{\mathcal{M}}(\mathsf{q},\mathsf{s},\mathsf{c})\ \mathbf{in}$
$\tilde{\mathcal{Q}}(\mathsf{t},\lceil\mathsf{b};\mathsf{q}\rceil,\langle\lceil\mathsf{b}\rceil :: \mathsf{b},\lceil\mathsf{q}\rceil :: \mathsf{q}\rangle)]$

$[\mathrm{Block}_2(\mathsf{b}) \doteq$
$\mathbf{if}\ \mathsf{b} \stackrel{\mathrm{r}}{=} \lceil\mathsf{x}\vdash\mathsf{y}\rceil\ \mathbf{then}\ \tilde{\mathcal{Q}}(\mathsf{b},\lceil\mathsf{x}\vdash\mathsf{y}\rceil,\langle\lceil\mathsf{x}\rceil :: \mathsf{b}^1,\lceil\mathsf{y}\rceil :: \mathrm{Block}_2(\mathsf{b}^2)\rangle)\ \mathbf{else}$
$\mathbf{if}\ \mathsf{b} \stackrel{\mathrm{r}}{=} \lceil\mathsf{x}\Vdash\mathsf{y}\rceil\ \mathbf{then}\ \tilde{\mathcal{Q}}(\mathsf{b},\lceil\mathsf{x}\Vdash\mathsf{y}\rceil,\langle\lceil\mathsf{x}\rceil :: \mathsf{b}^1,\lceil\mathsf{y}\rceil :: \mathrm{Block}_2(\mathsf{b}^2)\rangle)\ \mathbf{else}$
$\mathbf{if}\ \mathsf{b} \stackrel{\mathrm{r}}{=} \lceil\Pi\mathsf{x}\colon\mathsf{y}\rceil\ \mathbf{then}\ \tilde{\mathcal{Q}}(\mathsf{b},\lceil\Pi\mathsf{x}\colon\mathsf{y}\rceil,\langle\lceil\mathsf{x}\rceil :: \mathsf{b}^1,\lceil\mathsf{y}\rceil :: \mathrm{Block}_2(\mathsf{b}^2)\rangle)\ \mathbf{else}$
$\mathbf{if}\ \mathsf{b} \stackrel{\mathrm{r}}{=} \lceil\mathsf{x};\mathsf{y}\rceil\ \mathbf{then}\ \mathrm{Block}_2(\mathsf{b}^2)\ \mathbf{else}$
$\mathbf{if}\ \mathsf{b} \stackrel{\mathrm{r}}{=} \lceil\mathsf{x}\gg\mathsf{y}\rceil\ \mathbf{then}\ \mathsf{b}^2\ \mathbf{else}\ \perp\!\!\!\perp]$

$[\mathrm{Last\ block\ line}\,\mathsf{a}\gg\mathsf{i}\,;\ddot{=}(\mathsf{a}\gg\mathsf{i})]$

2.8 Modus ponens at the object level

We shall apply modus ponens to two arguments so often that this idiom deserves its own macro:

$[\mathsf{x}\trianglerighteq\mathsf{y} \ddot{=} \mathrm{MP}\triangleright\mathsf{x}\triangleright\mathsf{y}]$

3 Side conditions

3.1 Logiweb terms

Metavariables, object terms, statements, and sequent terms are all implemented as Logiweb terms which essentially have the following syntax:

pdigit	::=	“1” | “2” | “3” | “4” | “5” | “6” | “7” | “8” | “9”
digit	::=	“0” | pdigit
positive	::=	pdigit | positive digit
cardinal	::=	“0” | positive
reference	::=	cardinal
identifier	::=	cardinal
symbol	::=	“(” reference“␣” identifier “)”
arglist	::=	“)” | “␣” term arglist
term	::=	“(” symbol arglist

The only thing omitted from the syntax is a third element of symbols which contains debugging information. For details on debugging information, consult the base page.

Each Logiweb page introduces one or more new constructs. As an example, the present page introduces the binary operation $\mathsf{x}\Rightarrow\mathsf{y}$.

Each Logiweb page has a unique reference, and each construct introduced by a page has an identifier which is unique within the page. Hence, reference and identifier together determine a construct uniquely.

3.2 Deduction

$[\mathrm{Ded}(\mathsf{p},\mathsf{c}) \doteq \lambda\mathsf{x}.\mathrm{Ded}_0(\lceil\mathsf{p}\rceil,\lceil\mathsf{c}\rceil)]$

True if the premise p implies the conclusion c according to the deduction rule. The construct quotes its arguments and pass them on to $\mathrm{Ded}_0(\mathsf{p},\mathsf{c})$. The quoting and the $\lambda\mathsf{x}.\cdots$ makes $\mathrm{Ded}(\mathsf{p},\mathsf{c})$ suited as a side condition. The present side condition ignores x. But when a side condition is invoked, it is applied to the cache of the home page of the side condition (the page on which the side condition occurs). That way the side condition gets access to all definitions present on the home page and all pages transitively referenced by the home page. As an example, that has allowed to define an 'axiom of definition' in another theory on another Logiweb page in which all value definitions automatically become axiom schemes. This possibility has been excluded from the present paper because of the Ijcar page limitation and because the facility is not particularly useful in Peano arithmetic. It is much more useful in theories that build on lambda calculus.

$[\mathrm{Ded}_0(\mathsf{p},\mathsf{c}) \doteq \mathsf{c}!\mathrm{Ded}_8(\mathsf{p},\mathsf{T}) \mathbin{\ddot{\wedge}} \mathrm{Ded}_1(\mathrm{Ded}_7(\mathsf{p}),\mathsf{c},\mathsf{T})]$

True if the premise p implies the conclusion c according to the deduction rule. The function checks the premise for meta-closedness, strips meta-quantifiers from p and then calls $\mathrm{Ded}_1(\mathsf{p},\mathsf{c},\mathsf{s})$ with an empty list s of side conditions. The $\mathsf{x}!\mathsf{y}$ construct evaluates and discards x and then returns y. It is included to ensure strictness of $\mathrm{Ded}_0(\mathsf{p},\mathsf{c})$ which in turn increases the efficiency. Some day the present author will define a 'strict value definition' operator so that users don't have to add $\mathsf{x}!\mathsf{y}$ operators manually. The $\mathsf{x}!\mathsf{y}$ construct takes zero CPU-time when used in functions that are 'fit' for optimization. See the base page for details on 'fitness' analysis.

