GNU MPRIA

The GNU Multi-Precision Rational Interval Arithmetic Library
Edition 0.7.1 for Release 0.7.1
3 February 2014

Jérôme Benoit
'bug-mpria@gnu.org'

This manual describes how to install and use the GNU Multi-Precision Rational Interval Arithmetic Library, release 0.7.1. Please report any errors in this manual to 'bug-mpria@gnu.org'.

More information about the GNU MPRIA Library can be found at the project homepage, http://www.gnu.org/software/mpria/.

Table of Contents

MPRIA Copying Conditions

The GNU MPRIA Library (or MPRIA for short) is *free software*: this means that everyone is free to use it and free to redistribute it on a free basis. The library is not in the public domain; it is copyrighted and there are restrictions on its distribution, but these restrictions are designed to permit everything that a good cooperating citizen would want to do. What is not allowed is to try to prevent others from further sharing any version of this library that they might get from you.

Specifically, we want to make sure that you have the right to give away copies of the library, that you receive source code or else can get it if you want it, that you can change this library or use pieces of it in new free programs, and that you know you can do these things.

To make sure that everyone has such rights, we have to forbid you to deprive anyone else of these rights. For example, if you distribute copies of the GNU MPRIA Library, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must tell them their rights.

Also, for our own protection, we must make certain that everyone finds out that there is no warranty for the GNU MPRIA Library. If it is modified by someone else and passed on, we want their recipients to know that what they have is not what we distributed, so that any problems introduced by others will not reflect on our reputation.

The precise conditions of the license for the GNU MPRIA Library are found in the General Public License version 3 that accompanies the source code, see `COPYING`. A copy of the license is also included in Appendix B [GNU General Public License], page 23.

1 Introduction to MPRIA

1.1 Description

GNU MPRIA is intended to be a portable mathematical library written in C for rational interval arithmetic computations with arbitrary precision.

The basic principle of rational interval arithmetic consists in enclosing every number by a rational interval containing it: each number is stored as its lower and upper endpoints and these bounds are rational numbers; their absolute difference measures the precision. The purpose is on the right hand to obtain guaranteed results, thanks to interval computation, and on the left hand to compute accurate results, thanks to arbitrary precision arithmetic.

The arithmetic operations are extended for interval operands in such a way that the exact result of the operation belongs to the computed rational interval.

The GNU MPRIA library is built upon the GNU MP library for operating on rational numbers; see

> `http://gmplib.org/`.

1.2 Up-to-date Material

The latest information about the library can be found at the project homepage

> `http://www.gnu.org/software/mpria/`,

while the primary distribution point for stable releases is at

> `ftp://ftp.gnu.org/gnu/mpria/`.

Many sites around the world mirror '`ftp.gnu.org`', please use a mirror near you; for a full list, see

> `http://www.gnu.org/order/ftp.html`.

1.3 Mailing Lists

There are three public mailing lists of interest: one for release announcements, one for general questions and discussions about usage of the GNU MPRIA Library and one for bug reports. For more information, visit

> `http://lists.gnu.org/mailman/listinfo/bug-mpria/`.

The proper place for bug reports is '`bug-mpria@gnu.org`'. See Chapter 3 [Reporting Bugs], page 5, for information about reporting bugs.

1.4 How to use this Manual

Everyone should read Chapter 4 [MPRIA Basics], page 6. If you need to install the library yourself, then read Chapter 2 [Installing MPRIA], page 3. To use the library you will need to refer to Chapter 5 [Rational Interval Functions], page 12; for more advanced usage you want to peruse Chapter 6 [Low-Level Rational Interval Functions], page 17.

The rest of the manual can be used for later reference, although it is probably a good idea to glance through it.

2 Installing MPRIA

2.1 How to Install

For a generic installation of the MPRIA library, you have first to install a recent version of the GNU MP on your computer. You need a C compiler, preferably `gcc`, but any reasonable C compiler should work. And you need the standard Unix `make` command, plus some other standard Unix utility commands.

Then, in the MPRIA build directory, type the following commands.

1. '`./configure`'

   This will prepare the build and setup the options according to your system. You can give options to specify the install directories (instead of the default `/usr/local`), threading support, and so on. See the `INSTALL` file or the output of '`./configure --help`' for detailed information, in particular if you get error messages.
2. '`make`'

   This will compile MPRIA and create library files with respect to your platform and environment.
3. '`make check`'

   This will make sure MPRIA was built correctly. If you get error messages, please send a bug report to '`bug-mpria@gnu.org`'. See Chapter 3 [Reporting Bugs], page 5, for information about reporting bugs.
4. '`make install`'

   This will copy the C header file `mpria.h` to the '`include`' directory `/usr/local/include`, the library files (as the share object file `libmpria.so` on GNU/Linux computers) to the '`lib`' directory `/usr/local/lib`, possibly the file `mpria.info` to the '`info`' directory `/usr/local/share/info`, and some other documentation files into the document folder `/usr/local/share/doc/mpria` (or, if you passed the `--prefix` option to `configure`, using the prefix directory given as argument to `--prefix` instead of `/usr/local`).

2.2 Other '`make`' Targets

There are some other useful '`make`' targets:

- '`mpria.info`' or '`info`'

  Create or update an info version of the manual, in `mpria.info`; this file is already provided in the MPRIA source tarball.
- '`mpria.pdf`' or '`pdf`'

  Create a PDF version of the manual, in `mpria.pdf`; this file is already provided in the MPRIA source tarball.
- '`mpria.dvi`' or '`dvi`'

  Create a DVI version of the manual, in `mpria.dvi`.
- '`mpria.ps`' or '`ps`'

  Create a PostScript version of the manual, in `mpria.ps`.
- '`mpria.html`' or '`html`'

  Create a HTML version of the manual, in several pages in the folder `doc/mpria.html`; to obtain one single page HTML document, type '`makeinfo --html --no-split mpria.texi`' from the '`doc`' directory instead.

- 'clean'
  Delete all object files and archive files, but not the configuration files.
- 'distclean'
  Delete all generated files not included in the distribution.
- 'uninstall'
  Delete all files copied by 'make install'.

2.3 Known Build Problems

The installation procedure and the GNU MPRIA library itself have been only tested in some Unix-like environments. Because it has not been yet intensively tested, you may discover that the GNU MPRIA library suffers from all bugs of the underlying GNU MP library, plus many many more.

