

Table of Contents

<u>Welcome to MetaPRL</u>	1
<u>System goals</u>	1
<u>MetaPRL Documentation</u>	2
<u>MetaPRL links</u>	4
<u>Installing MetaPRL</u>	5
<u>Downloading MetaPRL</u>	5
<u>Subversion Access to MetaPRL Sources</u>	5
<u>Installation and configuration</u>	6
<u>OCaml</u>	6
<u>OMake</u>	6
<u>MetaPRL</u>	6
<u>MetaPRL history and people</u>	8
<u>History</u>	8
<u>People</u>	8
<u>MetaPRL License</u>	10
<u>GNU GENERAL PUBLIC LICENSE</u>	10
<u>Preamble</u>	10
<u>TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION</u>	11
<u>How to Apply These Terms to Your New Programs</u>	14

Welcome to MetaPRL



MetaPRL is two things: it is a *logical framework* where multiple logics can be defined and related, and it is a *system implementation* with support for interactive proof and automated reasoning. A primary feature of MetaPRL is a semantic connection to programming languages, that allows the system to be used as a *logical programming environment*, where programs are constructed as a mixture of specifications, implementations, and verifications.

System goals

The MetaPRL system was implemented with the purpose of supporting *relations* between logics. There is a huge investment in formal work in systems like PVS, HOL, Coq, ELF, Nuprl, and others. These systems use different logics and different methodologies, but they have common goals and their results share fundamental mathematical underpinnings. Mathematical developments are expensive; our first goal is to expose the logical foundations that the systems share, to allow the *results* to be shared between systems. MetaPRL supports sharing with three features:

- Programs and logics are developed as modules that define *computational*, *heuristic*, and *mathematical* properties.
- Modules are constructed *incrementally* by adding formal properties to existing modules (the empty module is the root of every logic). Just as object-oriented programming allows code re-use, incremental module construction allows *logical* re-use: logics can share a common core with properties that are *inherited* as the logics are extended.
- Modules are *first-class*: relations between logics are either constructed by inheritance, or explicitly constructed as functions between modules.

This first goal give formal underpinnings to logical sharing; our next goal is to provide practical impact. Work is underway to relate the PVS, HOL, Isabelle, and Nuprl mathematical foundations. Precursors to these developments are available in the MetaPRL distribution, including logics for:

- computational type theory (ITT), based on the Martin-Lof style Nuprl type theory,
- constructive set theory, based on Aczel's axiomatization,
- the LF logical framework,
- first order logic.

These logics provide the principles of formal logical relations. Aczel's set theory is modeled in the type theory (as is first-order logic). The LF work is less complete.

MetaPRL

Our work focuses specifically on *programming*: even though large-scale programming is unreliable, the advantage of high-speed automation has pushed computation into our safety-critical infrastructure, including the telephone system, the power grid, the financial sector, and the transportation industry. The reliability of critical systems must be ensured—while retaining productivity. MetaPRL addresses the problem in two ways:

- The system provides a semantic framework to allow program *development* in an environment of mixed specification and implementation. Logical foundations are used both to guide the development and to ensure the reliability of developing software. The distribution contains a semantic account of core OCaml that allows ML programs to be intermixed with specifications in type theory.
- The majority of existing code is poorly documented and poorly structured. While this is the most speculative of our projects, MetaPRL supports the *hardening* of existing code by developing program-specific type systems that verify program abstractions.

Finally, our goal is *speed*. Higher-order formal systems require a mixture of interactive and automated proving; the load on the programmer is proportional to how efficient the system is in deriving formal properties. MetaPRL focuses on this problem at its foundation with these features:

- Modularity is used to provide domain-specific heuristics and algorithms, and to restrict the logical search complexity.
- The implementation language, OCaml, provides performance close to C, but with a formal semantics, type safety, modularity, and extensive checking to assist code modification and development.
- Abstract data structures provide optimized domain-specific logical operations.
- MetaPRL is a fully-distributed, fault-tolerant theorem prover. The distribution over Ensemble scales to over 100 machines over a wide-area network (depending on the parallel properties of the domain problem). Faults are tolerated transparently: processors may join and leave a process group arbitrarily—problem outcomes are not affected by process failures. Furthermore, the distribution is transparent: programmers use the standard search language.

