The Parma Polyhedra Library OCaml Language Interface User's Manual[*] (version 1.1)

Roberto Bagnara[†]
Patricia M. Hill[‡]
Enea Zaffanella[§]
Abramo Bagnara[¶]

October 28, 2013

[*]This work has been partly supported by: University of Parma's FIL scientific research project (ex 60%) "Pure and Applied Mathematics"; MURST project "Automatic Program Certification by Abstract Interpretation"; MURST project "Abstract Interpretation, Type Systems and Control-Flow Analysis"; MURST project "Automatic Aggregate- and Number-Reasoning for Computing: from Decision Algorithms to Constraint Programming with Multisets, Sets, and Maps"; MURST project "Constraint Based Verification of Reactive Systems"; MURST project "Abstract Interpretation: Design and Applications"; EPSRC project "Numerical Domains for Software Analysis"; EPSRC project "Geometric Abstractions for Scalable Program Analyzers".

[†]bagnara@cs.unipr.it, Department of Mathematics, University of Parma, Italy, and BUGSENG srl.
[‡]patricia.hill@bugseng.com, BUGSENG srl.
[§]zaffanella@cs.unipr.it, Department of Mathematics, University of Parma, Italy, and BUGSENG srl.
[¶]abramo.bagnara@bugseng.com, BUGSENG srl.

For the most up-to-date information see the Parma Polyhedra Library site:

`http://bugseng.com/products/ppl/`

Contents

1 OCaml Language Interface

The Parma Polyhedra Library comes equipped with an interface for the OCaml language.

The main features of the library are described in Section OCaml Interface Features. Section OCamldoc Documentation lists all the functions available to the default generated domains in the OCaml interface. Section Compilation and Installation explains how the OCaml interface is compiled and installed.

In the sequel, `prefix` is the prefix under which you have installed the library (typically `/usr` or `/usr/local`).

OCaml Interface Features

The OCaml interface provides access to the numerical abstractions (convex polyhedra, BD shapes, octagonal shapes, etc.) implemented by the PPL library. A general introduction to the numerical abstractions, their representation in the PPL and the operations provided by the PPL is given in the main *PPL user manual*. Here we just describe those aspects that are specific to the OCaml interface.

Overview

First, here is a list of notes with general information and advice on the use of the OCaml interface.

- The numerical abstract domains available to the OCaml user consist of the *simple* domains, *powersets* of a simple domain and *products* of simple domains.
  - The simple domains are:
    - convex polyhedra, which consist of C_Polyhedron and NNC_Polyhedron;
    - weakly relational, which consist of BD_Shape_N and Octagonal_Shape_N where N is one of the numeric types short, signed_char, int, long, long_long, mpz_class, mpq_class;
    - boxes which consist of Int8_Box, Int16_Box, Int32_Box, Int64_Box, Uint8_Box, Uint16_Box, Uint32_Box, Uint64_Box, Double_Box, Long_Double_Box, Z_Box, Rational_Box, Float_Box; and
    - the Grid domain.
  - The powerset domains are Pointset_Powerset_S where S is a simple domain.
  - The product domains consist of Direct_Product_S_T, Smash_Product_S_T and Constraints_Product_S_T where S and T are simple domains.
- In the following, any of the above numerical abstract domains is called a PPL *domain* and any element of a PPL domain is called a *PPL object*.
- The OCaml interface files are all installed in the directory `prefix/lib/ppl`. Since this includes shared and dynamically loaded libraries, you must make your dynamic linker/loader aware of this fact. If you use a GNU/Linux system, try the commands `man ld.so` and `man ldconfig` for more information.
- A PPL object such as a polyhedron can only be accessed by means of a OCaml term called a *handle*. Note, however, that the data structure of a handle, is implementation-dependent, system-dependent and version-dependent, and, for this reason, deliberately left unspecified. What we do guarantee is that the handle requires very little memory.
- An OCaml program can obtain a valid handle for a PPL object by using functions such as

  ```
  ppl_new_C_Polyhedron_from_space_dimension,
  ppl_new_C_Polyhedron_from_C_Polyhedron,
  ppl_new_C_Polyhedron_from_constraints,
  ppl_new_C_Polyhedron_from_generators.
  ```

  These functions will return a new handle for referencing a PPL polyhedron.

- For a PPL object with space dimension k, the identifiers used for the PPL variables must lie between 0 and $k-1$ and correspond to the indices of the associated Cartesian axes. For example, when using the functions that combine PPL polyhedra or add constraints or generators to a representation of a PPL polyhedron, the polyhedra referenced and any constraints or generators in the call should follow all the (space) dimension-compatibility rules stated in Section *Representations of Convex Polyhedra* of the main PPL user manual.
- As explained above, a polyhedron has a fixed topology C or NNC, that is determined at the time of its initialization. All subsequent operations on the polyhedron must respect all the topological compatibility rules stated in Section *Representations of Convex Polyhedra* of the main PPL user manual.
- Any application using the PPL should make sure that only the intended version(s) of the library are ever used. Functions

  ```
  ppl_version_major,
  ppl_version_minor,
  ppl_version_revision,
  ppl_version_beta,
  ppl_version,
  ppl_banner.
  ```

  allow run-time checking of information about the version being used.

Function Descriptions

Below is a short description of many of the interface functions. For full definitions of terminology used here, see the main PPL user manual.

Domain Independent Functions

First we describe some domain independent functions included with all instantiations of the OCaml interfaces.

`ppl_version_major`

Returns the major number of the PPL version.

`ppl_version_minor`

Returns the minor number of the PPL version.

`ppl_version_revision`

Returns the revision number of the PPL version.

`ppl_version_beta`

Returns the beta number of the PPL version.

`ppl_version`

Returns the PPL version.