$[\mathrm{Ded}_1(\mathsf{p},\mathsf{c},\mathsf{s}) \doteq \textbf{if } \mathsf{c} \stackrel{\mathrm{r}}{=} \lceil\mathsf{x} \Vdash \mathsf{y}\rceil \textbf{ then } \mathrm{Ded}_1(\mathsf{p},\mathsf{c}^2,\mathsf{c}^1 :: \mathsf{s}) \textbf{ else } \mathrm{Ded}_2(\mathsf{p},\mathsf{c},\mathsf{s})]$

Move side conditions from the conclusion c to the list s. $\mathsf{x} \stackrel{\mathrm{r}}{=} \mathsf{y}$ is true if the terms x and y have the same root (i.e. the same principal operator) so $\mathsf{c} \stackrel{\mathrm{r}}{=} \lceil\mathsf{x} \Vdash \mathsf{y}\rceil$ is true if the principal operator of c is an endorsement. c^1 and c^2 are the first and second argument, respectively, of the endorsement.

$$[\mathrm{Ded}_2(\mathsf{p},\mathsf{c},\mathsf{s}) \doteq \mathsf{s}!\mathsf{p} \stackrel{\mathrm{r}}{=} \lceil\mathsf{x} \vdash \mathsf{y}\rceil \wedge \mathsf{c} \stackrel{\mathrm{r}}{=} \lceil\mathsf{x} \Rightarrow \mathsf{y}\rceil \begin{cases} \mathrm{Ded}_3(\mathsf{p}^1,\mathsf{c}^1,\mathsf{s},\mathsf{T}) \wedge \mathrm{Ded}_2(\mathsf{p}^2,\mathsf{c}^2,\mathsf{s}) \\ \mathrm{Ded}_4(\mathsf{p},\mathsf{c},\mathsf{s},\mathrm{Ded}_6(\mathsf{p},\mathsf{c},\mathsf{T},\mathsf{T})) \end{cases}]$$

This function is true if the premise p is equal to the conclusion c except that inferences are replaced by implications. **if** x **then** y **else** z

and $\mathsf{x}\left\{\begin{array}{l}\mathsf{y}\\ \mathsf{z}\end{array}\right.$ express the same conditional construct. Choosing between the two is a purely stylistic choice.

$[\mathrm{Ded}_3(\mathsf{p},\mathsf{c},\mathsf{s},\mathsf{b}) \doteq$ **if** $\neg\mathsf{c} \stackrel{\mathrm{r}}{=} \lceil\forall\mathsf{x}\colon\mathsf{y}\rceil$ **then** $\mathrm{Ded}_4(\mathsf{p},\mathsf{c},\mathsf{s},\mathsf{b})$ **else**
if $\mathsf{p} \stackrel{\mathrm{r}}{=} \lceil\forall\mathsf{x}\colon\mathsf{y}\rceil \wedge \mathsf{p}^1 \stackrel{\mathrm{t}}{=} \mathsf{c}^1$ **then** $\mathrm{Ded}_4(\mathsf{p},\mathsf{c},\mathsf{s},\mathsf{b})$ **else**
$\mathrm{Ded}_3(\mathsf{p},\mathsf{c}^2,\mathsf{s},(\mathsf{c}^1 :: \mathsf{c}^1) :: \mathsf{b})]$
$\mathrm{Ded}_3(\mathsf{p},\mathsf{c},\mathsf{s},\mathsf{b})$ moves quantified variables from the premise p to the association list b of 'bindings'. Each quantified variable x is added to b as the pair $\mathsf{x} :: \mathsf{x}$ indicating the x should be replaced by x (which is a way of saying that the variable cannot be instantiated). Moving of bound variables stops when there are no more bound variables to move or when the premise and conclusion happen to quantify the same variable. In both cases we have reached a point where premise and conclusion should be identical except for instantaition of variables. $\mathsf{x} \stackrel{\mathrm{t}}{=} \mathsf{y}$ is true if the terms x and y are identical. Logiweb terms contain debugging information which indicates the precise location each operator before macro expansion. $\mathsf{x} \stackrel{\mathrm{t}}{=} \mathsf{y}$ ignores this debugging information when it tests two terms for identity.

$[\mathrm{Ded}_4(\mathsf{p},\mathsf{c},\mathsf{s},\mathsf{b}) \doteq \mathsf{s}!\mathsf{b}!$
if $\mathsf{p} \stackrel{\mathrm{r}}{=} \lceil x\rceil$ **then lookup**$(\mathsf{p},\mathsf{b},\mathsf{T}) \stackrel{\mathrm{t}}{=} \mathsf{c}$ **else**
if $\neg\mathsf{p} \stackrel{\mathrm{r}}{=} \mathsf{c}$ **then** F **else**
if $\mathsf{p} \stackrel{\mathrm{r}}{=} \lceil\forall\mathsf{x}\colon\mathsf{y}\rceil$ **then** $\mathsf{p}^1 \stackrel{\mathrm{t}}{=} \mathsf{c}^1 \wedge \mathrm{Ded}_4(\mathsf{p}^2,\mathsf{c}^2,\mathsf{s},(\mathsf{p}^1 :: \mathsf{p}^1) :: \mathsf{b})$ **else**
if $\neg\mathsf{p} \stackrel{\mathrm{r}}{=} \lceil\mathcal{X}\rceil$ **then** $\mathrm{Ded}_4^*(\mathsf{p}^{\mathrm{t}},\mathsf{c}^{\mathrm{t}},\mathsf{s},\mathsf{b})$ **else**
$\mathsf{p}^1 \stackrel{\mathrm{t}}{=} \mathsf{c}^1 \wedge \mathrm{Ded}_5(\mathsf{p},\mathsf{s},\mathsf{b})]$
Test that the premise p is identical to the conclusion c except for instantiation of object variables as specified by the association list b. A term is an object variable if the root of the term is the 'object variable operator'. For that reason, one can test whether or not a variable is an object variable by comparing its root to the root of an arbitrary object variable. For each meta variable, check that there are side conditions which ensure that the meta variables denote terms whose free object variables do not get caught in the context. Meta variables are recognized the same way as object variables. The object variable x and the meta variable $\mathcal{X}$ macro expand into $\overline{\mathsf{x}}$ and $\underline{\mathsf{x}}$, respectively, adding the object and meta variable operator, respectively, to the term x. The term x is neither an object nor a meta variable. x is a Logiweb programming language variable because no Logiweb definition assigns a value to it.

