Version 1.3, 3 November 2008

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

   You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

   The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

   In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

   Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

    The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ''GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with... Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

GNU Aris

Edition 1.3, for `aris` version 2.2
28 August 2013

by Ian Dunn

Table of Contents

1 Introduction

This manual is for GNU Aris, a sequential proof program, designed to assist anyone interested in solving logical proofs. Aris supports both propositional and predicate logic, as well as Boolean algebra and arithmetical logic in the form of abstract sequences. It uses a predefined set of both inference and equivalence rules, however gives the user options to use older proofs as lemmas, including Isabelle's Isar proofs.

2 Terms

Biconditional
: A biconditional is a connective that connects two sentences, denoted by '`<->`'. A biconditional claims that '`sentence a if and only if sentence b`' is a new sentence. A biconditional can be inserted in Aris using the key combination *CTRL+5* (see Section 5.3 [Other Key Shortcuts], page 11).

Conclusion
: A conclusion is a sentence that is derived from a combination of other sentences and a rule. A focused conclusion will be highlighted in cyan. A conclusion has a set of references associated with it, which are highlighted in violet. Both of these colors can be changed using customization (see Chapter 7 [Customization], page 21).

Conditional
: A conditional is a connective that connects two sentences, denoted by '→'. A conditional claims that '`if sentence a, then sentence b`' is a new sentence. A conditional can be inserted in Aris using the key combination *CTRL+4* (see Section 5.3 [Other Key Shortcuts], page 11).

Conjunction
: A conjunction is a connective that connects two or more sentences, denoted by '`^`'. A conjunction claims that '`sentence a and sentence b`' is a new sentence. A conjunction can be inserted in Aris using the key combination *CTRL+7* (see Section 5.3 [Other Key Shortcuts], page 11).

Connective
: A connective is a logical symbol that connects one or more sentences. The connectives used in system PSI are conjuction ('`^`'), disjunction ('`v`'), negation, ('`~`'), conditional ('→'), and biconditional ('`<->`'). In addition, system PSI recognizes the one-place connectives of the tautology ('`T`') and the contradiction ('`!`').

Contradiction
: A contradiction is a zero-place connective that stands on its own, denoted by '`!`'. A contradiction represents something that is always false. A contradiction is only used with the boolean rules (see Section 6.4 [Boolean Rules], page 19), and can be inserted using the key combination *CTRL+6*.

Disjunction
: A disjunction is a connective that connects two or more sentences, denoted by '`v`'. A disjunction claims that '`sentence a or sentence b`' is a new sentence. A disjunction can be inserted in Aris using the key combination *CTRL+* (see Section 5.3 [Other Key Shortcuts], page 11).

Evaluate
: To evaluate a sentence means different things depending on the type of sentence. At the very least, evaluation checks the sentence for text errors, i.e. mismatched parenthesis, etc. Evaluating a conclusion checks that the sentence's text logically follows from the given references and its rule. Evaluating a goal means checking the corresponding proof for a sentence with this exact same

text. To evaluate a sentence use the key combination *CTRL+E* (see Section 5.1 [Proof Windows], page 9).

Evaluation Value
: An evaluation value is the value that appears to the right of a sentence's text entry. It can either be a square, a light green check, an X, an X polygon, a pointer box, or a dark green check icon. The square means that the sentence is awaiting evaluation. The light gren check means that either the conclusion logically follows from its references and rule, or that the goal has been found in the corresponding proof. In the case of premises, this means that the premise has no syntactic errors. The red X means that either the conclusion does not logically follow from its references and rule, or that the goal has not been found in the corresponding proof. The X polygon means that there is a text error with this sentence. The green check means that one of the conclusion's references has a text error. The pointer box means that the conclusion is missing a rule.

Existential
: An existential is a quantifier that precedes the rest of sentence, denoted by '`3`'. An existential claims that '`there exists at least on item that has property property P`' is a new sentence, assuming that '`P`' is a valid predicate. An existential quantifier can be inserted into Aris using the key combination *CTRL+3*.

Function Symbol
: A function symbol maps one object to another object. These are always lower case. Examples of a function symbol is seqlog's '`z`' and '`s`' symbols (see Chapter 9 [Sequence Logic], page 24).

Goal
: A goal is a sentence that the user is looking to meet in a certain proof. The goal window contains all of these sentences, and can be toggled by the key combination *CTRL+L* (see Section 5.1 [Proof Windows], page 9). When a sentence in the proof matches a goal, the proof sentence's line number is highlighted in red, while the goal's line number is changed to match the proof sentence's line number.