`ppl_banner`

Returns information about the PPL version, the licensing, the lack of any warranty whatsoever, the C++ compiler used to build the library, where to report bugs and where to look for further information.

ppl_max_space_dimension

Returns the maximum space dimension the C++ interface can handle.

ppl_Coefficient_bits

Returns the number of bits used in the C++ interface for PPL coefficients; 0 if unbounded.

ppl_Coefficient_is_bounded

Returns true if and only if the coefficients in the C++ interface are bounded.

ppl_Coefficient_max

If the coefficients are bounded, returns the maximum coefficient the C++ interface can handle.

ppl_Coefficient_min

If the coefficients are bounded, returns the minimum coefficient the C++ interface can handle.

ppl_io_wrap_string source_string indent_depth preferred_first_line_length preferred_line_length

Utility function for the wrapping of lines of text. The function wraps the lines of text stored in its first string argument according to the next three integer arguments, which are interpreted as the indentation depth, the preferred length for the first line and the preferred length for all the other lines, respecively; it returns a string containing the wrapped text.

ppl_set_timeout csecs

Computations taking exponential time will be interrupted some time after `csecs` centiseconds after that call. If the computation is interrupted that way, a timeout exception will be thrown. An exception is immediately thrown if `csecs` is not strictly greater than zero, or if the PPL Watchdog library is not enabled.

ppl_reset_timeout

Resets the timeout time so that the computation is not interrupted. An exception is thrown if the PPL Watchdog library is not enabled.

ppl_set_deterministic_timeout unscaled_weight scale

Computations taking exponential time will be interrupted some time after reaching the complexity threshold $\text{weight} = \text{unscaled\_weight} \cdot 2^{\text{scale}}$. If the computation is interrupted that way, a timeout exception will be thrown. `unscaled_weight` must be strictly greater than zero; `scale` must be non-negative; an exception is thrown if the computed weight threshold exceeds the maximum allowed value. *NOTE:* This "timeout" checking functionality is said to be *deterministic* because it is not based on actual elapsed time. Its behavior will only depend on (some of the) computations performed in the PPL library and it will be otherwise independent from the computation environment (CPU, operating system, compiler, etc.). The weight mechanism is under beta testing: client applications should be ready to reconsider the tuning of these weight thresholds when upgrading to newer version of the PPL.

ppl_reset_deterministic_timeout

Resets the timeout time so that the computation is not interrupted. An exception is thrown if the PPL Watchdog library is not enabled.

ppl_set_rounding_for_PPL

Sets the FPU rounding mode so that the PPL abstractions based on floating point numbers work correctly. This is performed automatically at initialization-time. Calling this function is needed only if `restore_pre_PPL_rounding` has previously been called.

ppl_restore_pre_PPL_rounding

Sets the FPU rounding mode as it was before initialization of the PPL. After calling this function it is absolutely necessary to call `set_rounding_for_PPL` before using any PPL abstractions based on floating point numbers. This is performed automatically at finalization-time.

ppl_irrational_precision

Returns the precision parameter for irrational calculations.

ppl_set_irrational_precision

Sets the precision parameter `p` for irrational calculations. In the following irrational calculations returning an unbounded rational (e.g., when computing a square root), the lesser between numerator and denominator will be limited to `2**p`.

MIP Functions

Here we describe some functions available for PPL objects defining mixed integer (linear) programming problems.

ppl_new_MIP_Problem_from_space_dimension dimension

Return a handle to an MIP Problem MIP with the feasible region the vector space of dimension `dimension`, objective function 0 and optimization mode `max`.

ppl_new_MIP_Problem dimension constraint_system lin_expr optimization_mode

Return a handle to an MIP Problem MIP having space dimension `dimension`, a feasible region represented by `constraint_system`, objective function `lin_expr` and optimization mode `optimization_mode`.

ppl_MIP_Problem_get_control_parameter handle param_name

Returns the value of the control parameter named `param_name`.

ppl_MIP_Problem_set_control_parameter handle param_value

Sets control parameter value `param_value`.

ppl_MIP_Problem_swap handle_1 handle_2

Swaps the MIP Problem referenced by `handle_1` with the one referenced by `handle_2`.

ppl_MIP_Problem_space_dimension handle

Returns the dimension of the vector space in which the MIP Problem referenced by `handle` is embedded.

ppl_MIP_Problem_integer_space_dimensions handle

Returns a list of variables representing representing the integer space dimensions of the MIP Problem referenced by `handle`.

ppl_MIP_Problem_constraints handle

Returns a list of the constraints in the constraints system representing the feasible region for the MIP Problem referenced by `handle`.

ppl_MIP_Problem_objective_function handle

Returns the objective function for the MIP Problem referenced by `handle`.

ppl_MIP_Problem_optimization_mode handle

Returns the optimization mode for the MIP Problem referenced by `handle`.

ppl_MIP_Problem_clear handle

Resets the MIP problem referenced by `handle` to be the trivial problem with the feasible region the 0-dimensional universe, objective function 0 and optimization mode `Maximization`.

ppl_MIP_Problem_add_space_dimensions_and_embed handle dimension

Embeds the MIP problem referenced by `handle` in a space that is enlarged by `dimension` dimensions,

ppl_MIP_Problem_add_to_integer_space_dimensions handle vars_list

Updates the MIP Problem referenced by `handle` so that the variables in `vars_list` are added to the set of integer space dimensions.

ppl_MIP_Problem_add_constraint handle constraint

Updates the MIP Problem referenced by `handle` so that the feasible region is represented by the original constraint system together with the constraint `constraint`.

ppl_MIP_Problem_add_constraints handle constraint_system

Updates the MIP Problem referenced by `handle` so that the feasible region is represented by the original constraint system together with all the constraints in `constraint_system`.

ppl_MIP_Problem_set_objective_function handle lin_expr

Updates the MIP Problem referenced by `handle` so that the objective function is changed to `lin_expr`.

ppl_MIP_Problem_set_optimization_mode handle optimization_mode

Updates the MIP Problem referenced by `handle` so that the optimization mode is changed to `optimization-_mode`.

ppl_MIP_Problem_is_satisfiable handle

Returns true if the MIP Problem referenced by `handle` is satisfiable and false otherwise.

ppl_MIP_Problem_solve handle

Solves the MIP problem referenced by `handle` and returns 0, if the MIP problem is not satisfiable; 1, if the MIP problem is satisfiable but there is no finite bound to the value of the objective function; 2, if the MIP problem admits an optimal solution.