Please report any problem to 'bug-mpria@gnu.org'. See Chapter 3 [Reporting Bugs], page 5, for information about reporting bugs.

2.4 Getting the Latest Version

The latest stable version of MPRIA is available from

ftp://ftp.gnu.org/gnu/mpria/.

3 Reporting Bugs

If you think you have found a bug in the MPRIA library, please investigate it and report it. Likewise, if you think you have figure out a valuable enhancement for the MPRIA library, please mature it and suggest it. This library has been made available to you: it is expected you will report the bugs that you find or you will suggest the enhancements that you wish.

For bug reports, please include enough information to reproduce the problem. Generally speaking, that means:

- The MPRIA library version, along with the involved GMP library version.
- A test case that makes it possible to reproduce the bug; do not forget to include instructions on how to run the test case.
- A description of what goes wrong; please clearly explain what is incorrect and in what way, whether or not you get a crash.
- Options given to `configure` other than specifying installation directories.
- The output from running '`./configure`', as printed to `stdout`, with any options used.
- The name of the involved compiler and its version; for `gcc`, get the version with '`gcc -v`', otherwise perhaps '``what `which cc` ``', or similar.
- Hardware and operating system names, versions and details; the output from '`uname -a`' along with the output from running '`./build-aux/config.guess`' should be sufficient.
- If the bug is related to `configure`, then attach the compressed contents of `config.log`.
- Anything else that you think would be helpful; when in doubt whether something is needed or not, include it since it is better to include too much than to leave out something important.

If your bug report is good, I will do my best to help you to get a corrected version of the library; if the bug report is poor, I will not do anything about it (aside of chiding you to send better bug reports).

Patches are welcome; if possible, please make them with '`diff -u`' and include `ChangeLog` entries. Please follow the existing coding style (even if you do not like it).

Please send your bug reports, your suggestions, your patches or your comments to:

> '`bug-mpria@gnu.org`'.

If you think something in this manual is unclear, or downright incorrect, or if the language needs to be improved, please send a note to the same address.

4 MPRIA Basics

As MPRIA is built upon GMP, it is very advisable to read the GMP Manual first.

4.1 Headers and Libraries

All declarations needed to use MPRIA are collected in the C header file `mpria.h`; it is designed to work with both C and C++ compilers. You should include this file in any program using MPRIA:

```
#include <mpria.h>
```

All programs using MPRIA must link against both `libmpria` and `libgmp` libraries. On typical Unix-like systems this can be done with '`-lmpria -lgmp`' (in that order), for example:

```
gcc -o myprogram myprogram.c -lmpria -lgmp
```

GMP and MPRIA libraries are both built using Libtool, thus an application can use that to link if desired (see Section "Integrating libtool" in *GNU Libtool*).

If GMP or MPRIA have been installed to non-standard locations then it may be necessary to use '`-I`' and '`-L`' compiler options to point to the right directories, and some sort of run-time path for shared libraries.

4.2 Nomenclature and Types

A *rational interval* is a closed connected set of rational numbers, it is represented in MPRIA by its endpoints which are GMP rational numbers. The C data type for these objects is `mpri_t`.

MPRIA functions operate on valid rational intervals, while their behaviour remains undefined with non-valid rational intervals; a valid rational interval is defined as follows[1]:

- A *valid rational interval* can have finite or infinite endpoints, but its left endpoint is not larger than its right endpoint and cannot be *+infinity* $(+1/0)$ while the right endpoint cannot be *-infinity* $(-1/0)$. Whenever the left and right endpoints are equal to a same rational q, the valid rational interval reduces then to the singleton interval $[q, q]$ which represents exactly the rational q; conversely, any rational q is perfectly represented by the singleton interval $[q, q]$.

MPRIA functions may return intervals that are not valid as input value; their semantic is defined as follows[2]:

- Whenever the left endpoint or the right endpoint is *NaN* $(0/0)$, it indicates that an *invalid operation* has been performed and that the resulting rational interval has no mathematical meaning.
- Whenever the left endpoint is strictly greater than the right endpoint, it means that the resulting rational interval is the *empty interval*.

Some functions on rational intervals return a rational number. Among such functions, there are `mpri_get_left` and `mpri_get_right` that respectively return the left and right endpoints of a rational interval, and there is `mpri_diam_abs` that computes the width of a rational interval.

[1] The definition of a valid rational interval might be refined in future releases of MPRIA.

[2] The meaning of an invalid operation, the representation of the empty interval and their handling may evolve in future releases of MPRIA, according to the standardisation of interval arithmetic in *IEEE-1788* (see Appendix A [References], page 22).

Rational numbers (or *rationals* for short) and rational arithmetic functions are brought as is from the GMP library. The C data type for rationals is `mpq_t`, while their related functions start with the prefix `mpq_` (see Section "Rational Number Functions" in *The GNU MP Manual*).

For rational intervals, because their endpoints are numbers exactly representable that are meant to enclose a result not exactly representable, the notion of precision is essentially related to their width which is meant to be arbitrarily small. The *precision* of a rational interval designs the *integer binary logarithm* of the reciprocal of its width; as such, it expresses in bits. The corresponding C data type is `mpri_prec_t`.

When a MPRIA function implements some sort of convergent algorithm to return rational intervals, besides passing a precision parameter in bits to terminate the computation, a *surrounding mode* parameter specifies whether to place the *best convert* either at the left endpoint, at the right endpoint or arbitrarily. The C data type for these modes is `mpri_srnd_t`. Typically it concerns implementations based on the Euclidean algorithm (which are omnipresent).

Some MPRIA functions that involve heavy computations admit as last parameter an *assignment mode* which specifies whether to assign either only the left endpoint, only the right endpoint, or the two endpoints. The C data type for these modes is `mpri_asgmt_t`. Those functions are considered as low-level and are both appended with the capitalised suffix `_ASGMT` and wrapped by a macro that assigns the two endpoints.