MetaPRL Documentation

Here is the listing of some information provided in MetaPRL documentation. If you want to print MetaPRL documentation or read in Postscript or PDF format, you can either download it from <http://files.metaprl.org/doc/> (updated nightly) or compile it yourself. To compile it yourself, you'll need to [download and install](#) MetaPRL sources and the `htmldoc` utility and then run `omake docs` from the MetaPRL top-level directory.

People

gives a list of people known to be working on the project.

Links

provides links to the other related systems and software.

Downloading

describes how to download, configure, and install MetaPRL.

Logical framework

presents the logical properties of the system.

System description

gives an overview of MetaPRL's architecture and features.

User guide

provides instructions on using MetaPRL.

Tutorial

will help you to get started using the system.

Logical Theories

MetaPRL

is a daily updated PDF file that gives a listing of most of the logical theories implemented in MetaPRL together with some documentation.

Developer Guide

should be your first stop when you decide to add or update some MetaPRL code.

MetaPRL-Announce

mailing list is a low-volume list for announcing major MetaPRL developments.

MetaPRL-CVS

mailing list receives all the MetaPRL Commit messages and is ideal for tracking day-to-day development of MetaPRL.

Commit Logs

show MetaPRL revision history.

Bugzilla

MetaPRL bug reports repository — use it to file new bug reports and to browse existing ones.

This work is supported in part by the DoD Multidisciplinary University Research Initiative (MURI) program administered by the Office of Naval Research (ONR) under Grant N00014-01-1-0765, the Defense Advanced Research Projects Agency (DARPA), the United States Air Force, and the Lee Center.

MetaPRL links

MetaPRL is a part of the [Cornell Prl Automated Reasoning Project](#).

MetaPRL is implemented in the [OCaml language](#). OCaml has proved to an efficient and reliable ML implementation, thanks to Xavier Leroy and the [Project Cristal](#) group at [INRIA](#).

The MetaPRL language is an extension of the OCaml programming language. The syntax extensions are implemented with [CamlP4](#), a **pretty-printer pre-processor** for OCaml, developed by Daniel de Rauglaudre.

MetaPRL is listed in the [Science: Math: Logic and Foundations: Computational Logic: Logical Frameworks](#) section of the [Open Directory](#)

Installing MetaPRL

Warning: MetaPRL is currently Beta and could still have some nasty bugs.

Downloading MetaPRL

In order to compile and run MetaPRL you are going to need the following packages:

1. Objective Caml: our version contains a small patch to CamlP4 to compile the CamlP4 distribution in native code.
2. The OMake build system.
3. OpenSSL libraries and header files are optional, but highly recommended.
4. MetaPRL sources.

OCaml is currently available as source, Fedora Core 2, 3, 4 and 5, Mandrake 10.1, and Red Hat Enterprise Linux 3 and 4 RPM files for Linux x86. You can get the listing of available RPMs and download them from <http://rpm.nogin.org/ocaml.html> or a plain listing directly from <http://rpmbin.nogin.org/MetaPRL/>. If you are going to build OCaml yourself, get OCaml sources from <http://caml.inria.fr/ocaml/release.en.html> and our patches at <http://svn.metaprl.org/viewcvs/mojave/metaprl/patches/> (see the [README file](#) for the list of patches you need to apply).

OMake source and binaries for a number of Linux distributions OS X and Windows can be downloaded from <http://omake.metaprl.org/download.html>.

OCaml and OMake may also be obtained via the [GODI Distribution Framework](#).

If you are using a Unix-like operation system, you probably already have a copy of OpenSSL around (note that you need the development files as well, which are often distributed in a separate `openssl-devel` or `libopenssl-devel` package). OpenSSL for Windows is available at <http://www.slproweb.com/products/Win32OpenSSL.html>.

You should be able to download MetaPRL sources tarball from <http://files.metaprl.org/>, but if you plan to participate in MetaPRL development, you should consider using [the Subversion access](#) to MetaPRL sources.

Subversion Access to MetaPRL Sources

You can [browse the MetaPRL sources repository](#).