ppl_MIP_Problem_feasible_point handle

Returns a feasible point for the MIP problem referenced by `handle`.

ppl_MIP_Problem_optimizing_point handle

Returns an optimizing point for the MIP problem referenced by `handle`.

ppl_MIP_Problem_optimal_value handle

Returns a pair of numbers, the first being the numerator and the second the denominator, for the optimal value for the MIP problem referenced by `handle`.

ppl_MIP_Problem_evaluate_objective_function handle generator

Evaluates the objective function of the MIP problem referenced by `handle` at point `generator`. Returns a pair of numbers, the first being the numerator and the second the denominator, for the objective function value for the MIP problem referenced by `handle`.

ppl_MIP_Problem_OK handle

Returns true if the MIP Problem referenced by `handle` is well formed, i.e., if it satisfies all its implementation invariants and false, otherwise. Useful for debugging purposes.

ppl_MIP_Problem_ascii_dump handle

Returns a string containing an ASCII dump of the internal representation of the MIP_Problem referenced by `handle`. Useful for debugging purposes.

PIP Functions

Here we describe some functions available for PPL objects defining parametric integer programming problems.

ppl_new_PIP_Problem_from_space_dimension dimension

Return a handle to a PIP Problem PIP with the feasible region the vector space of dimension `dimension`, empty constraint_system and empty set of parametric variables.

ppl_new_PIP_Problem dimension constraint_system vars_list

Return a handle to a PIP Problem PIP having space dimension `dimension`, a feasible region represented by `constraint_system` and parametric variables represented by `vars_list`.

ppl_PIP_Problem_get_control_parameter handle param_name

Returns the value of the control parameter named `param_name`.

ppl_PIP_Problem_set_control_parameter handle param_value

Sets control parameter value `param_value`.

ppl_PIP_Problem_swap handle_1 handle_2

Swaps the PIP Problem referenced by `handle_1` with the one referenced by `handle_2`.

ppl_PIP_Problem_space_dimension handle

Returns the dimension of the vector space in which the PIP Problem referenced by `handle` is embedded.

ppl_PIP_Problem_parameter_space_dimensions handle

Returns a list of variables representing representing the parameter space dimensions of the PIP Problem referenced by `handle`.

ppl_PIP_Problem_constraints handle

Returns a list of the constraints in the constraints system representing the feasible region for the PIP Problem referenced by `handle`.

ppl_PIP_Problem_clear handle

Resets the PIP problem referenced by `handle` to be the trivial problem with space dimension 0.

ppl_PIP_Problem_add_space_dimensions_and_embed handle dimension_0 dimension_1

Embeds the PIP problem referenced by `handle` in a space that is enlarged by `dimension_0` non-parameter dimensions and `dimension_1` parameter dimensions,

ppl_PIP_Problem_add_to_parameter_space_dimensions handle vars_list

Sets the space dimensions whose indexes are in `vars_list` to be parameter space dimensions.

ppl_PIP_Problem_add_constraint handle constraint

Updates the PIP Problem referenced by `handle` so that the feasible region is represented by the original constraint system together with the constraint `constraint`.

ppl_PIP_Problem_add_constraints handle constraint_system

Updates the PIP Problem referenced by `handle` so that the feasible region is represented by the original constraint system together with all the constraints in `constraint_system`.

ppl_PIP_Problem_set_big_parameter_dimension handle dimension

Sets the dimension for the big parameter to `dimension`.

ppl_PIP_Problem_get_big_parameter_dimension handle

Returns the dimension for the big parameter. Exception is thrown if no big parameter dimension has been set.

ppl_PIP_Problem_has_big_parameter_dimension handle

Returns true if and only if the dimension for the big parameter has been set.

ppl_PIP_Problem_is_satisfiable handle

Returns true if the PIP Problem referenced by `handle` is satisfiable and false otherwise.

ppl_PIP_Problem_solve handle

Solves the PIP problem referenced by `handle` and returns a status flag indicating the outcome of the optimization attempt: `Optimized_Pip_Problem` if the optimization attempt succeeds; `Unfeasible_Pip_Problem` otherwise.

ppl_PIP_Problem_solution handle

Solves the PIP problem referenced by `handle` and returns a handle to a PIP_Tree representing a feasible solution, if it exists and bottom otherwise.

ppl_PIP_Problem_optimizing_solution handle

Solves the PIP problem referenced by `handle` and returns a handle to a PIP_Tree representing an optimizing_solution, if it exists and bottom otherwise.

ppl_PIP_Problem_OK handle

Returns true if the PIP Problem referenced by `handle` is well formed, i.e., if it satisfies all its implementation invariants and false, otherwise. Useful for debugging purposes.

ppl_PIP_Problem_ascii_dump handle

Returns a string containing an ASCII dump of the internal representation of the PIP_Problem referenced by `handle`. Useful for debugging purposes.

ppl_PIP_Tree_Node_swap handle_1 handle_2

Swaps the PIP tree node referenced by `handle_1` with the one referenced by `handle_2`.

ppl_PIP_Tree_Node_OK handle

Returns true if the PIP tree node referenced by `handle` is well formed, i.e., if it satisfies all its implementation invariants and false, otherwise. Useful for debugging purposes.

ppl_PIP_Tree_Node_ascii_dump handle

Returns a string containing an ASCII dump of the internal representation of the Pip tree node referenced by `handle`. Useful for debugging purposes.

ppl_PIP_Tree_Node_constraints handle

Returns a list of the parameter constraints in the PIP tree node referenced by `handle`.

ppl_PIP_Tree_Node_artificials handle

Returns a list of the artificial parameters in the PIP tree node referenced by `handle`.

ppl_PIP_Tree_Node_is_bottom handle

Returns true if and only if `handle` represents bottom.

ppl_PIP_Tree_Node_is_decision handle

Returns true if and only if `handle` represents a decision node.

ppl_PIP_Tree_Node_is_solution handle

Returns true if and only if `handle` represents a solution node.

ppl_PIP_Tree_Node_parametric_values handle var

Returns a linear expression representing the values of problem variable `var` in the solution node represented by `handle`. The returned linear expression may involve problem parameters as well as artificial parameters.

ppl_PIP_Tree_Node_true_child handle var

Returns a handle to the child on the true branch of the PIP tree node represented by `handle`.

ppl_PIP_Tree_Node_false_child handle var

Returns a handle to the child on the false branch of the PIP tree node represented by `handle`.