Negation
: A negation is a connective that is inserted in front of a sentence, denoted by '`~`'. A neagion claims that the opposite of the negation is true. A negation can be inserted into Aris using the key combination *CTRL+'* (see Section 5.3 [Other Key Shortcuts], page 11).

Null Object
: A null object is an object, denoted by '`nil`'. In Aris it resembles a null byte, '`\0`', and represents an undefined object in a sequence. A null object can be inserted by using the key combination *CTRL+.*.

Predicate
: A predicate is a type of logical symbol that denotes a property of or relation between one or more objects. These always begin with capital letters, and generally use prefix notation. Exceptions of this are the identity predicate ('`=`'), the less than predicate ('`<`'), and the element of predicate.

Premise
: A premise is a sentence that is given. A premise has no rule associated with it, nor does it have an evaluation value, unless there is an error in it. Any variables introduced in a premise are not considered arbitrary.

Proof
: A proof is a set of sentences, beginning with a set of premises and ending with a set of conclusions, that the user is trying to derive something from. The proof window is the main window that appears when the user opens up a proof.

Quantifier
: A quantifier is a type of logical symbol that claims something about the amount, or quantity, of an object that holds a specific property. The quantifiers used in system PSI are the universal (`V`), and the existential (`3`).

Reference
: A reference is a sentence that is being used to derive a conclusion. A reference is highlighted in violet, and can be added or removed from the current sentence by holding down *CTRL*, and left-clicking on the desired reference.

Rules
: The rules in Aris are a combination of inference rules, equivalence rules, and predicate rules that the user can use to derive sentences. The rules window (also referred to as the rules tablet) is shared amongst all of the proofs in Aris. It can be toggled by the key combination *CTRL+R* (see Section 5.1 [Proof Windows], page 9) from any proof window. For the list of rules, Chapter 6 [Rules Index], page 13.

Sentence
: A sentence is a line in Aris. A sentence always consists of a text entry, a line number, and an evaluation value.

Subproof
: A proof within a proof. To begin a subproof in Aris, use the key combination *CTRL+B*, and to exit one, use the key combination *CTRL+D*.

Tautology
: A tautology is a zero-place connective that stands on its own, denoted by `T`. A tautology represents something that is always true. A tautology is only used with the boolean rules (see Section 6.4 [Boolean Rules], page 19), and can be inserted using the key combination *CTRL+1*.

Universal
: A universal is a quantifier that precedes the rest of sentence, denoted by `V`. A universal claims that `for all items, they have property P` is a new sentence, assuming that `P` is a valid predicate. A universal quantifier can be inserted into Aris using the key combination *CTRL+2*.

Variable
: A variable represents an object within a proof. A variable is introduced when it is first used. If the line that it is introduced in is a premise, the start of a subproof, or a line using existential instantiation (see Section 6.3.4 [ei], page 18), then it is **not** considered arbitrary. Otherwise, it is. Only the variables from lines that can be selected from the current line are taken into account when processing it. This means that after a subproof is ended, then lines after it don't worry about variables introduced within it.

3 Options

'-a VARIABLE'
'--variable=VARIABLE'
: Use VARIABLE as a known variable in evaluation mode. Prepend an '*' to specifiy that the variable is arbitrary.

'-b'
'--boolean'
: Start Aris in boolean mode.

'-c CONCLUSION'
'--conclusion=CONCLUSION'
: Use CONCLUSION as a conclusion in evaluation mode. This flag can only be specified once.

'-e'
'--evaluate'
: Run Aris in evaluation mode. This means that no GUI will be loaded.

'-f FILE'
'--file=FILE'
: Evaluate FILE if running Aris in evaluation mode, otherwise load FILE in Aris. This flag can be specified multiple times.

'-g'
'--grade'
: Grades a file specified by the file flag. This flag is ignored if used more than once.

'-l'
'--list'
: List the rules available in Aris, and exit.

'-p PREMISE'
'--premise=PREMISE'
: Use PREMISE as a premise in evalution mode. This flag can be specified multiple times.

'-r RULE'
'--rule=RULE'
: Use RULE as a rule in evaluation mode. This flag can only be specified once.

'-t TEXT'
'--text=TEXT'
: Simply check the correctness of TEXT in evaluation mode.

'-v'
'--verbose'
: Run Aris verbosely, printing status and error messages.

'-x'
'--latex=FILE'
: Convert FILE to a LaTeX proof file in evaluation mode.