$[\mathrm{Ded}_4^*(\mathsf{p},\mathsf{c},\mathsf{s},\mathsf{b}) \doteq \mathsf{c}!\mathsf{s}!\mathsf{b}!$**if** p **then** T **else** $\mathrm{Ded}_4(\mathsf{p}^{\mathrm{h}},\mathsf{c}^{\mathrm{h}},\mathsf{s},\mathsf{b}) \wedge \mathrm{Ded}_4^*(\mathsf{p}^{\mathrm{t}},\mathsf{c}^{\mathrm{t}},\mathsf{s},\mathsf{b})]$
Elementwise application of $\mathrm{Ded}_4(\mathsf{p},\mathsf{c},\mathsf{s},\mathsf{b})$. x^{h} and x^{t} denote the head and tail, respectively, of the pair x. Terms are represented as lists comprising the principal operator followed by arguments just like Lisp S-expressions. The principal operator in turn is a list whose

first two elements are cardinals that identify the operator followed by debugging information.

$[\mathrm{Ded}_5(\mathsf{p},\mathsf{s},\mathsf{b}) \doteq \mathsf{p}!\mathsf{s}!\textbf{if } \mathsf{b} \textbf{ then } \top \textbf{ else}$
$\langle \lceil \mathsf{x}\#\mathsf{y} \rceil^{\mathrm{h}}, \langle \lceil \lceil * \rceil \rceil^{\mathrm{h}}, \mathsf{b}^{\mathrm{hh}} \rangle, \langle \lceil \lceil \mathsf{x} \rceil \rceil^{\mathrm{h}}, \mathsf{p} \rangle \rangle \in_{\mathrm{t}} \mathsf{s} \wedge \mathrm{Ded}_5(\mathsf{p},\mathsf{s},\mathsf{b}^{\mathrm{t}})]$
For each variable that is bound in the context according to the association list b, check that the list s of side conditions ensures that the meta variables p denotes a term whose free variables do not get caught in the context. $\mathsf{x} \in_{\mathrm{t}} \mathsf{y}$ is true if the term x belongs to the list y of terms.

$[\mathrm{Ded}_6(\mathsf{p},\mathsf{c},\mathsf{e},\mathsf{b}) \doteq \mathsf{p}!\mathsf{c}!\mathsf{b}!\mathsf{e}!$
$\textbf{if } \mathsf{p} \stackrel{\mathrm{r}}{=} \lceil x \rceil \textbf{ then } \mathsf{p} \in_{\mathrm{t}} \mathsf{e} \begin{cases} \mathsf{b} \\ (\mathsf{p} :: \mathsf{c}) :: \mathsf{b} \end{cases} \textbf{ else}$
$\textbf{if } \neg\mathsf{p} \stackrel{\mathrm{r}}{=} \mathsf{c} \textbf{ then } \top \textbf{ else}$
$\textbf{if } \mathsf{p} \stackrel{\mathrm{r}}{=} \lceil \mathcal{A} \rceil \textbf{ then } \mathsf{b} \textbf{ else}$
$\textbf{if } \mathsf{p} \stackrel{\mathrm{r}}{=} \lceil \forall \mathsf{x}\colon \mathsf{y} \rceil \textbf{ then } \mathrm{Ded}_6(\mathsf{p}^2,\mathsf{c}^2,\mathsf{c}^1 :: \mathsf{e},\mathsf{b}) \textbf{ else}$
$\mathrm{Ded}_6^*(\mathsf{p}^{\mathrm{t}},\mathsf{c}^{\mathrm{t}},\mathsf{e},\mathsf{b})]$

$[\mathrm{Ded}_6^*(\mathsf{p},\mathsf{c},\mathsf{e},\mathsf{b}) \doteq \mathsf{p}!\mathsf{c}!\mathsf{b}!\mathsf{e}!\textbf{if } \mathsf{p} \textbf{ then } \mathsf{b} \textbf{ else } \mathrm{Ded}_6^*(\mathsf{p}^{\mathrm{t}},\mathsf{c}^{\mathrm{t}},\mathsf{e},\mathrm{Ded}_6(\mathsf{p}^{\mathrm{h}},\mathsf{c}^{\mathrm{h}},\mathsf{e},\mathsf{b}))]$
Elementwise application of $\mathrm{Ded}_6(\mathsf{p},\mathsf{c},\mathsf{e},\mathsf{b})$.

$[\mathrm{Ded}_7(\mathsf{p}) \doteq \mathsf{p} \stackrel{\mathrm{r}}{=} \lceil \Pi\mathsf{x}\colon \mathsf{y} \rceil \begin{cases} \mathrm{Ded}_7(\mathsf{p}^2) \\ \mathsf{p} \end{cases}]$
Removal of all metaquantifiers.