C_Polyhedron Functions

Here we describe the main functions available for PPL objects defining convex and closed polyhedra.

ppl_new_C_Polyhedron_from_space_dimension space_dimension universe_or_empty

Returns a handle to a C polyhedron $\mathcal{P}$ with `space_dimension` dimensions; it is empty or the universe polyhedron depending on whether `universe_or_empty` is `empty` or `universe`, respectively.

ppl_new_C_Polyhedron_from_C_Polyhedron handle

If `handle` refers to a C polyhedron $\mathcal{P}_1$, then the expression will returns a handle to a copy $\mathcal{P}_2$ of $\mathcal{P}_1$.

ppl_new_C_Polyhedron_from_NNC_Polyhedron handle

If `handle` refers to an NNC polyhedron $\mathcal{P}_1$, then the expression returns a handle to a copy $\mathcal{P}_2$ of $\mathcal{P}_1$.

When using `ppl_new_C_Polyhedron_from_NNC_Polyhedron/2`, care must be taken that the source polyhedron referenced by `handle` is topologically closed.

ppl_new_C_Polyhedron_from_constraints constraint_system

Returns a handle to a C polyhedron $\mathcal{P}$ represented by `constraint_system`.

ppl_new_C_Polyhedron_from_generators generator_system

Returns a handle to a C polyhedron $\mathcal{P}$ represented by `generator_system`.

ppl_Polyhedron_swap handle_1 handle_2

Swaps the polyhedron $\mathcal{P}$ referenced by `handle_1` with the polyhedron $\mathcal{Q}$ referenced by `handle_2`. The polyhedra $\mathcal{P}$ and $\mathcal{Q}$ must have the same topology.

ppl_Polyhedron_space_dimension handle

Returns the dimension of the vector space in which the polyhedron referenced by `handle` is embedded.

ppl_Polyhedron_affine_dimension handle

Returns the actual dimension of the polyhedron referenced by `handle`.

ppl_Polyhedron_get_constraints handle

Return a list of the constraints in the constraints system representing the polyhedron referenced by `handle`.

ppl_Polyhedron_get_minimized_constraints handle

Returns a minimized list of the constraints in the constraints system representing the polyhedron referenced by `handle`.

ppl_Polyhedron_get_generators handle

Returns a list of the generators in the generators system representing the polyhedron referenced by `handle`.

ppl_Polyhedron_get_minimized_generators handle

Returns a minimized list of the generators in the generators system representing the polyhedron referenced by `handle`.

ppl_Polyhedron_relation_with_constraint handle constraint

Returns the list of relations the polyhedron referenced by `handle` has with `constraint`. The possible relations and their meaning is given in Section *Relation-With Operators* of the main PPL user manual.

ppl_Polyhedron_relation_with_generator handle generator

Returns the list of relations the polyhedron referenced by `handle` has with `generator`. The possible relations and their meaning is given in Section *Relation-With Operators* of the main PPL user manual.

ppl_Polyhedron_is_empty handle

Returns true if the polyhedron referenced by `handle` is empty and false, otherwise.

ppl_Polyhedron_is_universe handle

Returns true if the polyhedron referenced by `handle` is the universe and false, otherwise.

ppl_Polyhedron_is_bounded handle

Returns true if the polyhedron referenced by `handle` is bounded and false, otherwise.

ppl_Polyhedron_contains_integer_point handle

Returns true if the polyhedron referenced by `handle` contains at least one integer point and false, otherwise.

ppl_Polyhedron_bounds_from_above handle lin_expr

Returns true if the polyhedron referenced by `handle` is bounded from above by `lin_expr` and false, otherwise.

ppl_Polyhedron_bounds_from_below handle lin_expr

Returns true if the polyhedron referenced by `handle` is bounded from below by `lin_expr` and false, otherwise.

ppl_Polyhedron_maximize handle lin_expr

Returns a record `bool_1 * coefficient_1 * coefficient_2 * bool_2` where: `bool_1` is true if the polyhedron P referenced by `handle` is not empty and `lin_expr` is bounded from above in P and false, otherwise. `coefficient_1` is the numerator of the supremum value and `coefficient_2` the denominator of the supremum value. If the supremum is also the maximum, `bool_2` is true and false, otherwise.

ppl_Polyhedron_maximize_with_point handle lin_expr

Returns a record `bool_1 * coefficient_1 * coefficient_2 * bool_2 * Point` `bool_1` is true if the polyhedron P referenced by `handle` is not empty and `lin_expr` is bounded from above in P and false, otherwise. `coefficient_1` is the numerator of the supremum value and `coefficient_2` the denominator of the supremum value. If the supremum is also the maximum, `bool_2` is true and false, otherwise. `Point` is the point or closure point where `lin_expr` reaches the supremum.

ppl_Polyhedron_minimize handle lin_expr

Returns a record `bool_1 * coefficient_1 * coefficient_2 * bool_2` `bool_1` is true if the polyhedron P referenced by `handle` is not empty and `lin_expr` is bounded from below in P and false, otherwise. `coefficient_1` is the numerator of the infinum value and `coefficient_2` the denominator of the infinum value. If the infinum is also the minimum, `bool_2` is true and false, otherwise.

ppl_Polyhedron_minimize_with_point handle lin_expr

Returns a record `bool_1 * coefficient_1 * coefficient_2 * bool_2` `bool_1` is true if the polyhedron P referenced by `handle` is not empty and `lin_expr` is bounded from below in P and false, otherwise. `coefficient_1` is the numerator of the infinum value and `coefficient_2` the denominator of the infinum value. If the infinum is also the minimum, `bool_2` is true and false, otherwise. `Point` is the point or closure point where `lin_expr` reaches the infinum.

ppl_Polyhedron_is_topologically_closed handle

Returns true if the polyhedron referenced by `handle` is topologically closed and false, otherwise.