'`--version`'
: Print the version of Aris and exit.

'`-h`'
'`--help`' Print a help message and exit.

4 Basic Usage

This chapter describes the basic usage of GNU Aris.

4.1 Startup

When Aris is loaded up, you will see a few things. You will see the rules window and a proof window. The rules window contains the different rules. A rule will not be selected if a premise is in focus.

The initial layout of Aris is a single sentence. From left to right, the items of a sentence are: its line number, its text entry, its evaluation value, and its rule. The rule will not be initially visible, since no rule has been selected. In addition, premises do not have rules, and thus the rule will not appear.

4.2 Connectives

Aris has several connectives (see Chapter 2 [Terms], page 2) When the keyboard command for the desired connective is activated, the desired connective will be inserted at the current cursor point, overwriting selected text.

For example, pressing *Ctrl+7* inserts a conjunction. This is the character that looks like an upside-down 'v'. This is also called a 'logical and'.

4.3 Adding Sentences

Hitting *Ctrl+J* adds a conclusion to the proof. A conlusion is always added after the current line, or, if the line is a premise, then the conclusion is added after the last premise. If the current line is a conclusion, then it is highlighted in cyan.

Hitting *Ctrl+P* adds a premise to the proof. A premise will always be added after the last premise.

Pressing *Ctrl+B* adds a subproof to the proof. When in a subproof, a conclusion will always be added within the subproof. The first line of a subproof does not require a rule, but instead acts as a premise. Pressing *Ctrl+D* ends the current subproof. This creates a new conclusion just after the subproof.

To undo a command, simply press *Ctrl+Z*. This will undo the last text modification, insertion, or deletion. In the case of several insertions or several deletions, undo will undo all of them. To undo an undo, press *Ctrl+Y*, or redo.

4.4 Selecting Sentences

Holing *CTRL*, and left-clicking on a sentence will select a sentence as a reference sentence. The current line's reference sentences are highlighted in violet. Only a line before the current line can be selected as a reference. In addition, if the sentence is in a different subproof than the current line, then the sentence can not be selected as a reference. Each rule requires a specific amount of references (see Chapter 6 [Rules Index], page 13).

Holding *SHIFT* and left-clicking on a sentence will select the sentence. The sentence will be highlighted in red-orange. Multiple sentences can be selected this way, however when another action is taken, all of them will be de-selected. Pressing *CTRL+K* will kill (cut) the selected lines, and *CTRL+G* will copy the selected lines. If no lines are selected, then the current line will be used.

4.5 Aris Syntax

Aris expects a certain form of syntax. Most of the connectives are infix, which means that they are placed in between their arguments. The negation is one exception to this.

Aris expects that all predicates start with an uppercase character. Aris also expects that all function symbols begin with a lower case character. After this, Aris will except any combination of upper case letters, lower case letters, numbers, or '`_`'.

To assist in understanding the proofs, Aris also allows for comments in sentences. To make a comment, simply insert a '`;`'. Aris will ignore everything after a '`;`' when evaluating a sentence. This way, plaintext can be written into sentences.

5 Menu Options

The main GUIs for Aris all have menu bars. Each of the three types of GUIs have different menu bars, and the options for each of these are described in this section.

The keyboard shortcuts here (with the exception of the connectives) can be changed using customization See Chapter 7 [Customization], page 21. The shortcuts listed here are the default shortcuts for Aris.

5.1 Proof Windows

These are the menu options for the main proof windows. Each one can be assigned a key command.

'`New`'
'`Ctrl+N`' Start a new proof. A new window is opened for this proof.

'`Open`'
CTRL+O Open an existing proof in a new window.

'`Save`'
CTRL+S Save the current proof.

'`Save As`'
CTRL+SHIFT+S
Save the current proof under a different name.

'`Export to LaTeX...`'
Export the current proof to a LaTeX file.

'`Close`'
CTRL+W Close the current proof.

'`Quit`'
CTRL+Q Exit Aris. But since logic is so much fun, I doubt you'll ever want to use this one.

'`Add Premise`'
CTRL+P Insert a new premise at the end of the other premises.

'`Add Conclusion`'
CTRL+J Insert a new conclusion after the current line if it is a conclusion, or at the start of the conclusions if it is a premise.

'`Add Subproof`'
CTRL+B Begin a new subproof after the current line if it is a conclusion, or at the start of the conclusions if it is a premise. This is unavailable in boolean mode, since subproofs can't be used.