4.3 Function Classes

There are four classes of functions in the MPRIA library:

1. Functions for intervals computation based on rational numbers: their names begin with `mpri_` and their associated type is `mpri_t`. This class gathers the standard computing assignment methods and concomitants, computing subroutines for rational interval approximations of quadratic irrational numbers, the four basic binary arithmetic operations and the classic unary operators built around them, and computing subroutines for rational interval approximations of elementary analytic mathematical functions. (See Chapter 5 [Rational Interval Functions], page 12.)
2. Low-level functions for rational interval approximations of analytic mathematical functions: their names are both prepended by `mpri_` and appended by `_ASGMT`, their associated type is `mpri_t` while their last parameter is an assignment mode of type `mpri_asgmt_t`. These low-level functions are not meant to be called directly but rather efficiently enwrapped within inline or macro functions. (See Chapter 6 [Low-Level Rational Interval Functions], page 17.)
3. Fast and convenient low-level functions that operate on signed integers and rational numbers: their names begin with `mpria_mpq_` and `mpria_mpz_`, respectively; their associated type are `mpz_t` and `mpq_t`, respectively. Implemented with great efficiency and handiness in mind, these functions are mainly inline and macro functions that are intensively used by the functions in the precedent categories; you are highly encouraged to employ them directly within time-critical or intricate subroutines. They intently complete rather than substitute their already furnished alikes in the GNU MP library, the prefix `mpria_` preventing from possible naming conflicts. (See Chapter 7 [Extra Number Functions], page 18.)
4. Miscellaneous functions. As memory management is inherited from the GNU MP library by design, this miscellanea essentially concerns functions for handling up different versions of the library. Two kinds of version handling function are distinguished: the functions that treat the version data of the library against which the application is effectively compiled, as such they act at compile time; the functions that deal with the version data of the library against which the application is dynamically linked, therefore they rather serve at run time. The formers are C preprocessor macros with names beginning with `MPRIA_VERSION_`,

the latters are C plain functions with names beginning with `mpria_libversion_`. (See Chapter 8 [General Library Functions], page 20.)

4.4 Variable Conventions

MPRIA functions expect output arguments before input arguments. This general rule, which is inherited from the GNU MP library, is based on an analogy with the assignment operator.

As a matter of fact, the analogy has been pushed further by allowing to use the same variable for both input and output in the same expression; this extension of the general rule is also inherited from the GNU MP library. For example, the square function, `mpri_sqr`, can be used as follows:

```
mpri_sqr (x, x);
```

what computes the set of squares of every rational number belonging to `x` and puts the results back in `x`.

As for MP variables, MPRIA variables must be initialised once before any assignment and may be cleared out after use. A (MP or) MPRIA variable should be initialised only once, or at least be cleared out between each initialisation. After such a variable has been initialised, it can be assigned numerous times; it will have the same allocated space during all its lifetime.

For efficiency reasons, avoid excessive initialising and clearing out: as a rule of thumb, initialise near the beginning of an application and clear out near its ending; better still, implement workspaces or garbage collections to pass and reuse these variables all along the computing process.

4.5 Precision Handling and Surrounding Modes

The following six PRECision parameters are predefined with respect to the *IEEE-754* standard (see Appendix A [References], page 22), except notably for the *meaningless precision*:

- `MPRI_PREC_BITS_NIL`: meaningless precision,
- `MPRI_PREC_BITS_HALF`: half precision (binary16) or 11 bits,
- `MPRI_PREC_BITS_SINGLE`: single precision (binary32) or 24 bits,
- `MPRI_PREC_BITS_DOUBLE`: double precision (binary64) or 53 bits,
- `MPRI_PREC_BITS_QUADRUPLE`: quadruple precision (binary128) or 113 bits,
- `MPRI_PREC_BITS_OCTUPLE`: octuple precision or 237 bits.

The following three SuRrouNDing modes are supported:

- `MPRI_SRND_BCAL`: Best Convert At Left endpoint,
- `MPRI_SRND_BCAA`: Best Convert At Any endpoint,
- `MPRI_SRND_BCAR`: Best Convert At Right endpoint.

4.6 Assignment Modes

The following three ASsiGnMenT modes are supported:

- `MPRI_ASGMT_OL`: assign Only Left endpoint,
- `MPRI_ASGMT_LR`: assign Left and Right endpoints,
- `MPRI_ASGMT_OR`: assign Only Right endpoint.

4.7 Memory Management

Basically MPRIA mimics and relays to the GNU MP memory management, except notably for temporary use (see Section "Memory Management" in *The GNU MP Manual*).

The `mpq_t` type is for the implementation of the `mpri_t` type what the `mpz_t` type is for the implementation of the `mpq_t` type itself: `mpri_t` variables never reduce their allocated space, as `mpq_t` variables.

All memory is allocated, reallocated and freed by passing on to the GNU MP memory functions as grabbed from `mp_get_memory_functions` (see Section "Custom Allocation" in *The GNU MP Manual*).

While GMP uses *temporary memory on the stack* (via `alloca`), MPRIA creates, passes along and intensively reuses *workspaces* for internal computation; the various created workspaces are freed before exiting with the help of the standard C `atexit` function (see Section "Cleanups on Exit" in *The GNU C Library Reference Manual*), therefore no memory leaks should be reported by tools like `valgrind` (`http://valgrind.org/`).

> **Teething Note:** At the time of writing, this internal workspace machinery is robust but **global**, read **not yet thread safe**, and no high-level function is yet implemented to free the created workspaces, or part of them, from time to time.

4.8 Autoconf

For applications using `autoconf` and its friends, the macro `mpria_AM_PATH_MPRIA` available in the file `mpria.m4` can be employed to link with the MPRIA automatically from the `configure` script. As preliminary work, this macro checks whether MPRIA is properly installed and performs compatibility test against either a specified version of the library or a default workable version of a recent major release of the library. To use this macro simply add the following line to the `configure.ac` `autoconf` input file:

```
mpria_AM_PATH_MPRIA([MPRIA_VERSION],
                    [action-if-found],
                    [action-if-not-found])
```

where the arguments are optional. The first argument `MPRIA_VERSION` should be either the one digit version number MAJOR, the two digit dotted version number MAJOR.MINOR or the three digit dotted version number MAJOR.MINOR.MICRO of the required release of the GNU MPRIA library. While `action-if-found` might be worthily empty or `:`, a suitable choice for `action-if-not-found` is

```
AC_MSG_ERROR([no suitable GNU MPRIA library found])
```