ppl_Polyhedron_contains_Polyhedron handle_1 handle_2

Returns true if the polyhedron referenced by `handle_2` is included in or equal to the polyhedron referenced by `handle_1` and false, otherwise.

ppl_Polyhedron_strictly_contains_Polyhedron handle_1 handle_2

Returns true if the polyhedron referenced by `handle_2` is included in but not equal to the polyhedron referenced by `handle_1` and false, otherwise.

ppl_Polyhedron_is_disjoint_from_Polyhedron handle_1 handle_2

Returns true if the polyhedron referenced by `handle_1` is disjoint from the polyhedron referenced by `handle_2` and false, otherwise.

ppl_Polyhedron_equals_Polyhedron handle_1 handle_2

Returns true if the polyhedron referenced by `handle_1` is equal to the polyhedron referenced by `handle_2` and false, otherwise.

ppl_Polyhedron_OK handle

Returns true if the polyhedron referenced by `handle` is well formed, i.e., if it satisfies all its implementation invariants and false, otherwise. Useful for debugging purposes.

ppl_Polyhedron_add_constraint handle constraint

Updates the polyhedron referenced by `handle` to one obtained by adding `constraint` to its constraint system.

ppl_Polyhedron_add_generator handle generator

Updates the polyhedron referenced by `handle` to one obtained by adding `generator` to its generator system.

ppl_Polyhedron_add_constraints handle constraint_system

Updates the polyhedron referenced by `handle` to one obtained by adding to its constraint system the constraints in `constraint_system`.

ppl_C_Polyhedron_add_generators handle generator_system

Updates the polyhedron referenced by `handle` to one obtained by adding to its generator system the generators in `generator_system`.

ppl_Polyhedron_intersection_assign handle_1 handle_2

Assigns to the polyhedron referenced by `handle_1` its intersection with the polyhedron referenced by `handle_2`.

ppl_Polyhedron_poly_hull_assign handle_1 handle_2

Assigns to the polyhedron referenced by `handle_1` its poly-hull with the polyhedron referenced by `handle_2`.

ppl_Polyhedron_poly_difference_assign handle_1 handle_2

Assigns to the polyhedron referenced by `handle_1` its poly-difference with the polyhedron referenced by `handle_2`.

ppl_Polyhedron_affine_image handle var lin_expr coefficient

Transforms the polyhedron referenced by `handle` assigning the affine expression `lin_expr`/`coefficient` to `var`.

ppl_Polyhedron_affine_preimage handle var lin_expr coefficient

This is the inverse transformation to that for `ppl_affine_image`.

ppl_Polyhedron_bounded_affine_image handle var lin_expr_1 lin_expr_2 coefficient

Transforms the polyhedron referenced by `handle` assigning the image with respect to the transfer relation `lin_expr_1/coefficient <= var <= lin_expr_2/coefficient`.

ppl_Polyhedron_generalized_affine_image handle var Relation_Symbol lin_expr coefficient

Transforms the polyhedron referenced by `handle` assigning the generalized affine image with respect to the transfer function `var Relation_Symbol lin_expr/coefficient`.

ppl_Polyhedron_generalized_affine_image_lhs_rhs handle lin_expr_1 Relation_Symbol lin_expr_2

Transforms the polyhedron referenced by `handle` assigning the generalized affine image with respect to the transfer function `lin_expr_1 Relation_Symbol lin_expr_2`.

ppl_Polyhedron_time_elapse_assign handle_1 handle_2

Assigns to the polyhedron $\mathcal{P}$ referenced by `handle_1` the time-elapse ($\mathcal{P} \nearrow \mathcal{Q}$) with the polyhedron $\mathcal{Q}$ referenced by `handle_2`.

ppl_Polyhedron_BHRZ03_widening_assign handle_1 handle_2

If the polyhedron $\mathcal{P}_1$ referenced by `handle_1` contains the polyhedron $\mathcal{P}_2$ referenced by `handle_2`, then `handle_1` will refer to the BHRZ03-widening of $\mathcal{P}_1$ with $\mathcal{P}_2$.

ppl_Polyhedron_BHRZ03_widening_assign_with_tokens handle_1 handle_2 c_unsigned_1

It is assumed that the polyhedron $\mathcal{P}_1$ referenced by `handle_1` contains the polyhedron $\mathcal{P}_2$ referenced by `handle_2`; let $\mathcal{P}$ denote the BHRZ03-widening of $\mathcal{P}_1$ with $\mathcal{P}_2$, Assuming that the quantity t_1 given by `c_unsigned_1` is the number of tokens available, Then this function will return the number of tokens remaining at the end of the operation.

ppl_Polyhedron_limited_BHRZ03_extrapolation_assign handle_1 handle_2 constraint_system

If the polyhedron $\mathcal{P}_1$ referenced by `handle_1` contains the polyhedron $\mathcal{P}_2$ referenced by `handle_2`, then `handle_1` will refer to the BHRZ03-extrapolation of $\mathcal{P}_1$ with $\mathcal{P}_2$ improved by enforcing the constraints in `constraint_system`.

ppl_Polyhedron_limited_BHRZ03_extrapolation_assign_with_tokens handle_1 handle_2 constraint_system c_unsigned_1

It is assumed that the polyhedron $\mathcal{P}_1$ referenced by `handle_1` contains the polyhedron $\mathcal{P}_2$ referenced by `handle_2`; let $\mathcal{P}$ denote the BHRZ03-extrapolation of $\mathcal{P}_1$ with $\mathcal{P}_2$, improved by enforcing those constraints in `constraint_system`.