$[\mathrm{Ded}_8(\mathsf{p},\mathsf{b}) \doteq$
$\textbf{if } \mathsf{p} \stackrel{\mathrm{r}}{=} \lceil \Pi\mathsf{x}\colon \mathsf{y} \rceil \textbf{ then } \mathrm{Ded}_8(\mathsf{p}^2,\mathsf{p}^1 :: \mathsf{b}) \textbf{ else}$
$\textbf{if } \mathsf{p} \stackrel{\mathrm{r}}{=} \lceil \mathcal{A} \rceil \textbf{ then } \mathsf{p} \in_{\mathrm{t}} \mathsf{b} \textbf{ else } \mathrm{Ded}_8^*(\mathsf{p}^{\mathrm{t}},\mathsf{b})]$
True if all free meta variables of p occur in the list b of meta variables.

$[\mathrm{Ded}_8^*(\mathsf{p},\mathsf{b}) \doteq \mathsf{b}!\textbf{if } \mathsf{p} \textbf{ then } \top \textbf{ else } \mathrm{Ded}_8(\mathsf{p}^{\mathrm{h}},\mathsf{b}) \ddot{\wedge} \mathrm{Ded}_8^*(\mathsf{p}^{\mathrm{t}},\mathsf{b})]$
Elementwise application of $\mathrm{Ded}_8(\mathsf{p},\mathsf{b})$.

3.3 Avoidance

$[\mathsf{x}^{\mathrm{var}} \doteq \mathsf{x} \stackrel{\mathrm{r}}{=} \lceil \bar{\mathsf{x}} \rceil]$

$[\mathsf{x}\#\mathsf{y} \ddot{=} \lceil \mathsf{x} \rceil \#^0 \lceil \mathsf{y} \rceil]$

$[\mathsf{x}\#^0\mathsf{y} \doteq \lambda\mathsf{c}.\mathsf{x}^{\mathrm{var}} \wedge \mathsf{y}^{\mathcal{C}} \wedge \mathsf{x}\#^1\mathsf{y}]$

$[\mathsf{x}\#^1\mathsf{y} \doteq \textbf{if } \mathsf{y}^{\mathrm{var}} \textbf{ then } \neg\mathsf{x} \stackrel{\mathrm{t}}{=} \mathsf{y} \textbf{ else}$
$\textbf{if } \neg\mathsf{y} \stackrel{\mathrm{r}}{=} \lceil \forall \mathsf{x}\colon \mathsf{y} \rceil \textbf{ then } (\mathsf{x}\#^*\mathsf{y}^{\mathrm{t}}) \textbf{ else}$
$\textbf{if } \mathsf{x} \stackrel{\mathrm{t}}{=} \mathsf{y}^1 \textbf{ then } \top \textbf{ else } (\mathsf{x}\#^1\mathsf{y}^2)]$

$[\mathsf{x}\#^*\mathsf{y} \doteq \mathsf{x}!\mathrm{If}(\mathsf{y},\top,(\mathsf{x}\#^1\mathsf{y}^{\mathrm{h}}) \ddot{\wedge} (\mathsf{x}\#^*\mathsf{y}^{\mathrm{t}}))]$

3.4 Substition

$[\langle \mathsf{a} \equiv \mathsf{b} | \mathsf{x} := \mathsf{t} \rangle \doteq \langle \lceil \mathsf{a} \rceil \equiv^0 \lceil \mathsf{b} \rceil | \lceil \mathsf{x} \rceil := \lceil \mathsf{t} \rceil \rangle]$

$[\langle \mathsf{a} \equiv^0 \mathsf{b} | \mathsf{x} := \mathsf{t} \rangle \doteq \lambda \mathsf{c}.\mathsf{x}^{\mathrm{var}} \wedge \langle \mathsf{a} \equiv^1 \mathsf{b} | \mathsf{x} := \mathsf{t} \rangle]$

$$[\langle \mathsf{a} \equiv^1 \mathsf{b} | \mathsf{x} := \mathsf{t} \rangle \doteq \mathsf{a}!\mathsf{x}!\mathsf{t}!$$
$$\textbf{if } \mathsf{b} \stackrel{\mathrm{r}}{=} \lceil \forall \mathsf{u} \colon \mathsf{v} \rceil \barwedge \mathsf{b}^1 \stackrel{\mathrm{t}}{=} \mathsf{x} \textbf{ then } \mathsf{a} \stackrel{\mathrm{t}}{=} \mathsf{b} \textbf{ else}$$
$$\textbf{if } \mathsf{b}^{\mathrm{var}} \wedge \mathsf{b} \stackrel{\mathrm{t}}{=} \mathsf{x} \textbf{ then } \mathsf{a} \stackrel{\mathrm{t}}{=} \mathsf{t} \textbf{ else}$$
$$\mathsf{a} \stackrel{\mathrm{r}}{=} \mathsf{b} \barwedge \langle \mathsf{a}^{\mathrm{t}} \equiv^* \mathsf{b}^{\mathrm{t}} | \mathsf{x} := \mathsf{t} \rangle]$$

$[\langle \mathsf{a} \equiv^* \mathsf{b} | \mathsf{x} := \mathsf{t} \rangle \doteq \mathsf{b}!\mathsf{x}!\mathsf{t}!\mathrm{If}(\mathsf{a}, \mathsf{T}, \langle \mathsf{a}^{\mathrm{h}} \equiv^1 \mathsf{b}^{\mathrm{h}} | \mathsf{x} := \mathsf{t} \rangle \barwedge \langle \mathsf{a}^{\mathrm{t}} \equiv^* \mathsf{b}^{\mathrm{t}} | \mathsf{x} := \mathsf{t} \rangle)]$

4 Proofs

4.1 Proofs of FOL axioms using the inference of deduction

We now prove axiom schemes A1 and A2 from [Men87]. Furthermore, we prove two, particular instances of A4 and A5 in a way that generalizes to arbitrary instances of those axiom schemes.

