The Parma Polyhedra Library Java Language Interface Developer's Manual* (version 0.11.2)

Roberto Bagnara†
Patricia M. Hill‡
Enea Zaffanella§

February 27, 2011

*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.
‡hill@comp.leeds.ac.uk, School of Computing, University of Leeds, U.K.
§zaffanella@cs.unipr.it, Department of Mathematics, University of Parma, Italy.

Contents

1 Main Page

The Parma Polyhedra Library comes equipped with an interface for the Java language. The Java 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 Java interface. In the sequel, `prefix` is the path prefix under which the library has been installed (typically `/usr` or `/usr/local`).

Overview

Here is a list of notes with general information and advice on the use of the Java interface.

- When the Parma Polyhedra Library is configured, it will automatically test for the existence of the Java system (unless configuration options are passed to disable the build of the Java interface; see configuration option `--enable-interfaces`). If Java is correctly installed in a standard location, things will be arranged so that the Java interface is built and installed (see configuration option `--with-java` if you need to specify a non-standard location for the Java system).

- The Java 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.

- Any application using the PPL should:
  - Load the PPL interface library by calling `System.load` and passing the full path of the dynamic shared object;
  - Make sure that only the intended version(s) of the library has been loaded, e.g., by calling static method `version()` in class `parma_polyhedra_library.Parma_Polyhedra_Library`;
  - Starting from version 0.11, initialize the interface by calling static method `initialize_library()`; when all library work is done, finalize the interface by calling `finalize_library()`.

- The numerical abstract domains available to the Java user as Java classes consist of the *simple* domains, *powersets* of a simple domain and *products* of simple domains. Note that the default configuration will only enable a subset of these domains (if you need a different set of domains, see configuration option `--enable-instantiations`).
  - 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 signed_char, short, 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, Float_Box, Double_Box, Long_Double_Box, Z_Box, Rational_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*.

- A Java program can create a new object for a PPL domain by using the constructors for the class corresponding to the domain.

- 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 methods 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.

2 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.

3 GNU Free Documentation License

Version 1.2, November 2002

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 worldwide, 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.

4 Module Index

4.1 Modules

Here is a list of all modules:

5 Namespace Index

5.1 Namespace List

Here is a list of all namespaces with brief descriptions:

6 Class Index

6.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

7 Class Index

7.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

8 File Index

8.1 File List

Here is a list of all files with brief descriptions:

9 Module Documentation

9.1 Java Language Interface

Classes

- class parma_polyhedra_library::Artificial_Parameter_Sequence
  A sequence of artificial parameters.
- class parma_polyhedra_library::By_Reference< T >
  An utility class implementing mutable and non-mutable call-by-reference.
- class parma_polyhedra_library::Coefficient
  A PPL coefficient.
- class parma_polyhedra_library::Congruence
  A linear congruence.
- class parma_polyhedra_library::Congruence_System
  A system of congruences.
- class parma_polyhedra_library::Constraint
  A linear equality or inequality.
- class parma_polyhedra_library::Constraint_System
  A system of constraints.
- class parma_polyhedra_library::Domain_Error_Exception
  Exceptions caused by domain errors.
- class parma_polyhedra_library::Polyhedron
  The Java base class for (C and NNC) convex polyhedra.
- class parma_polyhedra_library::C_Polyhedron
  A topologically closed convex polyhedron.
- class parma_polyhedra_library::Pointset_Powerset_C_Polyhedron
  A powerset of C_Polyhedron objects.
- class parma_polyhedra_library::Pointset_Powerset_C_Polyhedron_Iterator
  An iterator class for the disjuncts of a Pointset_Powerset_C_Polyhedron.
- class parma_polyhedra_library::Generator
  A line, ray, point or closure point.
- class parma_polyhedra_library::Generator_System
  A system of generators.
- class parma_polyhedra_library::Grid_Generator
  A grid line, parameter or grid point.