If you plan any development in MetaPRL or if you want to download the latest version of MetaPRL, you may consider using the public Subversion repository. This will allow you to have access to the latest changes in the distribution. The normal service is read-only, but if you would like to become a contributor with write access, read the [read-write Subversion instructions](#).

The Subversion server is on the machine `svn.metaprl.org`. You can checkout the distribution by running:

```
% svn co svn://svn.metaprl.org/metaprl
```

From the same server you can also checkout the OMake distribution by running:

```
% svn co svn://svn.metaprl.org/omake-branches/0.9.8.x omake
```

MetaPRL

If for some reason (firewall, *etc*) you are having trouble using the `svn://` access method, you can use `http://` instead:

```
% svn co http://svn.metaprl.org/svnroot/mojave/metaprl
% svn co http://svn.metaprl.org/svnroot/mojave/omake-branches/0.9.8.x omake
```

You can find more information on Subversion from the [Subversion Home Page](#), [Subversion FAQ](#), and from [The Subversion Book](#).

Installation and configuration

OCaml

If you are not using [the provided RPMs](#), then in order to install OCaml and Camlp4 download and unpack the distributions, then apply the [patches](#) (see the [README file](#) for the list of patches you need to apply). After that, proceed as described in the `INSTALL` files in the distributions. If your OS supports pthreads, run OCaml configure script with `-with-pthread` option in order to get pthreads support in OCaml. After you have compiled and installed OCaml and Camlp4, go to the OCaml source tree and copy the following files:

- parsing/location.cmi
- parsing/location.mli
- parsing/longident.cmi
- parsing/longident.mli
- parsing/parsetree.cmi
- parsing/parsetree.mli
- typing/typecore.cmi
- typing/typecore.mli

into the OCaml library directory.

OMake

To build OMake from sources change to the OMake source directory and run `./configure` followed by `make` and `make install`

MetaPRL

To install MetaPRL, unpack the distribution into some directory `<metaprl>` and build the distribution with these steps:

```
% cd <metaprl>
% omake
```

After you run `omake` for the first time, it will create the `mk/config` file (as well as an empty `mk/config.local` file) that you will need to customize, before the `omake` process can continue. The `mk/config` file is self-documenting.

You may need to set variables `CAMLLIB` and `CAML4LIB` to point to the directories where you have installed OCaml and Camlp4 files. You can either set them in the `mk/config.local` file, for example:

```
CAMLLIB=/usr/lib/ocaml
```

MetaPRL

CAML4LIB=/usr/lib/ocaml/camlp4

or in the environment. If a variable is set in both, `mk/config.local` takes precedence over the environment. The defaults are `/usr/lib/ocaml` and `/usr/lib/ocaml/camlp4` respectively.

Additional configuration options might be necessary on Windows and Mac OS X. See the corresponding README files in the top-level MetaPRL directory for more information.

MetaPRL can be run with the

```
<metaprl>/editor/ml/mp
```

script. By default, this will attempt to start your browser or prompt you to connect using your browser. MetaPRL looks better if you use Mozilla or Firefox, but other browsers (including Konqueror and IE) are likely to work as well. Internet Explorer is known to work well when "Arial Unicode MS" font is installed (`arialuni.ttf` is available on Microsoft Office CD inside `office1.cab`; note that it is not always installed automatically with MS Office) otherwise some symbols might not be displayed; some other "rich" font might work as well.

You don't have to use the browser. You can also use the command-line interface by running MetaPRL with the `-cli` option, which will expect to be running in a console with Unicode fonts. To run `mp -cli` in an `xterm` with the correct fonts, use `<metaprl>/editor/ml/mpxterm` (normal fonts), `<metaprl>/editor/ml/mpxterm-large` (larger font), or `<metaprl>/editor/ml/mpkonsole` (to run in KDE's `konsole`)

If starting MetaPRL by hands (or from Emacs), the programs to use are `<metaprl>/editor/ml/mpopt` (native code) and `<metaprl>/editor/ml/mptop` (bytecode). You will need to use a Unicode (`iso10646-1`) font to be able to interact with some sense. We currently recommend using the `-misc-fixed-medium-r-normal--15-140-75-75-c-90-iso10646-1` font (which is what `mpxterm` script uses), but in some cases another font may work better.