Then the variables `MPRIA_CPPFLAGS`, `MPRIA_CFLAGS`, `MPRIA_LDFLAGS` and `MPRIA_LIBS` can be added to the `Makefile.am` `automake` input files to obtain the correct preprocessor, compiler and linker flags. For example:

```
libfoo_la_CPPFLAGS = $(MPRIA_CPPFLAGS) $(GMP_CPPFLAGS)
libfoo_la_CFLAGS = $(MPRIA_CFLAGS) $(GMP_CFLAGS)
libfoo_la_SOURCES = foo-dim.c foo-dam.c foo-dom.c
libfoo_la_LDFLAGS = $(MPRIA_LDFLAGS) $(GMP_LDFLAGS)
libfoo_la_LIBADD = $(MPRIA_LIBS) $(GMP_LIBS) $(LIBM)
```

Note that the macro `mpria_AM_PATH_MPRIA` requires the macro `mpria_AM_PATH_GMP` which is provided in the file `mpria_ax_prog_path_gmp_cc.m4`; as you have already guessed, the macro `mpria_AM_PATH_GMP` is for the GNU MP library what the macro `mpria_AM_PATH_MPRIA` is for the

GNU MPRIA library. So, in the configure.ac file, the macro mpria_AM_PATH_GMP must precede the macro mpria_AM_PATH_MPRIA. In the previous example, the variables GMP_CPPFLAGS, GMP_CFLAGS, GMP_LDFLAGS and GMP_LIBS are furnished by the macro mpria_AM_PATH_GMP; the variable LIBM being set up by the Libtool macro LT_LIB_M.

For building more closely to the GNU MP library built, further tweaks are required. The main difficulty is to grab and use at proper time the compiler information stored at GNU MP build-time in the two macros __GMP_CC and __GMP_CFLAGS, which are defined in the header file gmp.h. Ideally this information should be first obtained with the help of a C PreProcessor (CPP) in such a way that the C Compiler (CC) could be then set up accordingly. Unfortunately, at the time of writing, the only ready-to-use autoconf macro meant to set up the C preprocessor to be employed, that is to say AC_PROG_CPP, depends to do so on the autoconf macro AC_PROG_CC, which determines with no easy comeback the C compiler to be employed: in short, the difficulty is harder than expected. As a matter of fact, the file mpria_ax_prog_path_gmp_cc.m4 contains a bunch of macros that allows to overcome the issue in a transparent way for the final developer: the macro mpria_AC_PROG_GMP_CC have to be used instead of the macro AC_PROG_CC. Typically the configure.ac file may so contain something similar to the following scrap of code:

```
dnl Setup CC and CFLAGS wrt GMP:
mpria_AC_PROG_GMP_CC

dnl Checks for libraries:
dnl  the math library:
LT_LIB_M
dnl  the GMP libray:
mpria_AM_PATH_GMP([5.1.3])
dnl  the GNU MPRIA library:
mpria_AM_PATH_MPRIA([0.7.1])
```

Besides, the usage of mpria_AC_PROG_GMP_CC reinforces the checks done by mpria_AM_PATH_GMP. To allow code readability improvement, the two latter macros have been combined into the single macro mpria_AC_PROG_PATH_GMP_CC. The above scrap of code can thus be rewritten as follows:

```
dnl Setup CC and CFLAGS wrt GMP:
mpria_AC_PROG_PATH_GMP_CC([5.1.3])

dnl Checks for libraries:
dnl  the math library:
LT_LIB_M
dnl  the GNU MPRIA library:
mpria_AM_PATH_MPRIA([0.7.1])
```

Last but not least, non-standard installation locations of the MPRIA and GMP libraries are handled with respect to customary use; in particular, command line options are implemented in the configure script to specify these locations. The macro mpria_AM_PATH_MPRIA affords the following command line options which accept an absolute path as compulsory argument:

- --with-mpria-prefix=PREFIX assumes that MPRIA is installed in the PREFIX directory, the default assumption being /usr/local;

- --with-mpria-include=PATH specifies that PATH is the MPRIA include directory, the default being PREFIX/include;

- --with-mpria-lib=PATH specifies that PATH is the MPRIA lib directory, the default being PREFIX/lib.

The macros mpria_AC_PROG_GMP_CC, mpria_AM_PATH_GMP and mpria_AC_PROG_PATH_GMP_CC implement command line options that have exactly the same usage but for the GMP library instead: --with-gmp-prefix, --with-gmp-include and --with-gmp-lib, respectively. In addition, these macros declare the environment variable GMP_GPP as *precious*: this advanced feature enables to specify a Generic PreProcessor command for early processing of the header file gmp.h.

5 Rational Interval Functions

5.1 Initialisation Functions

An `mpri_t` object must be initialised before storing the first value in it: the function `mpri_init` is used for that purpose, the function `mpri_clear` clears it out.

`void mpri_init` (*mpri_t* `x`) [Inline Function]

Initialise *x* and set it to the singleton interval $[0/1, 0/1]$. Normally, a variable should be initialised once only or at least be cleared out (using `mpri_clear`) between consecutive initialisation.

`void mpri_clear` (*mpri_t* `x`) [Inline Function]

Free the space occupied by the endpoints of *x*. Make sure to call this function for all `mpri_t` variables when you are done with them.

5.2 Assignment Functions

These functions and macros assign new values to already initialised rational intervals.

`void mpri_set` (*mpri_t* `rop`, *const mpri_t* `op`) [Inline Function]

Assign *rop* from *op*.

`void MPRI_SET_ZERO` (*mpri_t* `op`) [Macro]
`void MPRI_SET_NAN` (*mpri_t* `op`) [Macro]

Set the value of *op* to the singleton intervals $[0/1, 0/1]$ (zero) and $[0/0, 0/0]$ (NaN), respectively.

`void MPRI_SET_Q` (*mpri_t* `rop`, *const mpq_t* `op`) [Macro]

Set the value of *rop* to the singleton interval $[op, op]$.