- class parma_polyhedra_library::Grid_Generator_System
  A system of grid generators.
- class parma_polyhedra_library::Invalid_Argument_Exception
  Exceptions caused by invalid arguments.
- class parma_polyhedra_library::IO
  A class collecting I/O functions.
- class parma_polyhedra_library::Length_Error_Exception
  Exceptions caused by too big length/size values.
- class parma_polyhedra_library::Linear_Expression
  A linear expression.
- class parma_polyhedra_library::Linear_Expression_Coefficient
  A linear expression built from a coefficient.
- class parma_polyhedra_library::Linear_Expression_Difference
  The difference of two linear expressions.
- class parma_polyhedra_library::Linear_Expression_Sum
  The sum of two linear expressions.
- class parma_polyhedra_library::Linear_Expression_Times
  The product of a linear expression and a coefficient.
- class parma_polyhedra_library::Linear_Expression_Unary_Minus
  The negation of a linear expression.
- class parma_polyhedra_library::Linear_Expression_Variable
  A linear expression built from a variable.
- class parma_polyhedra_library::Logic_Error_Exception
  Exceptions due to errors in low-level routines.
- class parma_polyhedra_library::MIP_Problem
  A Mixed Integer (linear) Programming problem.
- class parma_polyhedra_library::Overflow_Error_Exception
  Exceptions due to overflow errors.
- class parma_polyhedra_library::Pair< K, V >
  A pair of values of type K and V.
- class parma_polyhedra_library::Parma_Polyhedra_Library
  A class collecting library-level functions.
- class parma_polyhedra_library::Partial_Function

A partial function on space dimension indices.

- class parma_polyhedra_library::PIP_Problem

  A Parametric Integer Programming problem.

- class parma_polyhedra_library::Poly_Con_Relation

  The relation between a polyhedron and a constraint.

- class parma_polyhedra_library::PPL_Object

  Smart pointer to a PPL, C++ object.

- class parma_polyhedra_library::Timeout_Exception

  Exceptions caused by timeout expiring.

- class parma_polyhedra_library::Variable

  A dimension of the vector space.

Namespaces

- namespace parma_polyhedra_library

  The PPL Java interface package.

Enumerations

- enum parma_polyhedra_library::Bounded_Integer_Type_Overflow { parma_polyhedra_library::OVERFLOW_WRAPS, parma_polyhedra_library::OVERFLOW_UNDEFINED, parma_polyhedra_library::OVERFLOW_IMPOSSIBLE }

  Overflow behavior of bounded integer types.

- enum parma_polyhedra_library::Bounded_Integer_Type_Representation { parma_polyhedra_library::UNSIGNED, parma_polyhedra_library::SIGNED_2_COMPLEMENT }

  Representation of bounded integer types.

- enum parma_polyhedra_library::Bounded_Integer_Type_Width {

  parma_polyhedra_library::BITS_8, parma_polyhedra_library::BITS_16, parma_polyhedra_library::BITS_32, parma_polyhedra_library::BITS_64,

  parma_polyhedra_library::BITS_128 }

  Widths of bounded integer types.

- enum parma_polyhedra_library::Complexity_Class { parma_polyhedra_library::POLYNOMIAL_COMPLEXITY, parma_polyhedra_library::SIMPLEX_COMPLEXITY, parma_polyhedra_library::ANY_COMPLEXITY }

  Possible Complexities.

- enum parma_polyhedra_library::Control_Parameter_Name { parma_polyhedra_library::PRICING }

  Names of MIP problems' control parameters.

- enum parma_polyhedra_library::Control_Parameter_Value { parma_polyhedra_library::PRICING_STEEPEST_EDGE_FLOAT, parma_polyhedra_library::PRICING_STEEPEST_EDGE_EXACT, parma_polyhedra_library::PRICING_TEXTBOOK }

  Possible values for MIP problem's control parameters.

- enum parma_polyhedra_library::Degenerate_Element { parma_polyhedra_library::UNIVERSE, parma_polyhedra_library::EMPTY }

  Kinds of degenerate abstract elements.