Assuming that the quantity t_1 given by `c_unsigned_1` is the number of tokens available, then this function will return the number of tokens t_2 remaining at the end of the operation.

ppl_Polyhedron_bounded_BHRZ03_extrapolation_assign handle_1 handle_2 constraint-_system

If the polyhedron $\mathcal{P}_1$ referenced by `handle_1` contains the polyhedron $\mathcal{P}_2$ referenced by `handle_2`, then `handle_1` will refer to the BHRZ03-extrapolation of $\mathcal{P}_1$ with $\mathcal{P}_2$ improved by enforcing the constraints in `constraint_system` together with all constraints of the form $\pm x \leq r$ and $\pm x < r$ that are satisfied by every point in $\mathcal{P}_1$.

ppl_Polyhedron_bounded_BHRZ03_extrapolation_assign_with_tokens handle_1 handle-_2 constraint_system c_unsigned_1

It is assumed that the polyhedron $\mathcal{P}_1$ referenced by `handle_1` contains the polyhedron $\mathcal{P}_2$ referenced by `handle_2`; let $\mathcal{P}$ denote the BHRZ03-extrapolation of $\mathcal{P}_1$ with $\mathcal{P}_2$ improved by enforcing those constraints in `constraint_system` together with all constraints of the form $\pm x \leq r$ and $\pm x < r$ that are satisfied by every point in $\mathcal{P}_1$.

Assuming that the quantity t_1 given by `c_unsigned_1` is the number of tokens available, this function will return the number of tokens t_2 remaining at the end of the operation.

ppl_Polyhedron_H79_widening_assign handle_1 handle_2

If the polyhedron $\mathcal{P}_1$ referenced by `handle_1` contains the polyhedron $\mathcal{P}_2$ referenced by `handle_2`, then `handle_1` will refer to the H79-widening of $\mathcal{P}_1$ with $\mathcal{P}_2$.

ppl_Polyhedron_H79_widening_assign_with_tokens handle_1 handle_2 c_unsigned-_1

It is assumed that the polyhedron $\mathcal{P}_1$ referenced by `handle_1` contains the polyhedron $\mathcal{P}_2$ referenced by `handle_2`; let $\mathcal{P}$ denote the H79-widening of $\mathcal{P}_1$ with $\mathcal{P}_2$, Assuming that the quantity t_1 given by `c_unsigned_1` is the number of tokens available, Then this function will return the number of tokens remaining at the end of the operation.

ppl_Polyhedron_limited_H79_extrapolation_assign handle_1 handle_2 constraint-_system

If the polyhedron $\mathcal{P}_1$ referenced by `handle_1` contains the polyhedron $\mathcal{P}_2$ referenced by `handle_2`, then `handle_1` will refer to the H79-extrapolation of $\mathcal{P}_1$ with $\mathcal{P}_2$ improved by enforcing the constraints in `constraint_system`.

ppl_Polyhedron_limited_H79_extrapolation_assign_with_tokens handle_1 handle-_2 constraint_system c_unsigned_1

It is assumed that the polyhedron $\mathcal{P}_1$ referenced by `handle_1` contains the polyhedron $\mathcal{P}_2$ referenced by `handle_2`; let $\mathcal{P}$ denote the H79-extrapolation of $\mathcal{P}_1$ with $\mathcal{P}_2$, improved by enforcing those constraints in `constraint_system`.

Assuming that the quantity t_1 given by `c_unsigned_1` is the number of tokens available, then this function will return the number of tokens t_2 remaining at the end of the operation.

ppl_Polyhedron_bounded_H79_extrapolation_assign handle_1 handle_2 constraint-_system

If the polyhedron $\mathcal{P}_1$ referenced by `handle_1` contains the polyhedron $\mathcal{P}_2$ referenced by `handle_2`, then `handle_1` will refer to the H79-extrapolation of $\mathcal{P}_1$ with $\mathcal{P}_2$ improved by enforcing the constraints in `constraint_system` together with all constraints of the form $\pm x \leq r$ and $\pm x < r$ that are satisfied by every point in $\mathcal{P}_1$.

ppl_Polyhedron_bounded_H79_extrapolation_assign_with_tokens handle_1 handle_2 constraint_system c_unsigned_1

It is assumed that the polyhedron $\mathcal{P}_1$ referenced by `handle_1` contains the polyhedron $\mathcal{P}_2$ referenced by `handle_2`; let $\mathcal{P}$ denote the H79-extrapolation of $\mathcal{P}_1$ with $\mathcal{P}_2$, improved by enforcing those constraints in `constraint_system` together with all constraints of the form $\pm x \leq r$ and $\pm x < r$ that are satisfied by every point in $\mathcal{P}_1$.

Assuming that the quantity t_1 given by `c_unsigned_1` is the number of tokens available, this function will return the number of tokens t_2 remaining at the end of the operation.

ppl_Polyhedron_topological_closure_assign handle

Assigns to the polyhedron referenced by `handle` its topological closure.

ppl_Polyhedron_add_space_dimensions_and_embed handle space_dimension

Embeds the polyhedron referenced by `handle` in a space that is enlarged by `space_dimension` dimensions,

ppl_Polyhedron_concatenate_assign handle_1 handle_2

Updates the polyhedron $\mathcal{P}_1$ referenced by `handle_1` by first embedding $\mathcal{P}_1$ in a new space enlarged by the space dimensions of the polyhedron $\mathcal{P}_2$ referenced by `handle_2`, and then adds to its system of constraints a renamed-apart version of the constraints of $\mathcal{P}_2$.

ppl_Polyhedron_add_space_dimensions_and_project handle space_dimension

Projects the polyhedron referenced by `handle` onto a space that is enlarged by `space_dimension` dimensions,

ppl_Polyhedron_remove_space_dimensions handle Int_List

Removes the space dimensions given by the identifiers of the PPL variables in list `Int_List` from the polyhedron referenced by `handle`. The identifiers for the remaining PPL variables are renumbered so that they are consecutive and the maximum index is less than the number of dimensions.

ppl_Polyhedron_remove_higher_space_dimensions handle space_dimension

Projects the polyhedron referenced to by `handle` onto the first `space_dimension` dimensions.

ppl_Polyhedron_expand_space_dimension handle var space_dimension

`space_dimension` copies of the space dimension referenced by variable `var` are added to the polyhedron referenced to by `handle`.

ppl_Polyhedron_fold_space_dimensions handle list_of_vars var

The space dimensions referenced by the PPL variables in list `list_of_vars` are folded into the dimension referenced by `var` and removed. The result is undefined if `list_of_vars` does not have the properties described in Section *Folding Multiple Dimensions of the Vector Space into One Dimension* of the main PPL user manual.