`void mpri_set_qi_z` (*mpri_t* `rop`, *const mpz_t* `op1`, *const mpz_t* `op2`, *const mpz_t* `op3`, *mpri_prec_t* `prec`, *mpri_srnd_t* `srnd`) [Macro]
`void mpri_set_qi_q` (*mpri_t* `rop`, *const mpq_t* `op1`, *const mpq_t* `op2`, *const mpq_t* `op3`, *mpri_prec_t* `prec`, *mpri_srnd_t* `srnd`) [Inline Function]

Set the value of *rop* to the best rational interval approximation of the quadratic irrational number $(op1 + \sqrt{op2})/op3$ with a **guaranteed** precision of at least *prec* bits and with respect to the surrounding *srnd*. The result remains undefined if the radicand *op2* is negative or if the divisor *op3* is zero. While the macro `mpri_set_qi_z` is its natural high-level wrapper, the inline function `mpri_set_qi_q` belongs to one of the efficient wrappers implemented around the low-level function `mpri_set_qi_z_ASGMT`.

`void mpri_set_q` (*mpri_t* `rop`, *const mpq_t* `op`, *mpri_prec_t* `prec`, *mpri_srnd_t* `srnd`) [Inline Function]
`void mpri_set_d` (*mpri_t* `rop`, *double* `op`, *mpri_prec_t* `prec`, *mpri_srnd_t* `srnd`) [Inline Function]

Set the value of *rop* to the best rational interval approximation of the number *op* (respectively, a rational number and a `double`) with a **guaranteed** precision of at least *prec* bits and with respect to the surrounding *srnd*. Both are inline wrappers efficiently built around the low-level function `mpri_set_qi_z_ASGMT`; a rational being a degenerate quadratic irrational, a `double` an approximative rational representation of a real number.

void **mpri_set_sqrt_q** (*mpri_t* **rop**, *const mpq_t* **op**, *mpri_prec_t* **prec**, *mpri_srnd_t* **srnd**) [Inline Function]

Set the value of *rop* to the best rational interval approximation of the square root of *op*, $\sqrt{op}$, with a **guaranteed** precision of at least *prec* bits and with respect to the surrounding *srnd*. The result is undefined if the radicand *op* is negative. It is an inline function that efficiently wraps around the low-level function `mpri_set_qi_z_ASGMT`.

void **mpri_set_rsqrt_q** (*mpri_t* **rop**, *const mpq_t* **op**, *mpri_prec_t* **prec**, *mpri_srnd_t* **srnd**) [Inline Function]

Set the value of *rop* to the best rational interval approximation of the reciprocal square root of *op*, literally $\sqrt{op}/op$, with a **guaranteed** precision of at least *prec* bits and with respect to the surrounding *srnd*. The result stays undefined if the operand *op* is either negative or zero. This inline function is an efficient wrapper built around the low-level function `mpri_set_qi_z_ASGMT`.

void **mpri_swap** (*mpri_t* **rop1**, *mpri_t* **rop2**) [Inline Function]

Swap the values *rop1* and *rop2* efficiently.

5.3 Interval Conversion Functions

void **mpri_get_q** (*mpq_t* **rop**, *const mpri_t* **op**) [Inline Function]

Convert *op* to a rational number, which is its centre.[1]

double **mpri_get_d** (*const mpri_t* **op**) [Function]

Convert *op* to a `double`, this conversion is the composition of `mpri_get_q` and `mpq_get_d`.

5.4 Interval Comparison Functions

int **mpri_equal** (*const mpri_t* **op1**, *const mpri_t* **op2**) [Inline Function]

Return either `1` (read *true*) if the rational intervals *op1* and *op2* are equal or `0` (read *false*) if they are non-equal.

int **mpri_is_zero** (*const mpri_t* **op**) [Inline Function]

Return `1` (read *true*) if the rational interval *op* is the singleton interval $[0/1, 0/1]$ (zero), `0` (read *false*) otherwise.

int **mpri_is_nonzero** (*const mpri_t* **op**) [Inline Function]

Return `1` (read *true*) if the rational interval *op* does not reduce to the singleton interval $[0/1, 0/1]$ (zero), `0` (read *false*) otherwise.

int **mpri_has_zero** (*const mpri_t* **op**) [Inline Function]

Return `1` (read *true*) if zero belongs to the rational interval *op*, `0` (read *false*) otherwise.

int **mpri_hasnot_zero** (*const mpri_t* **op**) [Inline Function]

Return either `-1` if the rational interval *op* is strictly negative, or `+1` if it is strictly positive, or `0` if it contains zero

[1] An other conversion choice might be made in future releases of MPRIA; to explicitly obtain the centre of a rational interval, use `mpri_mid` instead.

5.5 Interval Basic Functions

Some MPRIA functions on rational intervals return rational results, such as the diameter or the centre of a rational interval.

`void mpri_diam_abs` (*mpq_t* `rop`, *const mpri_t* `op`) [Inline Function]
Set the value of *rop* to the absolute diameter of the rational interval *op*, that is to say, to the difference between its right endpoint and its left one.

`void mpri_diam_rel` (*mpq_t* `rop`, *const mpri_t* `op`) [Function]
Set the value of *rop* to the relative diameter of the rational interval *op*, in other words, either to the difference between its right endpoint and its left one divided by the absolute value of its centre when it is not symmetric or to *NaN* ($[0/0, 0/0]$) when it is symmetric.

`void mpri_diam` (*mpq_t* `rop`, *const mpri_t* `op`) [Inline Function]
Set the value of *rop* to the relative diameter of the rational interval *op* if it does not contains zero and to its absolute diameter otherwise.

`void mpri_mig` (*mpq_t* `rop`, *const mpri_t* `op`) [Inline Function]
`void mpri_mag` (*mpq_t* `rop`, *const mpri_t* `op`) [Inline Function]
Set the value of *rop* to the mignitude and magnitude of the rational interval *op*, respectively, that is to say, to the smallest and largest absolute value of its elements, respectively.

`void mpri_mid` (*mpq_t* `rop`, *const mpri_t* `op`) [Inline Function]
Set the value of *rop* to the value of the middle of the rational interval *op*, namely, to the half sum of its endpoints.

`mpq_t mpri_lepref` (*const mpri_t* `op`) [Macro]
`mpq_t mpri_repref` (*const mpri_t* `op`) [Macro]
Return a reference to the left and right endpoint of the rational interval *op*, respectively.