- enum parma_polyhedra_library::Generator_Type { parma_polyhedra_library::LINE, parma_polyhedra_library::RAY, parma_polyhedra_library::POINT, parma_polyhedra_library::CLOSURE_POINT }

  The generator type.

- enum parma_polyhedra_library::Grid_Generator_Type { parma_polyhedra_library::LINE, parma_polyhedra_library::PARAMETER, parma_polyhedra_library::POINT }

  The grid generator type.

- enum parma_polyhedra_library::MIP_Problem_Status { parma_polyhedra_library::UNFEASIBLE_MIP_PROBLEM, parma_polyhedra_library::UNBOUNDED_MIP_PROBLEM, parma_polyhedra_library::OPTIMIZED_MIP_PROBLEM }

  Possible outcomes of the MIP_Problem solver.

- enum parma_polyhedra_library::Optimization_Mode { parma_polyhedra_library::MINIMIZATION, parma_polyhedra_library::MAXIMIZATION }

  Possible optimization modes.

- enum parma_polyhedra_library::PIP_Problem_Control_Parameter_Name { parma_polyhedra_library::CUTTING_STRATEGY, parma_polyhedra_library::PIVOT_ROW_STRATEGY }

  Names of PIP problems' control parameters.

- enum parma_polyhedra_library::PIP_Problem_Control_Parameter_Value {

  parma_polyhedra_library::CUTTING_STRATEGY_FIRST, parma_polyhedra_library::CUTTING_STRATEGY_DEEPEST, parma_polyhedra_library::CUTTING_STRATEGY_ALL, parma_polyhedra_library::PIVOT_ROW_STRATEGY_FIRST,

  parma_polyhedra_library::PIVOT_ROW_STRATEGY_MAX_COLUMN }

  Possible values for PIP problems' control parameters.

- enum parma_polyhedra_library::PIP_Problem_Status { parma_polyhedra_library::UNFEASIBLE_PIP_PROBLEM, parma_polyhedra_library::OPTIMIZED_PIP_PROBLEM }

  Possible outcomes of the PIP_Problem solver.

- enum parma_polyhedra_library::Relation_Symbol {

  parma_polyhedra_library::LESS_THAN, parma_polyhedra_library::LESS_OR_EQUAL, parma_polyhedra_library::EQUAL, parma_polyhedra_library::GREATER_OR_EQUAL,

  parma_polyhedra_library::GREATER_THAN }

  Relation symbols.

9.1.1 Detailed Description

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

9.1.2 Enumeration Type Documentation

9.1.2.1 enum parma_polyhedra_library::Bounded_Integer_Type_Overflow

Overflow behavior of bounded integer types.

Enumerator:

OVERFLOW_WRAPS On overflow, wrapping takes place.

OVERFLOW_UNDEFINED On overflow, the result is undefined.

OVERFLOW_IMPOSSIBLE Overflow is impossible.

Definition at line 28 of file Bounded_Integer_Type_Overflow.java.

9.1.2.2 enum parma_polyhedra_library::Bounded_Integer_Type_Representation

Representation of bounded integer types.

Enumerator:

UNSIGNED Unsigned binary.

SIGNED_2_COMPLEMENT Signed binary where negative values are represented by the two's complement of the absolute value.

Definition at line 28 of file Bounded_Integer_Type_Representation.java.

9.1.2.3 enum parma_polyhedra_library::Bounded_Integer_Type_Width

Widths of bounded integer types.

Enumerator:

BITS_8 Minimization is requested.

BITS_16 16 bits.

BITS_32 32 bits.

BITS_64 64 bits.

BITS_128 128 bits.

Definition at line 28 of file Bounded_Integer_Type_Width.java.

9.1.2.4 enum parma_polyhedra_library::Complexity_Class

Possible Complexities.

Enumerator:

POLYNOMIAL_COMPLEXITY Worst-case polynomial complexity.

SIMPLEX_COMPLEXITY Worst-case exponential complexity but typically polynomial behavior.