Furthermore, we prove a Repetition lemma which says $\Pi\mathcal{A}\colon \mathcal{A} \vdash \mathcal{A}$ by a 'hand made proof', i.e. by giving a proof metatactic explicitly which generates the sequent proof.

A page is verified by a top level verifier. The top level verifier of the present page is defined on the base page. That verifier invokes, among other, a proof verifier. The proof verifier invokes each proof (where each proof is supposed to be a proof metatactic). When it does so, it applies the proof metatactic to a Logiweb cache c and a pair $\mathsf{n} :: \mathsf{l}$. The cache c is the Logiweb cache of the home page of the proof, i.e. a structure which contains all definitions present on the home page plus all transitively referenced Logiweb pages. n is the identifier of the lemma (a cardinal which identifies the lemma uniquely within the home page) plus the contents of the lemma. The proof metatactic given below for proving Repetition ignores both arguments and just constructs a sequent term that proves $\Pi\mathcal{A}\colon \mathcal{A} \vdash \mathcal{A}$. All other proofs in the present paper macro expand into proofs that are proved by a metatactic which scans the proof for object tactics and invokes those object tactics. The only object tactic used is named $\gg$ and does unification.

[S **lemma** Repetition: $\Pi\mathcal{A}\colon \mathcal{A} \vdash \mathcal{A}$]

[S **lemma** A1′: $\Pi\mathcal{A}, \mathcal{B}\colon \mathcal{A} \Rightarrow \mathcal{B} \Rightarrow \mathcal{A}$]

[S **lemma** A2′: $\Pi\mathcal{A}, \mathcal{B}, \mathcal{C}\colon (\mathcal{A} \Rightarrow \mathcal{B} \Rightarrow \mathcal{C}) \Rightarrow (\mathcal{A} \Rightarrow \mathcal{B}) \Rightarrow \mathcal{A} \Rightarrow \mathcal{C}$]

[S **lemma** A4′: $\forall x, y\colon x + y = y + x \Rightarrow 2 + 3 = 3 + 2$]

[S **lemma** A5′: $\forall x\colon (2+3 = 5 \Rightarrow 2+3+x = 5+x) \Rightarrow 2+3 = 5 \Rightarrow \forall x\colon 2+3+x = 5+x$]

[**Proof of** Repetition: $\lambda c.\lambda x.[S \vdash \Pi\mathcal{A}{:}\,\mathcal{A}^I]$]

S **proof of** A1′:

L01:	Arbitrary $\gg$	$\mathcal{A}, \mathcal{B}$	;
L02:	Block $\gg$	Begin	;
L03:	Arbitrary $\gg$	$\mathcal{A}, \mathcal{B}$	;
L04:	Premise $\gg$	$\mathcal{A}$	;
L05:	Premise $\gg$	$\mathcal{B}$	;
L06:	Repetition $\triangleright$ L04 $\gg$	$\mathcal{A}$	;
L07:	Block $\gg$	End	;
L08:	Ded $\triangleright$ L07 $\gg$	$\mathcal{A} \Rightarrow \mathcal{B} \Rightarrow \mathcal{A}$	□

S **proof of** A2′:

L01:	Arbitrary $\gg$	$\mathcal{A}, \mathcal{B}, \mathcal{C}$	;
L02:	Block $\gg$	Begin	;
L03:	Arbitrary $\gg$	$\mathcal{A}, \mathcal{B}, \mathcal{C}$	;
L04:	Premise $\gg$	$\mathcal{A} \Rightarrow \mathcal{B} \Rightarrow \mathcal{C}$	;
L05:	Premise $\gg$	$\mathcal{A} \Rightarrow \mathcal{B}$	;
L06:	Premise $\gg$	$\mathcal{A}$	;
L07:	L05 $\trianglerighteq$ L06 $\gg$	$\mathcal{B}$	;
L08:	L04 $\trianglerighteq$ L06 $\gg$	$\mathcal{B} \Rightarrow \mathcal{C}$	;
L09:	L08 $\trianglerighteq$ L07 $\gg$	$\mathcal{C}$	;
L10:	Block $\gg$	End	;
L11:	Ded $\triangleright$ L10 $\gg$	$(\mathcal{A} \Rightarrow \mathcal{B} \Rightarrow \mathcal{C}) \Rightarrow (\mathcal{A} \Rightarrow \mathcal{B}) \Rightarrow \mathcal{A} \Rightarrow \mathcal{C}$	□

S **proof of** A4′:

L01:	Block $\gg$	Begin	;
L02:	Premise $\gg$	$x + y = y + x$	;
L03:	Repetition $\triangleright$ L02 $\gg$	$x + y = y + x$	;
L04:	Block $\gg$	End	;
L05:	Ded $\triangleright$ L04 $\gg$	$\forall x{:}\,\forall y{:}\, x + y = y + x \Rightarrow 2 + 3 = 3 + 2$	□

S **proof of** A5′:

L01:	Block $\gg$	Begin	;
L02:	Premise $\gg$	$2 + 3 = 5 \Rightarrow 2 + 3 + x = 5 + x$	;
L03:	Premise $\gg$	$2 + 3 = 5$	;
L04:	L02 $\trianglerighteq$ L03 $\gg$	$2 + 3 + x = 5 + x$	;
L05:	Gen $\triangleright$ L04 $\gg$	$\forall x{:}\, 2 + 3 + x = 5 + x$	;
L06:	Block $\gg$	End	;
L07:	Ded $\triangleright$ L06 $\gg$	$\forall x{:}\,(2 + 3 = 5 \Rightarrow 2 + 3 + x = 5 + x) \Rightarrow 2 + 3 = 5 \Rightarrow \forall x{:}\, 2 + 3 + x = 5 + x$	□

4.2 A proof of $x + y = y + x$

[S **lemma** Prop 3.2a: $\Pi\mathcal{A}{:}\,\mathcal{A} = \mathcal{A}$]

[S **lemma** Prop 3.2b: $\Pi\mathcal{A},\mathcal{B}\colon \mathcal{A}=\mathcal{B}\vdash\mathcal{B}=\mathcal{A}$]