'`End Subproof`'
CTRL+D End the current subproof, if there is one. Otherwise, this doesn't do anything. This is unavailable in boolean mode, since subproofs can't be used.

'`Undo`'
CTRL+Z Undo the last modification to the current proof. On a new file, this does nothing.

'Redo'
CTRL+Y Undo an undo operation. If no undo has been made, then this does nothing.

'Copy Line'
CTRL+G Copy the current line.

'Kill Line'
CTRL+K Kill, or cut, the current line. This removes the line from the proof.

'Insert Line'
CTRL+I Insert the copied/killed line.

'Evaluate Line'
CTRL+E Evaluate the logical validity of the current line.

'Evaluate Proof'
CTRL+F Evaluate the logical validity of the current proof. This evaluates each line of the proof.

'Toggle Goals...'
CTRL+L Toggle the goal window for the current proof.

'Toggle Boolean Mode'
CTRL+M Toggle boolean mode for the current proof See Section 6.4 [Boolean Rules], page 19.

'Import Proof'
Import another proof into this one. This will merge the premises of the other proof into the current one, and insert the goals of the other proof as conclusions. In addition, it sets the conclusions' references as the premises, and sets them all to use the lemma rule see Section 6.5.1 [lm], page 20.

'Toggle Rules'
CTRL+R Toggle the rules window.

'Small'
CTRL+- Change the font size to small (8pt).

'Medium'
CTRL+0 Change the font size to medium (12pt).

'Large'
CTRL+= Change the font size to large (16pt).

'Custom' Change the font size to a custom size. This menu option opens a dialog box with a numerical entry.

'Contents'
F1 Display Aris help. This is the only key command that cannot be modified.

'About GNU Aris'
Displays information about GNU Aris.

5.2 Rules Table

These are the commands for the rules window. Many of them are the same as for the main proof window.

`New`
`Ctrl+N`
: Start a new proof. A new window is opened for this proof.

`Open`
CTRL+O
: Open an existing proof in a new window.

`Submit Proofs...`
: Submits all open proofs for grading. There is more on this in the Submission session in this manual See Chapter 8 [Submission], page 23.

`Quit`
CTRL+Q
: Exit Aris.

`Small`
CTRL+-
: Change the font size to small (8pt).

`Medium`
CTRL+0
: Change the font size to medium (12pt).

`Large`
CTRL+=
: Change the font size to large (16pt).

`Custom`
: Change the font size to a custom size. This menu option opens a dialog box with a numerical entry.

`Contents`
F1
: Display Aris help. This is the only key command that cannot be modified.

`Customize...`
: Opens the customization dialog. For more information on this, see See Chapter 7 [Customization], page 21.

`About GNU Aris`
: Displays information about GNU Aris.

5.3 Other Key Shortcuts

These are the keyboard shortcuts for each of the connectives. Unlike most of the other keyboard shortcuts, these cannot be modified.

CTRL+7 Insert a conjunction (`^`) into Aris.

CTRL+ Insert a disjunction (`v`) into Aris.

*CTRL+`* Insert a negation (`-`) into Aris.

CTRL+4 Insert a conditional (‘→’) into Aris.

CTRL+5 Insert a biconditional (`<->`) into Aris.

CTRL+2 Insert a universal (`V`) into Aris.

CTRL+3 Insert an existential (`3`) into Aris.

CTRL+6 Insert a tautology (‘T’) into Aris.

CTRL+1 Insert a contradiction (‘!’) into Aris.

CTRL+; Insert an ’element of’ predicate into Aris.

CTRL+. Insert a null object (‘nil’) into Aris.

6 Rules Index

The rules are divided into five categories: Inference, Equivalence, Predicate, Boolean, and Miscellaneous.

6.1 Inference Rules

The premises of any of these rules can be in any order.

6.1.1 Modus Ponens

P -> Q
P
———
Q

One of the basic rules of logic, modus ponens say that '`if P happens, then Q must happen. P happened, so Q must happen`'.

For example, if it is known that '`If the dog begins to bark, then someone is at the door`', and it is also known that '`the dog has begun to bark`', then modus ponens says that '`someone must be at the door`'.

Modus Ponens requires exactly two references.

6.1.2 Addition

P
———
P v Q v R v ...

What addition says is that something is already known, so it must be true that that something or something else, or something else, etc. must also be true.

For example, if it is known that '`The sky is blue`', then addition says that it can be inferred that '`The sky is blue, or the sky is yellow, or the sky is pink`', since only one of those statements has to be true.

Addition requires exactly one reference.