That's it. To navigate the MetaPRL logics and programs, you will need to read the `<metaprl>/QUICKSTART` file or the [User Guide](#). To start developing new code, read the [Developer Guide](#).

If you need help installing MetaPRL, please feel free to contact us at `metaprl` at `metaprl.org`.

MetaPRL history and people

History

MetaPRL is the latest in the PRL family of systems developed over the last 25 years as a part of the Cornell PRL project. MetaPRL was conceived in January 1996 to address the concerns of scalability in formal systems. In the early part of the decade, NuPRL formalizations were mostly limited to smaller problems that focused on basic data structures and mathematical foundations. As the foundations were implemented, it became possible to formalize larger problems, like the verification of Ensemble being performed by Kreitz, Hayden, and Hickey. As with any growing system, *scalability* became a major concern: how do we share our results with other systems; how do we design domain-specific logics; how do we get the computational speed for proving properties of large systems? These questions led to the modular design on MetaPRL, where logical theories are treated as formal objects that include both the logical knowledge and domain-specific heuristics needed to use that knowledge. As more effort was put into the system MetaPRL eventually grew into a very general modern system whose modularity on all levels gives it flexibility to support a very wide range of applications.

People

Jason Hickey is the main author and architect of MetaPRL. The system was a large part of his PhD thesis "The MetaPRL Logical Programming Environment".

Aleksey Nogin joined the MetaPRL project in 1998. After Jason graduated from Cornell in 1999, Aleksey took over the coordination of the project. The system was a large part of his PhD thesis "Theory and Implementation of an Efficient Tactic-Based Logical Framework".

Robert Constable is the head of the Cornell PRL Group.

Alexei Kopylov is working on better understanding and improving MetaPRL Type Theory. In particular, he came up with and have implemented in MetaPRL a novel theory of extensible records.

Sergei Artemov is the head of the CUNY branch of the MetaPRL group.

Xin Yu is working on formalizing abstract algebra in MetaPRL.

Lori Lorigo and Richard Eaton are working on connecting MetaPRL and NuPRL LPE systems.

Stephan Schmitt and Christoph Kreitz implemented a first order prover ("JProver") in MetaPRL.

Vladimir Krupski is the coordinator of the Moscow branch of the MetaPRL group. He is working on improving MetaPRL unification algorithms.

Yegor Bryuhov implemented term cons-hashing using weak pointers and explicit DAG representations for terms. He is currently working on the arithmetical theories in MetaPRL Type Theory.

Eli Barzilay is working on implementing reflection in NuPRL and MetaPRL.

Carl Witty have contributed an extensive reference to MetaPRL Type Theory and lots of very helpful feedback.

MetaPRL

Brian Aydemir is working on formalizing the functional intermediate representation (FIR) of the MC compiler being developed at the California Institute of Technology by Jason Hickey. The goal is to be able to reason about code and code transformations.

Adam Granicz is working on generic parsing and formal compilation using the MetaPRL rewriting engine.

Nathan Gray and Cristian Tapus are working on implementing an extensible compiler (with special features like speculations and migration) using MetaPRL.

MetaPRL License

MetaPRL is distributed under the terms of the GNU General Public License. Ensemble, OCaml, and CamlP4 are distributed under separate licenses.

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc. 675 Mass Ave, Cambridge, MA 02139, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) 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 this service 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 make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, 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 show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

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

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute 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 and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

- a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
- b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
- c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate

MetaPRL

works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

- a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
- b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
- c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

MetaPRL

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), 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 distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the 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 a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software

MetaPRL

Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, 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.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE 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.

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 convey 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 AN IDEA OF WHAT IT DOES. Copyright (C) 19YY
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 2 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

MetaPRL

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, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

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

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

Gnomovision version 69, Copyright (C) 19YY NAME OF AUTHOR Gnomovision 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, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program `Gnomovision' (which makes passes at compilers) written by James Hacker. SIGNATURE OF TY COON, 1 April 1989 Ty Coon, President of Vice

This 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 Library General Public License instead of this License.