[S **lemma** Prop 3.2c: $\Pi\mathcal{A},\mathcal{B},\mathcal{C}\colon \mathcal{A}=\mathcal{B}\vdash\mathcal{B}=\mathcal{C}\vdash\mathcal{A}=\mathcal{C}$]

[S **lemma** Prop 3.2d: $\Pi\mathcal{A},\mathcal{B},\mathcal{C}\colon \mathcal{A}=\mathcal{C}\vdash\mathcal{B}=\mathcal{C}\vdash\mathcal{A}=\mathcal{B}$]

[S **lemma** Prop 3.2e: $\Pi\mathcal{A},\mathcal{B},\mathcal{C}\colon \mathcal{A}=\mathcal{B}\vdash\mathcal{A}+\mathcal{C}=\mathcal{B}+\mathcal{C}$]

[S **lemma** Prop 3.2f: $\Pi\mathcal{A}\colon \mathcal{A}=0+\mathcal{A}$]

[S **lemma** Prop 3.2g: $\Pi\mathcal{A},\mathcal{B}\colon \mathcal{A}'+\mathcal{B}=(\mathcal{A}+\mathcal{B})'$]

[S **lemma** Prop 3.2h: $\Pi\mathcal{A},\mathcal{B}\colon \mathcal{A}+\mathcal{B}=\mathcal{B}+\mathcal{A}$]

S **proof of** Prop 3.2a:

L01:	Arbitrary $\gg$	$\mathcal{A}$	;
L02:	S5 $\gg$	$\mathcal{A}+0=\mathcal{A}$	;
L03:	S1 $\rhd$ L02 $\rhd$ L02 $\gg$	$\mathcal{A}=\mathcal{A}$	□

S **proof of** Prop 3.2b:

L01:	Arbitrary $\gg$	$\mathcal{A},\mathcal{B}$	;
L02:	Premise $\gg$	$\mathcal{A}=\mathcal{B}$	;
L03:	Prop 3.2a $\gg$	$\mathcal{A}=\mathcal{A}$	;
L04:	S1 $\rhd$ L02 $\rhd$ L03 $\gg$	$\mathcal{B}=\mathcal{A}$	□

S **proof of** Prop 3.2c:

L01:	Arbitrary $\gg$	$\mathcal{A},\mathcal{B},\mathcal{C}$	;
L02:	Premise $\gg$	$\mathcal{A}=\mathcal{B}$	;
L03:	Premise $\gg$	$\mathcal{B}=\mathcal{C}$	;
L04:	Prop 3.2b $\rhd$ L02 $\gg$	$\mathcal{B}=\mathcal{A}$	;
L05:	S1 $\rhd$ L04 $\rhd$ L03 $\gg$	$\mathcal{A}=\mathcal{C}$	□

S **proof of** Prop 3.2d:

L01:	Arbitrary $\gg$	$\mathcal{A},\mathcal{B},\mathcal{C}$	;
L02:	Premise $\gg$	$\mathcal{A}=\mathcal{C}$	;
L03:	Premise $\gg$	$\mathcal{B}=\mathcal{C}$	;
L04:	Prop 3.2b $\rhd$ L03 $\gg$	$\mathcal{C}=\mathcal{B}$	;
L05:	Prop 3.2c $\rhd$ L02 $\rhd$ L04 $\gg$	$\mathcal{A}=\mathcal{B}$	□

[S **lemma** Prop 3.2e$_1$: $\Pi\mathcal{A},\mathcal{B}\colon \mathcal{A}=\mathcal{B}\Rightarrow\mathcal{A}+0=\mathcal{B}+0$]

S **proof of** Prop 3.2e$_1$:

L01:	Arbitrary $\gg$	$\mathcal{A},\mathcal{B}$	;
L02:	Block $\gg$	Begin	;
L03:	Arbitrary $\gg$	$\mathcal{A},\mathcal{B}$	;
L04:	Premise $\gg$	$\mathcal{A}=\mathcal{B}$	;
L05:	S5 $\gg$	$\mathcal{A}+0=\mathcal{A}$	;
L06:	Prop 3.2c $\rhd$ L05 $\rhd$ L04 $\gg$	$\mathcal{A}+0=\mathcal{B}$	;
L07:	S5 $\gg$	$\mathcal{B}+0=\mathcal{B}$	;
L08:	Prop 3.2d $\rhd$ L06 $\rhd$ L07 $\gg$	$\mathcal{A}+0=\mathcal{B}+0$	;

L09:	Block $\gg$	End	;
L10:	Ded $\triangleright$ L09 $\gg$	$\mathcal{A} = \mathcal{B} \Rightarrow \mathcal{A} + 0 = \mathcal{B} + 0$	□

[S **lemma** Prop 3.2e$_2$: $\Pi\mathcal{A}, \mathcal{B}, \mathcal{C}$: $(\mathcal{A} = \mathcal{B} \Rightarrow \mathcal{A} + \mathcal{C} = \mathcal{B} + \mathcal{C}) \Rightarrow (\mathcal{A} = \mathcal{B} \Rightarrow \mathcal{A} + \mathcal{C}' = \mathcal{B} + \mathcal{C}')$]

S **proof of** Prop 3.2e$_2$:

L01:	Arbitrary $\gg$	$\mathcal{A}, \mathcal{B}, \mathcal{C}$	;
L02:	Block $\gg$	Begin	;
L03:	Arbitrary $\gg$	$\mathcal{A}, \mathcal{B}, \mathcal{C}$	;
L04:	Premise $\gg$	$\mathcal{A} = \mathcal{B} \Rightarrow \mathcal{A} + \mathcal{C} = \mathcal{B} + \mathcal{C}$	;
L05:	Premise $\gg$	$\mathcal{A} = \mathcal{B}$	;
L06:	L04 $\trianglerighteq$ L05 $\gg$	$\mathcal{A} + \mathcal{C} = \mathcal{B} + \mathcal{C}$	;
L07:	S2 $\triangleright$ L06 $\gg$	$(\mathcal{A} + \mathcal{C})' = (\mathcal{B} + \mathcal{C})'$	;
L08:	S6 $\gg$	$\mathcal{A} + \mathcal{C}' = (\mathcal{A} + \mathcal{C})'$	;
L09:	Prop 3.2c $\triangleright$ L08 $\triangleright$ L07 $\gg$	$\mathcal{A} + \mathcal{C}' = (\mathcal{B} + \mathcal{C})'$	;
L10:	S6 $\gg$	$\mathcal{B} + \mathcal{C}' = (\mathcal{B} + \mathcal{C})'$	;
L11:	Prop 3.2d $\triangleright$ L09 $\triangleright$ L10 $\gg$	$\mathcal{A} + \mathcal{C}' = \mathcal{B} + \mathcal{C}'$	;
L12:	Block $\gg$	End	;
L13:	Ded $\triangleright$ L12 $\gg$	$(\mathcal{A} = \mathcal{B} \Rightarrow \mathcal{A} + \mathcal{C} = \mathcal{B} + \mathcal{C}) \Rightarrow$ $\mathcal{A} = \mathcal{B} \Rightarrow \mathcal{A} + \mathcal{C}' = \mathcal{B} + \mathcal{C}'$	□

S **proof of** Prop 3.2e:

L01:	Arbitrary $\gg$	$\mathcal{A}, \mathcal{B}, \mathcal{C}$	;
L02:	Premise $\gg$	$\mathcal{A} = \mathcal{B}$	;
L03:	Block $\gg$	Begin	;
L04:	Prop 3.2e$_1$ $\gg$	$x = y \Rightarrow x + 0 = y + 0$	;
L05:	Prop 3.2e$_2$ $\gg$	$(x = y \Rightarrow x + z = y + z) \Rightarrow$ $(x = y \Rightarrow x + z' = y + z')$	;
L06:	S9 @ z $\triangleright$ L04 $\triangleright$ L05 $\gg$	$x = y \Rightarrow x + z = y + z$	;
L07:	Block $\gg$	End	;
L08:	Ded $\triangleright$ L07 $\gg$	$\mathcal{A} = \mathcal{B} \Rightarrow \mathcal{A} + \mathcal{C} = \mathcal{B} + \mathcal{C}$	;
L09:	L08 $\trianglerighteq$ L02 $\gg$	$\mathcal{A} + \mathcal{C} = \mathcal{B} + \mathcal{C}$	□

[S **lemma** Prop 3.2f$_1$: $0 = 0 + 0$]

S **proof of** Prop 3.2f$_1$:

L01:	S5 $\gg$	$0 + 0 = 0$	;
L02:	Prop 3.2b $\triangleright$ L01 $\gg$	$0 = 0 + 0$	□

[S **lemma** Prop 3.2f$_2$: $\Pi\mathcal{A}$: $\mathcal{A} = 0 + \mathcal{A} \Rightarrow \mathcal{A}' = 0 + \mathcal{A}'$]

S **proof of** Prop 3.2f$_2$:

L01:	Arbitrary $\gg$	$\mathcal{A}$	;
L02:	Block $\gg$	Begin	;
L03:	Arbitrary $\gg$	$\mathcal{A}$	;
L04:	Premise $\gg$	$\mathcal{A} = 0 + \mathcal{A}$	;
L05:	S2 $\triangleright$ L04 $\gg$	$\mathcal{A}' = (0 + \mathcal{A})'$	;

L06: S6 $\gg$ $0 + \mathcal{A}' = (0 + \mathcal{A})'$;
L07: Prop 3.2d $\triangleright$ L05 $\triangleright$ L06 $\gg$ $\mathcal{A}' = 0 + \mathcal{A}'$;
L08: Block $\gg$ End ;
L09: Ded $\triangleright$ L08 $\gg$ $\mathcal{A} = 0 + \mathcal{A} \Rightarrow \mathcal{A}' = 0 + \mathcal{A}'$ □

S **proof of** Prop 3.2f:
L01: Arbitrary $\gg$ $\mathcal{A}$;
L02: Block $\gg$ Begin ;
L03: Prop $3.2f_1$ $\gg$ $0 = 0 + 0$;
L04: Prop $3.2f_2$ $\gg$ $x = 0 + x \Rightarrow x' = 0 + x'$;
L05: S9 @ x $\triangleright$ L03 $\triangleright$ L04 $\gg$ $x = 0 + x$;
L06: Block $\gg$ End ;
L07: Ded $\triangleright$ L06 $\gg$ $\mathcal{A} = 0 + \mathcal{A}$ □

[S **lemma** Prop $3.2g_1$: $\Pi\mathcal{A}$: $\mathcal{A}' + 0 = (\mathcal{A} + 0)'$]

S **proof of** Prop $3.2g_1$:
L01: Arbitrary $\gg$ $\mathcal{A}$;
L02: S5 $\gg$ $\mathcal{A}' + 0 = \mathcal{A}'$;
L03: S5 $\gg$ $\mathcal{A} + 0 = \mathcal{A}$;
L04: S2 $\triangleright$ L03 $\gg$ $(\mathcal{A} + 0)' = \mathcal{A}'$;
L05: Prop 3.2d $\triangleright$ L02 $\triangleright$ L04 $\gg$ $\mathcal{A}' + 0 = (\mathcal{A} + 0)'$ □

[S **lemma** Prop $3.2g_2$: $\Pi\mathcal{A}, \mathcal{B}$: $\mathcal{A}' + \mathcal{B} = (\mathcal{A} + \mathcal{B})' \Rightarrow \mathcal{A}' + \mathcal{B}' = (\mathcal{A} + \mathcal{B}')'$]