6.1.3 Simplification

P ^ Q ^ R ^ ...
———
P (or Q, or R, or ...)

Simplification says that if it is known that P and Q and R, etc. is known to be true, then P is true.

For instance, if it is known that '`It is cloudy, and it is raining`', then simplification allows the inference of '`It is cloudy`' and '`It is raining`'.

Simplification requires exactly one reference.

6.1.4 Conjunction

P

Q

R

———

P ^ Q ^ R

What conjunction is saying is the exact opposite of simplification. If P is known, and Q is known, and R is know, etc. then P and Q and R, etc. is also known.

Take for example, that it is known that '`I don't like green eggs and ham`', and '`I would not eat them in a house`', and '`I would not eat them with a mouse`'. Conjunction allows us to infer that '`I don't like green egss and ham, and I would not eat them in a house, and I would not eat them with a mouse.`'.

Conjunction requires at least two references.

6.1.5 Hypothetical Syllogism

$P \rightarrow Q$

$R \rightarrow S$

$Q \rightarrow R$

———

$P \rightarrow S$

Also referred to as the chain rule, hypothetical syllogism states that if one knows that 'if P then Q', and 'if R then S', then one can infer 'if P then S'. For example, if it is known '`if it is raining, then it is cloudy`', and '`if it is cloudy, then it is not sunny`', and '`if it is not sunny, then it is cold`', then hypothetical syllogism allows us to infer that '`if it is raining, then it is cold`'. This works with any number of conditional statements, as long as they all follow this pattern.

Hypothetical Syllogism requires at least two references.

6.1.6 Disjunctive Syllogism

~P

P v Q v R

~R

———

Q

Disjunctive syllogism is commonly used when disjunctions are present. It claims that if one knows that 'P or Q or R', and 'P is false', and 'R is false', then Q must be true. This works with any number of disjuncts.

6.1.7 Excluded Middle

———

P v ~P

A law of logic, excluded middle asserts that something is either true, or it is not true.

Excluded middle requires zero references.

6.1.8 Constructive Dilemma

P → R

P v Q

Q → S

———

R v S

Constructive Dilemma requires at least three references.

6.2 Equivalence Rules

Equivalence rules operate on any valid part of the sentence, and work both ways. Each equivalence rule requires one reference.

6.2.1 Implication

P → Q <=> ~P v Q

Implication uses the definition of the conditional. It is also valid to claim something such as ~(~P v Q) v (~R v S) <=> (P → Q) → (R → S), because implication is recursive.

6.2.2 DeMorgan

~(P ^ Q) <=> ~P v ~Q

~(P v Q) <=> ~P ^ ~Q

~3x(P(x)) <=> Vx(~P(x))

~Vx(P(x)) <=> 3x(~P(x))

DeMorgan's Laws.

6.2.3 Association

P ^ (Q ^ R) <=> P ^ Q ^ R

P v (Q v R) <=> P v Q v R

A note to users, typically association is used as P ^ (Q ^ R) <=> (P ^ Q) ^ R. While Aris will allow you to prove that this is equivalent, association allows the removal of one pair of parentheses at a time. (P ^ Q) ^ (R ^ S) <=> P ^ Q ^ R ^ S is also valid in Aris, because association allows recursion, but only when removing several sets of parentheses or adding several sets of parentheses.

6.2.4 Commutativity

P ^ Q ^ R <=> Q ^ R ^ P

P v Q v R <=> Q v R v P

Just like addition and multiplication, conjunctions and disjunctions are commutative. This of course means that '`I would like some pie and I would like some cake`' is the same as saying '`I would like some cake and I would like some pie`'.

6.2.5 Idempotence

P ^ P ^ Q ^ R ^ R ^ R <=> P ^ Q ^ R

P v P v Q v R v R v R <=> P v Q v R

Idempotence claims that '`I like blue and I like blue`' is the same as saying '`I like blue`'.

6.2.6 Distribution

P ^ (Q0 v Q1 v ... v Qn) <=> (P ^ Q0) v (P ^ Q1) v (P ^ Q2) v ... v (P ^ Qn)

P v (Q0 v Q1 ^ ... ^ Qn) <=> (P v Q0) ^ (P v Q1) ^ (P v Q2) ^ ... ^ (P v Qn)

3x(P(x) v Q(x)) <=> 3x(P(x)) v 3x(Q(x))

Vx(P(x) ^ Q(x)) <=> Vx(P(x)) ^ Vx(Q(x))

6.2.7 Equivalence

P <-> Q <=> (P → Q) ^ (Q → R)

Equivalence uses the definition of the biconditional. Claiming that '`P if and only if Q`' is exactly the same as claiming '`if P then Q`' and '`if Q then P`'. Equivalence is the only rule that works with biconditionals explicitly, and is thus used any time a biconditional is seen.