ANY_COMPLEXITY Any complexity.

Definition at line 28 of file Complexity_Class.java.

9.1.2.5 enum parma_polyhedra_library::Control_Parameter_Name

Names of MIP problems' control parameters.

Enumerator:

PRICING The pricing rule.

Definition at line 28 of file Control_Parameter_Name.java.

9.1.2.6 enum parma_polyhedra_library::Control_Parameter_Value

Possible values for MIP problem's control parameters.

Enumerator:

PRICING_STEEPEST_EDGE_FLOAT Steepest edge pricing method, using floating points (default).

PRICING_STEEPEST_EDGE_EXACT Steepest edge pricing method, using Coefficient.

PRICING_TEXTBOOK Textbook pricing method.

Definition at line 28 of file Control_Parameter_Value.java.

9.1.2.7 enum parma_polyhedra_library::Degenerate_Element

Kinds of degenerate abstract elements.

Enumerator:

UNIVERSE The universe element, i.e., the whole vector space.

EMPTY The empty element, i.e., the empty set.

Definition at line 28 of file Degenerate_Element.java.

9.1.2.8 enum parma_polyhedra_library::Generator_Type

The generator type.

Enumerator:

LINE The generator is a line.
RAY The generator is a ray.
POINT The generator is a point.
CLOSURE_POINT The generator is a closure point.

Definition at line 28 of file Generator_Type.java.

9.1.2.9 enum parma_polyhedra_library::Grid_Generator_Type

The grid generator type.

Enumerator:

LINE The generator is a line.
PARAMETER The generator is a parameter.
POINT The generator is a point.

Definition at line 28 of file Grid_Generator_Type.java.

9.1.2.10 enum parma_polyhedra_library::MIP_Problem_Status

Possible outcomes of the MIP_Problem solver.

Enumerator:

UNFEASIBLE_MIP_PROBLEM The problem is unfeasible.
UNBOUNDED_MIP_PROBLEM The problem is unbounded.
OPTIMIZED_MIP_PROBLEM The problem has an optimal solution.

Definition at line 28 of file MIP_Problem_Status.java.

9.1.2.11 enum parma_polyhedra_library::Optimization_Mode

Possible optimization modes.

Enumerator:

MINIMIZATION Minimization is requested.
MAXIMIZATION Maximization is requested.

Definition at line 28 of file Optimization_Mode.java.

9.1.2.12 enum parma_polyhedra_library::PIP_Problem_Control_Parameter_Name

Names of PIP problems' control parameters.

Enumerator:

CUTTING_STRATEGY The cutting strategy rule.

PIVOT_ROW_STRATEGY The pivot row strategy rule.

Definition at line 28 of file PIP_Problem_Control_Parameter_Name.java.

9.1.2.13 enum parma_polyhedra_library::PIP_Problem_Control_Parameter_Value

Possible values for PIP problems' control parameters.

Enumerator:

CUTTING_STRATEGY_FIRST Choose the first non-integer row.

CUTTING_STRATEGY_DEEPEST Choose row which generates the deepest cut.

CUTTING_STRATEGY_ALL Always generate all possible cuts.

PIVOT_ROW_STRATEGY_FIRST Choose the first row with negative parameter sign.

PIVOT_ROW_STRATEGY_MAX_COLUMN Choose the row which generates the lexico-maximal pivot column.

Definition at line 28 of file PIP_Problem_Control_Parameter_Value.java.

9.1.2.14 enum parma_polyhedra_library::PIP_Problem_Status

Possible outcomes of the PIP_Problem solver.

Enumerator:

UNFEASIBLE_PIP_PROBLEM The problem is unsatisfiable.

OPTIMIZED_PIP_PROBLEM The problem has an optimal solution.

Definition at line 28 of file PIP_Problem_Status.java.

9.1.2.15 enum parma_polyhedra_library::Relation_Symbol

Relation symbols.

Enumerator:

LESS_THAN Less than.