S **proof of** Prop $3.2g_2$:
L01: Arbitrary $\gg$ $\mathcal{A}, \mathcal{B}$;
L02: Block $\gg$ Begin ;
L03: Arbitrary $\gg$ $\mathcal{A}, \mathcal{B}$;
L04: Premise $\gg$ $\mathcal{A}' + \mathcal{B} = (\mathcal{A} + \mathcal{B})'$;
L05: S2 $\triangleright$ L04 $\gg$ $(\mathcal{A}' + \mathcal{B})' = (\mathcal{A} + \mathcal{B})''$;
L06: S6 $\gg$ $\mathcal{A}' + \mathcal{B}' = (\mathcal{A}' + \mathcal{B})'$;
L07: Prop 3.2c $\triangleright$ L06 $\triangleright$ L05 $\gg$ $\mathcal{A}' + \mathcal{B}' = (\mathcal{A} + \mathcal{B})''$;
L08: S6 $\gg$ $\mathcal{A} + \mathcal{B}' = (\mathcal{A} + \mathcal{B})'$;
L09: S2 $\triangleright$ L08 $\gg$ $(\mathcal{A} + \mathcal{B}')' = (\mathcal{A} + \mathcal{B})''$;
L10: Prop 3.2d $\triangleright$ L07 $\triangleright$ L09 $\gg$ $\mathcal{A}' + \mathcal{B}' = (\mathcal{A} + \mathcal{B}')'$;
L11: Block $\gg$ End ;
L12: Ded $\triangleright$ L11 $\gg$ $\mathcal{A}' + \mathcal{B} = (\mathcal{A} + \mathcal{B})' \Rightarrow \mathcal{A}' + \mathcal{B}' = (\mathcal{A} + \mathcal{B}')'$ □

S **proof of** Prop 3.2g:
L01: Arbitrary $\gg$ $\mathcal{A}, \mathcal{B}$;
L02: Block $\gg$ Begin ;
L03: Prop $3.2g_1$ $\gg$ $x' + 0 = (x + 0)'$;
L04: Prop $3.2g_2$ $\gg$ $x' + y = (x + y)' \Rightarrow x' + y' = (x + y')'$;
L05: S9 @ y $\triangleright$ L03 $\triangleright$ L04 $\gg$ $x' + y = (x + y)'$;

L06:	Block $\gg$	End	;
L07:	Ded $\triangleright$ L06 $\gg$	$\mathcal{A}' + \mathcal{B} = (\mathcal{A} + \mathcal{B})'$	$\square$

[S **lemma** Prop 3.2h$_1$: $\Pi\mathcal{A}$: $\mathcal{A} + 0 = 0 + \mathcal{A}$]

S **proof of** Prop 3.2h$_1$:

L01:	Arbitrary $\gg$	$\mathcal{A}$	;
L02:	S5 $\gg$	$\mathcal{A} + 0 = \mathcal{A}$	;
L03:	Prop 3.2f $\gg$	$\mathcal{A} = 0 + \mathcal{A}$	;
L04:	Prop 3.2c $\triangleright$ L02 $\triangleright$ L03 $\gg$	$\mathcal{A} + 0 = 0 + \mathcal{A}$	$\square$

[S **lemma** Prop 3.2h$_2$: $\Pi\mathcal{A}, \mathcal{B}$: $\mathcal{A} + \mathcal{B} = \mathcal{B} + \mathcal{A} \Rightarrow \mathcal{A} + \mathcal{B}' = \mathcal{B}' + \mathcal{A}$]

S **proof of** Prop 3.2h$_2$:

L01:	Arbitrary $\gg$	$\mathcal{A}, \mathcal{B}$	;
L02:	Block $\gg$	Begin	;
L03:	Arbitrary $\gg$	$\mathcal{A}, \mathcal{B}$	;
L04:	Premise $\gg$	$\mathcal{A} + \mathcal{B} = \mathcal{B} + \mathcal{A}$	;
L05:	S2 $\triangleright$ L04 $\gg$	$(\mathcal{A} + \mathcal{B})' = (\mathcal{B} + \mathcal{A})'$	;
L06:	S6 $\gg$	$\mathcal{A} + \mathcal{B}' = (\mathcal{A} + \mathcal{B})'$	;
L07:	Prop 3.2c $\triangleright$ L06 $\triangleright$ L05 $\gg$	$\mathcal{A} + \mathcal{B}' = (\mathcal{B} + \mathcal{A})'$	;
L08:	Prop 3.2g $\gg$	$\mathcal{B}' + \mathcal{A} = (\mathcal{B} + \mathcal{A})'$	;
L09:	Prop 3.2d $\triangleright$ L07 $\triangleright$ L08 $\gg$	$\mathcal{A} + \mathcal{B}' = \mathcal{B}' + \mathcal{A}$	;
L10:	Block $\gg$	End	;
L11:	Ded $\triangleright$ L10 $\gg$	$\mathcal{A} + \mathcal{B} = \mathcal{B} + \mathcal{A} \Rightarrow \mathcal{A} + \mathcal{B}' = \mathcal{B}' + \mathcal{A}$	$\square$

S **proof of** Prop 3.2h:

L01:	Arbitrary $\gg$	$\mathcal{A}, \mathcal{B}$	;
L02:	Block $\gg$	Begin	;
L03:	Prop 3.2h$_1$ $\gg$	$x + 0 = 0 + x$	;
L04:	Prop 3.2h$_2$ $\gg$	$x+y = y+x \Rightarrow x+y' = y'+x$	;
L05:	S9 @ y $\triangleright$ L03 $\triangleright$ L04 $\gg$	$x + y = y + x$	;
L06:	Block $\gg$	End	;
L07:	Ded $\triangleright$ L06 $\gg$	$\mathcal{A} + \mathcal{B} = \mathcal{B} + \mathcal{A}$	$\square$

References

[DBP96] 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.

[Men87] E. Mendelson. *Introduction to Mathematical Logic*. Wadsworth and Brooks, 3. edition, 1987.