6.2.8 Double Negation

~~P <=> P

You probably learned in english class that saying '`I would not like to disagree`' is the same thing as saying '`I would like to agree`'. That's what double negation claims.

6.2.9 Exportation

(P ^ Q) → R <=> P → (Q → R)

This is one of the few equivalence rules that deals with conditionals.

6.2.10 Subsumption

P ^ (P v Q) <=> P

P v (P ^ Q) <=> P

Also called absorption. This rule can be used in Boolean mode.

6.2.11 Recursion in the Equivalence Rules

For the convenience of the user, the equivalence rules work recursively. For example

~(~P v Q) v (~R v S)

———

(P → Q) → (R → S)

This is an example of using implication recursively. Recursion only works if the rule is being used the same way. For example, removing multiple parentheses with association is fine, however adding and removing parentheses with association is not.

Commutatvitity and idempotence work differently than the others when it comes to recursion. If commutativity is applied to a connective, then no parts of that connective, or parts of those parts, and so on, can be used in commutativity. However, other parts from the sentence can be rearranged. The same goes for idempotence.

6.3 Predicate Rules

The predicate rules are the rules that work specifically with predicate logic.

6.3.1 Universal Generalization

P(a) ; a is arbitrary

———

Vx(P(x))

Universal Generalization claims that if a property ‘`P`’ is true for some arbitrary object, then it is true for all objects. A symbol is arbitrary if nothing is known about, or rather if it was not introduced through a premise or using existential instantiation. Chapter 2 [Terms], page 2

Universal Generalization uses exactly one reference.

6.3.2 Universal Instantiation

Vx(P(x))

———

P(a)

Universal Generalization claims that if a property ‘`P`’ is true for all objects, then it must be true for an object ‘`a`’.

Universal Generalization uses exactly one reference.

6.3.3 Existential Generalization

P(a)

———

3x(P(x))

Existential Generalization claims that if ‘`P`’ is true for some object, then there exists an object for which ‘`P`’ is true.

Existential Generalization uses exactly one reference.

6.3.4 Existential Instantiation.

3x(P(x))

———

P(a) ; a must not have been used before

Existential Instantiation claims that if there exists an object for which property ‘`P`’ is true, then it can be claimed that some unused object has this property. In this case, ‘`a`’ becomes a placeholder for the object. Chapter 2 [Terms], page 2

Existential Instantiation uses exactly one reference.

6.3.5 Bound Variable

Vx(P(x)) <=> Vy(P(y))

3x(P(x)) <=> 3y(P(y))

Bound Variable allows the user to substitute any bound variable for another bound variable, given that the second bound variable does not appear anywhere in the scope of the quantifier of the first bound variable. For example, if it is known that Vx(Vy(P(x) ^ P(y))), an invalid use of bound variable would be to state that Vx(Vx(P(x) ^ P(x))).

As an equivalence rule, bound variable uses only one reference, and can work on any part of the sentence.

6.3.6 Null Quantifier

Vx(P(a)) <=> P(a)

If a quantifier’s bound variable does not appear in its scope, then the quantifier is said to be null, and can be removed using Null Quantifier.

As an equivalence rule, null quantifier can be used on any part of the sentence, and only uses one reference.

6.3.7 Prenex

3x(P(x) ^ Q(a)) <=> 3x(P(x)) ^ Q(a)

Vx(P(x) ^ Q(a)) <=> Vx(P(x)) ^ Q(a)

3x(P(x) v Q(a)) <=> 3x(P(x)) v Q(a)

Vx(P(x) v Q(a)) <=> Vx(P(x)) v Q(a)

The Prenex Laws are used to move quantifiers to the start of the sentence.

Prenex uses only one reference, and, being an equivalence rule, can be used on any part of the sentence.

6.3.8 Identity

———

a = a

Identity asserts that any variable is identical to itself.

Identity does not use any references.

6.3.9 Free Variable

a = b

P(a)

———

P(b)

Free Variable allows the user to substitute a free variable for another free variable, given that the two are identical.

Free Variable uses exactly two references.