`void mpri_get_left` (*mpq_t* `rop`, *const mpri_t* `op`) [Inline Function]
`void mpri_get_right` (*mpq_t* `rop`, *const mpri_t* `op`) [Inline Function]
Set the value of *rop* to the left and right endpoint of the rational interval *op*, respectively. These functions are equivalent to calling `mpq_set` with an appropriate `mpri_lepref` or `mpri_repref`. Direct use of `mpri_lepref` or `mpri_repref` is recommended instead of these functions.

5.6 Interval Arithmetic Functions

`void mpri_add` (*mpri_t* `rop`, *const mpri_t* `op1`, *const mpri_t* `op2`) [Inline Function]
`void mpri_add_q` (*mpri_t* `rop`, *const mpri_t* `op1`, *const mpq_t* `op2`) [Inline Function]
Set *rop* to $op1 + op2$.

`void mpri_sub` (*mpri_t* `rop`, *const mpri_t* `op1`, *const mpri_t* `op2`) [Inline Function]
`void mpri_sub_q` (*mpri_t* `rop`, *const mpri_t* `op1`, *const mpq_t* `op2`) [Inline Function]
`void mpri_q_sub` (*mpri_t* `rop`, *const mpq_t* `op1`, *const mpri_t* `op2`) [Inline Function]
Set *rop* to $op1 - op2$.

`void mpri_mul` (*mpri_t* `rop`, *const mpri_t* `op1`, *const mpri_t* `op2`) [Function]
`void mpri_mul_q` (*mpri_t* `rop`, *const mpri_t* `op1`, *const mpq_t* `op2`) [Inline Function]
Set *rop* to $op1 \times op2$. Multiplication by zero, passed as singleton interval $[0/1, 0/1]$ or literally, gives the singleton interval $[0/1, 0/1]$.

`void mpri_div` (*mpri_t* **rop**, *const mpri_t* **op1**, *const mpri_t* **op2**) [Function]
`void mpri_div_q` (*mpri_t* **rop**, *const mpri_t* **op1**, *const mpq_t* **op2**) [Inline Function]
`void mpri_q_div` (*mpri_t* **rop**, *const mpq_t* **op1**, *const mpri_t* **op2**) [Inline Function]

Set *rop* to *op1*/*op2*. When the dividend *op1* reduces to the singleton interval $[0/1, 0/1]$, viz. zero, the division returns the singleton interval $[0/1, 0/1]$ as result; when the divisor *op2* contains zero, the division returns $[0/0, 0/0]$, namely *NaN*.

`void mpri_neg` (*mpri_t* **rop**, *const mpri_t* **op**) [Inline Function]

Set *rop* to $-op$.

`void mpri_abs` (*mpri_t* **rop**, *const mpri_t* **op**) [Inline Function]

Set *rop* to $|op|$, the absolute value of *op*.

`void mpri_inv` (*mpri_t* **rop**, *const mpri_t* **op**) [Inline Function]

Set *rop* to $1/op$ when the rational interval *op* does not contains zero, to $[0/0, 0/0]$ (*NaN*) otherwise.

`void mpri_sqr` (*mpri_t* **rop**, *const mpri_t* **op**) [Inline Function]

Set *rop* to op^2.

`void mpri_sqrt` (*mpri_t* **rop**, *const mpri_t* **op**, *mpri_prec_t* **prec**) [Inline Function]

Set *rop* to the best rational interval approximation of the square root of *op*, $\sqrt{op}$, with a **guaranteed** precision of at least *prec* bits. If the rational interval radicand *op* is not positive, the return interval is $[0/0, 0/0]$, namely *NaN*. This inline function implements an efficient wrapper around the low-level function `mpri_set_qi_z_ASGMT`.

`void mpri_rsqrt` (*mpri_t* **rop**, *const mpri_t* **op**, *mpri_prec_t* **prec**) [Inline Function]

Set *rop* to the best rational interval approximation of the reciprocal square root of *op*, literally $\sqrt{op}/op$, with a **guaranteed** precision of at least *prec* bits. If the rational interval operand *op* is not strictly positive, the return interval is $[0/0, 0/0]$, to wit *NaN*. This inline function efficiently implements a wrapper around the low-level function `mpri_set_qi_z_ASGMT`.

`void mpri_mul_2exp` (*mpri_t* **rop**, *const mpri_t* **op**, *unsigned long int* **exponent**) [Inline Function]

Set *rop* to $op \times 2^{exponent}$.

`void mpri_div_2exp` (*mpri_t* **rop**, *const mpri_t* **op**, *unsigned long int* **exponent**) [Inline Function]

Set *rop* to $op/2^{exponent}$.

5.7 Interval Approximation of Elementary Functions

> **Teething Note:** At the time of writing, this part of the library is clearly at a very early stage as it basically contains only *one* function: more functions may be furnished in the coming minor releases, the all set of elementary functions in the next major release.

`void mpri_atan` (*mpri_t* **rop**, *const mpri_t* **op**, *mpri_prec_t* **prec**) [Inline Function]

Set *rop* to the best rational interval approximation of the arc-tangent of *op*, $\arctan(op)$, with a **guaranteed** precision of at least *prec* bits. This inline function straightforwardly wraps the function `mpri_2exp_atan`.

`void mpri_2exp_atan` (*mpri_t* **`rop`**, *unsigned long int* **`exponent`**, *const mpri_t* **`op`**, *mpri_prec_t* **`prec`**) [Function]

Set *rop* to the best rational interval approximation of 2 raised to *exponent* times the arc-tangent of *op*, $2^{exponent} \times \arctan(op)$, with a **guaranteed** precision of at least *prec* bits.

6 Low-Level Rational Interval Functions

6.1 Low-Level Interval Elementary Functions

void mpri_set_qi_z_ASGMT (*mpri_t* **rop**, *const mpz_t* **op1**, *const mpz_t* **op2**, *const mpz_t* **op3**, *mpri_prec_t* **prec**, *mpri_srnd_t* **srnd**, *mpri_asgmt_t* **asgmt**) [Function]

Set the value of *rop* to the best rational interval approximation of the quadratic irrational number $(op1+\sqrt{op2})/op3$ with a **guaranteed** precision of at least *prec* bits and with respect to both the surrounding *srnd* and the assignment mode *asgmt*. The result remains undefined if the radicand *op2* is negative or if the divisor *op3* is zero.