ppl_Polyhedron_map_space_dimensions handle p_func

Maps the space dimensions of the polyhedron referenced by `handle` using the partial function defined by a list of pairs of integers `p_func`. The result is undefined if `p_func` does not encode a partial function with the properties described in Section *Mapping the Dimensions of the Vector Space* of the main PPL user manual.

ppl_Polyhedron_wrap_assign handle list_of_vars width representation overflow constraint_system complexity_threshold wrap_indicator

Transforms the polyhedron referenced by `handle` by wrapping the dimensions given by `list_of_vars` while respecting the specified `width`, `representation` and `overflow` behavior of all these variables. The parameter `constraint_system` represents the conditional or looping construct guard with respect to which wrapping is performed. The non-negative integer `complexity_threshold` and Boolean `wrap_indicator` allow control of the complexity/precision ratio; higher values for `complexity_threshold` will lead to possibly greater precision while a true value for `wrap_indicator` indicates that the space dimensions should be wrapped individually. See Section *Wrapping Operator* for a more detailed description of this operator.

ppl_Polyhedron_ascii_dump handle

Returns a string containing an ASCII dump of the internal representation of the polyhedron referenced by `handle`. Useful for debugging purposes.

OCamldoc Documentation

NOTE: the complete documentation for module `Ppl_ocaml`, including all the types and functions that were enabled at configuration time, is only available in the *configuration dependent* OCamldoc documentation. The configuration independent OCamldoc documentation only contains those types and functions that are always enabled, which are grouped into module `Ppl_ocaml_globals`. Also note that module `Ppl_ocaml` automatically includes module `Ppl_ocaml_globals`.