6.4 Boolean Rules

Aris can be set to use 'boolean mode', a mode used for boolean algebra. In boolean mode, only equivalence rules that handle negations, conjunctions, or disjunctions and boolean rules can be used. In standard mode, the boolean rules can still be used, however.

6.4.1 Boolean Identity

A ^ T <=> A

A v ! <=> A

Boolean Identity claims that the conjunction of a sentence with a tautology is logically equivalent to the sentence. It also claims that the disjunction of a sentence an a contradiction is logically equivalent to the sentence.

6.4.2 Boolean Negation

A ^ ~A <=> !

A v ~A <=> T

Boolean Negation claims that the conjunction of a setentence and its contradiction is a contradiction, and the disjunction of a sentence and its negation is a tautology.

6.4.3 Boolean Dominance

A ^ ! <=> !

A v T <=> T

Boolean Dominance claims that the conjunction of a sentence and a contradiction is logically equivalent to a contradiction. It also claims that the disjunction of a sentence and a tautology is logically equivalent to a tautology.

6.4.4 Symbol Negation

~T <=> !

~! <=> T

Symbol Negation claims that a tautology is the opposite of a contradiction.

6.5 Miscellaneous Rules

6.5.1 Lemma

This handy little rule allows one to use proofs one has already done. The premises don't have to match exactly, but they must be of the same form. Aris will check for each symbol it recognizes (connectives, quantifiers, parentheses, comma, and identity). These symbols must match exactly. Aris will then check that the sentences match the correct form, or rather that they appear in the correct order.

For example, if you already did a proof of the form:

A <-> B

A

———

B

And want to reuse it, then your reference sentences must be in the form '`A`' `<->` '`B`', and '`A`'. In general, they do not have to be in that order, however. Then, your conclusion must be the second half of the biconditional.

This is where Isar interoperability comes in. Instead of selecting a previous Aris proof, a .thy file can be used. Aris will attempt to translate it into a form that it can use, using most of the keywords as references, and the lemmas and theorems as goals. These are the sentences that can be proved. For more information, see Section 10.1 [Isabelle/Isar], page 25.

6.5.2 Subproof

Given a subproof with premise 'P' and conclusion (the LAST sentence) 'Q', one can infer from subproof 'P → Q'. In some circles, this is called conditional introduction.

6.5.3 Sequence

This introduces a new sequence given a function. The sequence introduced this way must not have been used, and the final argument of the given function must be the bound variable of the sentence.

6.5.4 Induction

P(z(a))

P(x) → P(s(x))

———

Vx(P(x))

This rule is how Aris implements mathematical induction. '`P(z(x))`' is the base case, and the inductive step is '`P(x)`' → '`P(s(x))`'.

7 Customization

GNU Aris can be customized using the customization dialog, or manually through the customization file. The customization dialog can be accessed through the rules table, under 'Help'.

7.1 Customization Dialog

When the dialog appears, there are several tabs. These are explained as follows:

Main Keys
: This tab allows the user to customize the keyboard shortcuts that Aris uses with the proof and rules table menus. Each entry corresponds to a menu item. The only option that can not be edited is the 'Contents' menu keyboard shortcut.

Goal Keys
: This tab allows the user to customize the keyboard shortcuts that Aris uses with the goal menus.

Display
: This tab allows the user to customize the display settings. Included in here are the font size presets, which will be set to the indicated size when activated; the default font size, which Aris will be in when loaded; and the color preferences, which will change the colors that Aris hilights different objects in.

Grade Server
: This tab allows the user to set preferences specific to the grade server. The two options here are for the IP address of the grading server, and the password used to authenticate into the server. (see Chapter 8 [Submission], page 23)

For a description of the format of key commands, see Section 7.2 [Config File], page 21.

7.2 Customization File

The config file uses s-expressions to store the customization file. It is stored under the home directory, and called '`.aris`'. There are several key words that it recognizes:

'`key-cmd`'
'`(key-cmd 'cmd' 'key')`'
: Assigns menu '`cmd`' to keyboard shortcut '`key`'. The keyboard shortcuts are all in the same format, which is either *s* or *c*, a plus sign, then a letter. A *c* before the letter means '`Hold control, and press the key`', and *s* means the same except with the shift key.

'`font-size`'
'`(font-size 'type' size)`'
: Assigns '`size`' to font type '`type`'. The '`type`' key word can be either '`Small`', which means set the small font preset, '`Medium`', which means set the medium font preset, '`Large`', which means set the large font preset, or '`Default`', which means set the font size that Aris loads up with intially.