6.2 Hard-Coded Numbers

The following collections of hard-coded numbers are mainly meant to serve the previous low-level functions within enwrapping inline functions or plain functions. For illustrations on how to wrap with them, peruse the header file `mpria.h`.

const mpz_t __mpria_z_zero [Constant]
const mpz_t __mpria_z_pos_one [Constant]
const mpz_t __mpria_z_neg_one [Constant]
const mpz_t __mpria_z_pos_two [Constant]
const mpz_t __mpria_z_neg_two [Constant]

Collection of `mpz_t` signed integers with self-explanatory names.

const mpq_t __mpria_q_zero [Constant]
const mpq_t __mpria_q_pos_one [Constant]
const mpq_t __mpria_q_neg_one [Constant]
const mpq_t __mpria_q_pos_two [Constant]
const mpq_t __mpria_q_neg_two [Constant]

Collection of `mpq_t` rational numbers with self-explanatory names.

const mpri_t __mpria_ri_zero [Constant]
const mpri_t __mpria_ri_pos_one [Constant]
const mpri_t __mpria_ri_neg_one [Constant]

Collection of `mpri_t` rational singleton intervals with self-explanatory names.

7 Extra Number Functions

7.1 Extra Rational Number Functions

`MPRIA_MPQ_SET_ZERO` (*Q*) [Macro]
`MPRIA_MPQ_SET_POS_ONE` (*Q*) [Macro]
`MPRIA_MPQ_SET_NEG_ONE` (*Q*) [Macro]
`MPRIA_MPQ_SET_NAN` (*Q*) [Macro]
`MPRIA_MPQ_SET_POS_INF` (*Q*) [Macro]
`MPRIA_MPQ_SET_NEG_INF` (*Q*) [Macro]

Set the value of the rational number *Q* to `0`, `+1`, `-1`, `0/0` (*NaN*), `+1/0` (+*infinity*) and `-1/0` (-*infinity*), respectively. These utility functions are implemented as plain macros (with self-explanatory names).

`MPRIA_MPQ_IS_ZERO` (*Q*) [Macro]
`MPRIA_MPQ_IS_NONZERO` (*Q*) [Macro]
`MPRIA_MPQ_IS_POSITIVE` (*Q*) [Macro]
`MPRIA_MPQ_IS_NEGATIVE` (*Q*) [Macro]
`MPRIA_MPQ_IS_STRICTLY_POSITIVE` (*Q*) [Macro]
`MPRIA_MPQ_IS_STRICTLY_NEGATIVE` (*Q*) [Macro]

Return `1` (read *true*) if the rational number *Q* is either zero, nonzero, positive, negative, strictly positive or strictly negative, respectively, `0` (read *false*) otherwise. These test functions are plain macro functions (with self-explanatory names).

int mpria_mpq_is_nan (*const mpq_t op*) [Inline Function]

Return `1` (read *true*) if the rational number *op* is *Not-a-Number*, `0` (read *false)* otherwise.

NaN, the acronym for Not-a-Number, has the representation 0/0.[1]

int mpria_mpq_is_infinite (*const mpq_t op*) [Inline Function]

Return `+1` if the rational number *op* is *positive infinity*, `-1` if it is *negative infinity*, `0` otherwise.

Positive and negative infinities have the representation $+1/0$ and $-1/0$, respectively;[2] they are commonly written +*infinity* and -*infinity*, respectively.

int mpria_mpq_is_finite (*const mpq_t op*) [Inline Function]

Return `1` (read *true*) if the rational number *op* is finite, `0` (read *false*) if it is either infinite or Not-a-Number.

int mpria_mpq_sgn (*const mpq_t op*) [Inline Function]

Return `+1` if the rational *op* is strictly positive, `0` if it is zero, or `-1` if it is strictly negative. Its behaviour stays undefined if its argument is *NaN* (0/0).
While its counterpart `mpq_sgn` is implemented as a macro, this function is implemented as an inline function: it evaluates its argument only once.

int mpria_mpq_cmpabs (*const mpq_t op1, const mpq_t op2*) [Function]

Compare the absolute values of the rational numbers *op1* and *op2*. Return either a positive value if |*op1*| is strictly greater than |*op2*|, zero if |*op1*| is equal to |*op2*|, or a negative value if |*op1*| is strictly smaller than |*op2*|. Its behaviour remains undefined if at least one of its arguments is either -*infinity* ($-1/0$), +*infinity* ($+1/0$), or *NaN* (0/0).

---

[1] At the time of writing, GMP does not support *NaN* for `mpq_t` numbers.

[2] At the time of writing, GMP does not support infinities for `mpq_t` numbers.

`void mpria_mpq_min3` (*mpq_t* `rop`, *const mpq_t* `op1`, *const mpq_t* `op2`, *const mpq_t* `op3`) [Inline Function]

Set the value of *rop* to the minimum of the triplet $\{op1, op2, op3\}$. Its behaviour is undefined if the triplet contains *-infinity* ($-1/0$), *+infinity* ($+1/0$), or *NaN* ($0/0$).

7.2 Extra Signed Integer Functions

`MPRIA_MPZ_SET_ZERO` (*Z*) [Macro]
`MPRIA_MPZ_SET_POS_ONE` (*Z*) [Macro]
`MPRIA_MPZ_SET_NEG_ONE` (*Z*) [Macro]

Set the value of the signed integer *Z* to `0`, `+1` and `-1`, respectively. These utility functions are implemented as plain macros (with self-explanatory names).

`MPRIA_MPZ_IS_ZERO` (*Z*) [Macro]
`MPRIA_MPZ_IS_NONZERO` (*Z*) [Macro]
`MPRIA_MPZ_IS_POSITIVE` (*Z*) [Macro]
`MPRIA_MPZ_IS_NEGATIVE` (*Z*) [Macro]
`MPRIA_MPZ_IS_STRICTLY_POSITIVE` (*Z*) [Macro]
`MPRIA_MPZ_IS_STRICTLY_NEGATIVE` (*Z*) [Macro]

Return `1` (read *true*) if the signed integer *Z* is either zero, nonzero, positive, negative, strictly positive or strictly negative, respectively, and `0` (read *false*) otherwise. These test functions are plain macro functions (with self-explanatory names).

`int mpria_mpz_sgn` (*const mpz_t* `op`) [Inline Function]

Return `+1` if the signed integer *op* is strictly positive, `0` if it is zero, or `-1` if it is strictly negative.