2 Module `Ppl_ocaml_globals`

```
exception PPL_arithmetic_overflow of string
exception PPL_timeout_exception
exception PPL_internal_error of string
exception PPL_unknown_standard_exception of string
exception PPL_unexpected_error of string
type degenerate_element =
  | Universe
  | Empty
type linear_expression =
  | Variable of int
  | Coefficient of Gmp.Z.t
  | Unary_Plus of linear_expression
  | Unary_Minus of linear_expression
  | Plus of linear_expression * linear_expression
  | Minus of linear_expression * linear_expression
  | Times of Gmp.Z.t * linear_expression
type linear_constraint =
```

```
  | Less_Than of linear_expression * linear_expression
  | Less_Or_Equal of linear_expression * linear_expression
  | Equal of linear_expression * linear_expression
  | Greater_Than of linear_expression * linear_expression
  | Greater_Or_Equal of linear_expression * linear_expression
type linear_generator =
  | Line of linear_expression
  | Ray of linear_expression
  | Point of linear_expression * Gmp.Z.t
  | Closure_Point of linear_expression * Gmp.Z.t
type linear_grid_generator =
  | Grid_Line of linear_expression
  | Grid_Parameter of linear_expression * Gmp.Z.t
  | Grid_Point of linear_expression * Gmp.Z.t
type poly_gen_relation =
  | Subsumes
type poly_con_relation =
  | Is_Disjoint
  | Strictly_Intersects
  | Is_Included
  | Saturates
type relation_with_congruence =
  | Is_Disjoint
  | Strictly_Intersects
  | Is_Included
type linear_congruence = linear_expression * linear_expression *
  Gmp.Z.t
type constraint_system = linear_constraint list
type generator_system = linear_generator list
type grid_generator_system = linear_grid_generator list
type congruence_system = linear_congruence list
type relation_symbol =
  | Less_Than_RS
  | Less_Or_Equal_RS
  | Equal_RS
  | Greater_Than_RS
  | Greater_Or_Equal_RS
type bounded_integer_type_overflow =
  | Overflow_Wraps
  | Overflow_Undefined
  | Overflow_Impossible
type bounded_integer_type_representation =
  | Unsigned
  | Signed_2_Complement
type bounded_integer_type_width =
  | Bits_8
  | Bits_16
  | Bits_32
  | Bits_64
  | Bits_128
```

```
type complexity_class =
  | Polynomial_Complexity
  | Simplex_Complexity
  | Any_Complexity
type optimization_mode =
  | Minimization
  | Maximization
type mip_problem_status =
  | Unfeasible_Mip_Problem
  | Unbounded_Mip_Problem
  | Optimized_Mip_Problem
type control_parameter_name =
  | Pricing
type control_parameter_value =
  | Pricing_Steepest_Edge_Float
  | Pricing_Steepest_Edge_Exact
  | Pricing_Textbook
type pip_problem_status =
  | Unfeasible_Pip_Problem
  | Optimized_Pip_Problem
type pip_problem_control_parameter_name =
  | Cutting_Strategy
  | Pivot_Row_Strategy
type pip_problem_control_parameter_value =
  | Cutting_Strategy_First
  | Cutting_Strategy_Deepest
  | Cutting_Strategy_All
  | Pivot_Row_Strategy_First
  | Pivot_Row_Strategy_Max_Column
val ppl_version_major : unit -> int
val ppl_version_minor : unit -> int
val ppl_version_revision : unit -> int
val ppl_version_beta : unit -> int
val ppl_version : unit -> string
val ppl_banner : unit -> string
val ppl_io_wrap_string : string -> int -> int -> int -> string
val ppl_max_space_dimension : unit -> int
val ppl_Coefficient_bits : unit -> int
val ppl_Coefficient_is_bounded : unit -> bool
val ppl_Coefficient_max : unit -> Gmp.Z.t
val ppl_Coefficient_min : unit -> Gmp.Z.t
val ppl_Linear_Expression_is_zero : linear_expression -> bool
val ppl_Linear_Expression_all_homogeneous_terms_are_zero :
  linear_expression -> bool
val ppl_set_rounding_for_PPL : unit -> unit
val ppl_restore_pre_PPL_rounding : unit -> unit
val ppl_irrational_precision : unit -> int
val ppl_set_irrational_precision : int -> unit
```

```
val ppl_set_timeout : int -> unit
val ppl_reset_timeout : unit -> unit
val ppl_set_deterministic_timeout : int -> int -> unit
val ppl_reset_deterministic_timeout : unit -> unit
type mip_problem
val ppl_new_MIP_Problem_from_space_dimension : int -> mip_problem
val ppl_new_MIP_Problem :
  int ->
  constraint_system ->
  linear_expression ->
  optimization_mode -> mip_problem
val ppl_MIP_Problem_space_dimension : mip_problem -> int
val ppl_MIP_Problem_integer_space_dimensions : mip_problem -> int list
val ppl_MIP_Problem_constraints : mip_problem -> constraint_system
val ppl_MIP_Problem_add_space_dimensions_and_embed :
  mip_problem -> int -> unit
val ppl_MIP_Problem_add_to_integer_space_dimensions :
  mip_problem -> int list -> unit
val ppl_MIP_Problem_add_constraint : mip_problem -> linear_constraint -> unit
val ppl_MIP_Problem_add_constraints :
  mip_problem -> constraint_system -> unit
val ppl_MIP_Problem_set_objective_function :
  mip_problem -> linear_expression -> unit
val ppl_MIP_Problem_is_satisfiable : mip_problem -> bool
val ppl_MIP_Problem_solve : mip_problem -> mip_problem_status
val ppl_MIP_Problem_optimization_mode : mip_problem -> optimization_mode
val ppl_MIP_Problem_feasible_point : mip_problem -> linear_generator
val ppl_MIP_Problem_optimizing_point : mip_problem -> linear_generator
val ppl_MIP_Problem_objective_function : mip_problem -> linear_expression
val ppl_MIP_Problem_optimal_value : mip_problem -> Gmp.Z.t * Gmp.Z.t
val ppl_MIP_Problem_evaluate_objective_function :
  mip_problem ->
  linear_generator -> Gmp.Z.t * Gmp.Z.t
val ppl_MIP_Problem_OK : mip_problem -> bool
val ppl_MIP_Problem_clear : mip_problem -> unit
val ppl_MIP_Problem_set_optimization_mode :
  mip_problem -> optimization_mode -> unit
val ppl_MIP_Problem_set_control_parameter :
  mip_problem ->
  control_parameter_value -> unit
val ppl_MIP_Problem_get_control_parameter :
  mip_problem ->
  control_parameter_name ->
  control_parameter_value
val ppl_MIP_Problem_swap : mip_problem -> mip_problem -> unit
val ppl_MIP_Problem_ascii_dump : mip_problem -> string
type pip_problem
```

```
type pip_tree_node
type artificial_parameter = linear_expression * Gmp.Z.t
val ppl_new_PIP_Problem_from_space_dimension : int -> pip_problem
val ppl_new_PIP_Problem :
  int ->
  constraint_system ->
  int list -> pip_problem
val ppl_PIP_Problem_space_dimension : pip_problem -> int
val ppl_PIP_Problem_parameter_space_dimensions : pip_problem -> int list
val ppl_PIP_Problem_constraints : pip_problem -> constraint_system
val ppl_PIP_Problem_add_space_dimensions_and_embed :
  pip_problem -> int -> int -> unit
val ppl_PIP_Problem_add_to_parameter_space_dimensions :
  pip_problem -> int list -> unit
val ppl_PIP_Problem_add_constraint : pip_problem -> linear_constraint -> unit
val ppl_PIP_Problem_add_constraints :
  pip_problem -> constraint_system -> unit
val ppl_PIP_Problem_is_satisfiable : pip_problem -> bool
val ppl_PIP_Problem_solve : pip_problem -> pip_problem_status
val ppl_PIP_Problem_solution : pip_problem -> pip_tree_node
val ppl_PIP_Problem_optimizing_solution : pip_problem -> pip_tree_node
val ppl_PIP_Problem_get_big_parameter_dimension : pip_problem -> int
val ppl_PIP_Problem_set_big_parameter_dimension : pip_problem -> int -> unit
val ppl_PIP_Problem_has_big_parameter_dimension : pip_problem -> bool
val ppl_PIP_Problem_OK : pip_problem -> bool
val ppl_PIP_Problem_clear : pip_problem -> unit
val ppl_PIP_Problem_set_control_parameter :
  pip_problem ->
  pip_problem_control_parameter_value -> unit
val ppl_PIP_Problem_get_control_parameter :
  pip_problem ->
  pip_problem_control_parameter_name ->
  pip_problem_control_parameter_value
val ppl_PIP_Problem_swap : pip_problem -> pip_problem -> unit
val ppl_PIP_Problem_ascii_dump : pip_problem -> string
val ppl_PIP_Tree_Node_constraints : pip_tree_node -> constraint_system
val ppl_PIP_Tree_Node_artificials :
  pip_tree_node ->
  artificial_parameter list
val ppl_PIP_Tree_Node_ascii_dump : pip_tree_node -> string
val ppl_PIP_Tree_Node_OK : pip_tree_node -> bool
val ppl_PIP_Tree_Node_is_bottom : pip_tree_node -> bool
val ppl_PIP_Tree_Node_is_solution : pip_tree_node -> bool
val ppl_PIP_Tree_Node_parametric_values :
  pip_tree_node -> int -> linear_expression
val ppl_PIP_Tree_Node_is_decision : pip_tree_node -> bool
val ppl_PIP_Tree_Node_true_child : pip_tree_node -> pip_tree_node
val ppl_PIP_Tree_Node_false_child : pip_tree_node -> pip_tree_node
```

Compilation and Installation

When the Parma Polyhedra Library is configured, it tests for the existence of the OCaml system. If OCaml is correctly installed in a standard location, things are arranged so that the OCaml interface is built and installed.

3 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.

4 GNU Free Documentation License

Version 1.2, November 2002

```
Copyright (C) 2000,2001,2002  Free Software Foundation, Inc.
51 Franklin St, Fifth Floor, Boston, MA  02110-1301,  USA.
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.
```

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.

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 for under this License. Any other attempt to copy, modify, sublicense or distribute the Document 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.

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.

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 1.2
```

```
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.

5 Module Index

5.1 Modules

Here is a list of all modules:

6 Module Documentation

6.1 OCaml Language Interface

The Parma Polyhedra Library comes equipped with an interface for the OCaml language.

Index