'`color-pref`'
'`(color-pref 'type' color)`'
: Assigns '`color`' to color preference '`type`'. The '`color`' key word is in hexidecimal.

'grade'
'(grade 'key' 'value')'
: This allows customization of the grade server's information (see Chapter 8 [Submission], page 23). The two options for 'key' are 'ip' and 'pass'. The 'ip' key sets the grade server's IP address, and the 'pass' sets up the password GNU Aris will use for authentication.

8 Proof Submission

GNU Aris allows users to submit their proofs to a grading server, which allows instructors to use Aris in their classes. Aris submits proofs through FTP, and allows users to indicate an email for themselves and an optional email for their instructors.

Submission allows for all open proofs to be submitted. This is done by specifying a problem designation (‘11.10’, ‘nats’, etc.). Only those that have designations are submitted. The designation is changed by editing the text box next to each file name in the submission dialog box.

Grading runs by checking the correctness of the entire proof. It is the responsibility of the grading server to confirm that the proof is the correct proof.

When the files are submitted, Aris submits them to the server, along with a directive file. The directive file will be named ‘USER.directive’, where ‘USER’ is the base name of the email address specified. The files submitted will be renamed to be ‘BASE-USER.tle’, where ‘BASE’ is the original basename of the file. This prevents filename conflicts on the FTP server. The grading server will then run Aris in grade mode (see Chapter 3 [Options], page 5), and use the email provided to email the results back to the user. If the user specified an instructor’s email address, then Aris will CC the instructor.

Sample scripts are included with GNU Aris. These are the files ‘doc/collect.sh’ and ‘doc/collect.el’. The grade server would run ‘collect.sh’, which will call GNU Emacs in batch mode while loading ‘collect.el’. It is ‘collect.el’ that handles the emails.

9 Sequence Logic

Sequence Logic, often abbreviated 'seqlog', is an alternative arithmetical representation system from the standard Peano Axioms in First-Order Logic. Seqlog's original purpose was allowing more natural definition of recursive functions in FOL.

Seqlog uses the symbols 's' (the sucessor function), 'z' (the zero function), 'v' (the value function), and '\0' (null object).

9.1 Axioms

Sequence Logic, often abbreviated 'seqlog', uses the following six axioms:

- VxVy(˜s(x) = z(y))
- VxVy(s(x) = s(y) → x = y)
- Vx(v(S,x) = f_S(x))
- Vx(v(\0,x) = \0)

The first axiom states that no sucessor is the zero object, or, to put it differently, that the zero object is the first object. The second axiom states that no two different objects have the same sucessor. Using these two axioms, a 'Universal Sequence' can be defined, in a way similar to how the Peano Axioms define the natural numbers. The third axiom is the definition of a sequence, stating that the value under a given sequence 'S' of every object 'x' can be determined by a function. The rule 'sq' introduces such a sequence (see Section 6.5.3 [sq], page 20). The fourth axiom defines the nil object. This is a lot like '`NULL`' in C, or '`nil`' in lisp.

The natural numbers are defined as a sequence. For example, Vx(x = nat → VyVz(v(nat,y) = v(nat,z) → y = z) ˆ Vy(˜v(nat,y) = \0)). This will define an infinite sequence (2nd part), that is one-to-one (1st part). Then, to define zero, one simply states Vx(v(nat,z(x)) = 0). This means that the zero'th element of the '`nat`' sequence is the object '`0`'.

9.2 Induction

Mathematical induction requires a base case, and an inductive step. In Aris, this is used in conjunction with seqlog. For seqlog, the induction scheme is:

Vx(P(z(x)) ˆ (P(x) → P(s(x)))) → Vx(P(x))

10 Interoperability

In addition to everything else Aris can do, Aris can also use other proofs from other systems with the lemma rule (see Section 6.5.1 [lm], page 20).

10.1 Isar Interoperability

Aris will scan an Isar proof, which is a proof done using Isabelle, and look for certain keywords. This is still being tested, and doesn't work fully yet. This section will be updated as more of this is implemented.

10.1.1 fun keyword.

Standard definition of a function in seqlog.

10.1.2 type_synonym keyword

10.1.3 lemma and theorem keywords

Lemmas and theorems are treated the same. Lemmas end up as the goals of the proofs that Aris creates, and are the actual sentences that can be deduced. It takes the 'if-then' form of each lemma.

10.1.4 case keyword

10.1.5 primrec keyword

10.1.6 definition keyword

10.1.7 datatype keyword

10.1.8 class keyword

10.1.9 instance keyword

10.1.10 everything else