While its counterpart `mpz_sgn` is implemented as a macro, this function is implemented as an inline function: it evaluates its argument only once.

`void mpria_mpz_minabs3` (*mpz_t* `rop`, *const mpz_t* `op1`, *const mpz_t* `op2`, *const mpz_t* `op3`) [Inline Function]

Set the value of *rop* to the minimum of the triplet $\{|op1|, |op2|, |op3|\}$.

8 General Library Functions

8.1 Library Version Handling

Different releases of the GNU MPRIA library are distinguished by an **authoritative** version triplet of nonnegative integer constants defined as macro constants. Utilities are implemented to efficiently check against, to numerically pack or to stringify this triplet; packed variants of the triplet are also defined as macro constants.

`MPRIA_VERSION_MAJOR` [Macro]
`MPRIA_VERSION_MINOR` [Macro]
`MPRIA_VERSION_MICRO` [Macro]

The **authoritative** version triplet, respectively, as nonnegative integer constants: the major version number, the minor version number (or revision number), the micro version number (or major patch level).

`void mpria_libversion_get_numbers` (*int * major*, *int * minor*, *int * micro*) [Function]

Retrieve the *major*, *minor* and *micro* version numbers of the MPRIA library against which the application is currently linked. The `NULL` pointer is accepted as argument.

`int mpria_libversion_check_numbers` (*int major*, *int minor*, *int micro*) [Function]

Check the compatibility of the arbitrary *major*, *minor* and *micro* version numbers with their counterpart from the MPRIA library against which the application is currently linked. The returned response is as follows:

- 0 if the two version triplets are not compatible (incompatibility);
- 1 if they are compatible and exactly the same (strict or strong compatibility);
- 2 if they are compatible but not exactly the same (weak compatibility).

This function performs no action apart from checking and responding, in particular it does not cause the application to `abort` or to show up any kind of messages (it may be enwrapped within a `if else` statement to do so).

`int mpria_libversion_check` (*void*) [Macro]

Check the compatibility of the version triplet of the MPRIA library with which an application was compiled with the version triplet of the MPRIA library against which the application is currently linked. This is a convenient wrapping macro that passes the authoritative macro version numbers to the function `mpria_libversion_check_numbers`, as such it acts similarly. The most common cause for an incompatibility or a weak compatibility is that an application was compiled against one version of the MPRIA library while it is dynamically linked against a different one, what might be due to a misconfiguration, a downgrading or an upgrading. A typical usage may look like:

```
/* Check version of libmpria */
if (!(mpria_libversion_check ()))
{
  fprintf (stderr,"version miss-compatibility\n");
  fflush (stderr);
  abort ();
}
```

`MPRIA_VERSION_EXTRA` [Macro]

The extra version string suffix, only meant for development purposes. For production releases, *alpha* and *stable* ones, it must be reset to the empty string `""`.

`MPRIA_VERSION_NUMBER_PACK` (*Major*, *Minor*, *Micro*) [Macro]

`MPRIA_VERSION_STRING_PACK` (*Major*, *Minor*, *Micro*, *StrExtra*) [Macro]

Compact, respectively stringify, the arbitrary version triplet [*Major*, *Minor*, *Micro*] into a single number, resp. into a null-terminated string to which is appended the arbitrary extra version string suffix *StrExtra*.

`MPRIA_VERSION_NUMBER` [Macro]

`MPRIA_VERSION_STRING` [Macro]

The *non-authoritative* version number, respectively string, obtained by passing the **authoritative** version triplet to `MPRIA_VERSION_NUMBER_PACK`, resp. to `MPRIA_VERSION_STRING_PACK` with `MPRIA_VERSION_EXTRA` as fourth argument.

`int mpria_libversion_get_number` (*void*) [Function]

`const char * mpria_libversion_get_string` (*void*) [Function]

Retrieve the *non-authoritative* version number and string, respectively, of the MPRIA library against which the application is currently linked.

`const char * mpria_libversion` [Macro]

`const char * mpria_version` [Macro]

The version string of the MPRIA library against which the application is currently linked. While `mpria_libversion` is a convenient macro that wraps `mpria_libversion_get_string`, `mpria_version` is defined as synonymous of `mpria_libversion` with respect to the GNU MP naming scheme.

8.2 Miscellaneous Utilities

`MPRIA_STRINGIFY` (*Token*) [Macro]

Stringify *Token*.

Appendix A References

Teething Note: This is clearly a **non-exhaustive** list (in progress) of references.

- IEEE-1788, Interval Standard Working Group:
  http://grouper.ieee.org/groups/1788/.
- IEEE-754, Standard for Binary Floating-Point Arithmetic:
  http://grouper.ieee.org/groups/754/.

Appendix B GNU General Public License

Version 3, 29 June 2007

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program—to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

   "This License" refers to version 3 of the GNU General Public License.

   "Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

   "The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

   To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

   A "covered work" means either the unmodified Program or a work based on the Program.

   To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

   To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

   An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

   The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

   A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

   The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

   The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

   All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

   You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

   Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

   No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

   When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

   You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

   You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

   You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

   a. The work must carry prominent notices stating that you modified it, and giving a relevant date.

   b. The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c. You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d. If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a. Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b. Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c. Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d. Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e. Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

   "Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

   When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

   Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

   a. Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b. Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
c. Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
d. Limiting the use for publicity purposes of names of licensors or authors of the material; or
e. Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
f. Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

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, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

    Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

    An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

    You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

    A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

    A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

    Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

    In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

    If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

    If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

    A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

    If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

    Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

    The Free Software Foundation may publish revised and/or new versions of the GNU General Public 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.

    Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

    If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

    Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

    THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF

MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

    IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

    If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
one line to give the program's name and a brief idea of what it does.
Copyright (C) year name of author

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or (at
your option) any later version.

This program is distributed in the hope that it will be useful, but
WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see http://www.gnu.org/licenses/.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
program Copyright (C) year name of author
This program comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type 'show c' for details.
```

The hypothetical commands '`show w`' and '`show c`' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see `http://www.gnu.org/licenses/`.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read `http://www.gnu.org/philosophy/why-not-lgpl.html`.

Appendix C GNU Free Documentation License

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.

Appendix D Indices

D.1 Concept Index

D.2 Type Index

D.3 Variable Index

D.4 Function Index