LESS_OR_EQUAL Less than or equal to.

EQUAL Equal to.

GREATER_OR_EQUAL Greater than or equal to.

GREATER_THAN Greater than.

Definition at line 28 of file Relation_Symbol.java.

10 Namespace Documentation

10.1 Parma_Polyhedra_Library Namespace Reference

Namespaces

- namespace Interfaces

10.2 parma_polyhedra_library Namespace Reference

The PPL Java interface package.

Classes

- class Artificial_Parameter
- class Artificial_Parameter_Sequence

  A sequence of artificial parameters.

- class By_Reference< T >

  An utility class implementing mutable and non-mutable call-by-reference.

- class Coefficient

  A PPL coefficient.

- class Congruence

  A linear congruence.

- class Congruence_System

  A system of congruences.

- class Constraint

  A linear equality or inequality.

- class Constraint_System

  A system of constraints.

- class Domain_Error_Exception

  Exceptions caused by domain errors.

- class Fake_Class_For_Doxygen

- class Polyhedron

  The Java base class for (C and NNC) convex polyhedra.

- class C_Polyhedron

  A topologically closed convex polyhedron.

- class Pointset_Powerset_C_Polyhedron

  A powerset of C_Polyhedron objects.

- class Pointset_Powerset_C_Polyhedron_Iterator

  An iterator class for the disjuncts of a Pointset_Powerset_C_Polyhedron.

- class Generator

  A line, ray, point or closure point.

- class Generator_System

  A system of generators.

- class Grid_Generator

  A grid line, parameter or grid point.

- class Grid_Generator_System

  A system of grid generators.

- class Invalid_Argument_Exception

  Exceptions caused by invalid arguments.

- class IO

  A class collecting I/O functions.

- class Length_Error_Exception

  Exceptions caused by too big length/size values.

- class Linear_Expression

  A linear expression.

- class Linear_Expression_Coefficient

  A linear expression built from a coefficient.

- class Linear_Expression_Difference

  The difference of two linear expressions.

- class Linear_Expression_Sum

  The sum of two linear expressions.

- class Linear_Expression_Times

  The product of a linear expression and a coefficient.

- class Linear_Expression_Unary_Minus

  The negation of a linear expression.

Enumerations

- enum Bounded_Integer_Type_Overflow { OVERFLOW_WRAPS, OVERFLOW_UNDEFINED, OVERFLOW_IMPOSSIBLE }

  Overflow behavior of bounded integer types.

- enum Bounded_Integer_Type_Representation { UNSIGNED, SIGNED_2_COMPLEMENT }

  Representation of bounded integer types.

- enum Bounded_Integer_Type_Width {
  BITS_8, BITS_16, BITS_32, BITS_64,
  BITS_128 }

  Widths of bounded integer types.

- enum Complexity_Class { POLYNOMIAL_COMPLEXITY, SIMPLEX_COMPLEXITY, ANY_-COMPLEXITY }

  Possible Complexities.

- enum Control_Parameter_Name { PRICING }

  Names of MIP problems' control parameters.

- enum Control_Parameter_Value { PRICING_STEEPEST_EDGE_FLOAT, PRICING_-STEEPEST_EDGE_EXACT, PRICING_TEXTBOOK }

  Possible values for MIP problem's control parameters.

- enum Degenerate_Element { UNIVERSE, EMPTY }

  Kinds of degenerate abstract elements.

- enum Generator_Type { LINE, RAY, POINT, CLOSURE_POINT }

  The generator type.

- enum Grid_Generator_Type { LINE, PARAMETER, POINT }

  The grid generator type.

- enum MIP_Problem_Status { UNFEASIBLE_MIP_PROBLEM, UNBOUNDED_MIP_PROBLEM, OPTIMIZED_MIP_PROBLEM }

  Possible outcomes of the MIP_Problem solver.

- enum Optimization_Mode { MINIMIZATION, MAXIMIZATION }

  Possible optimization modes.

- enum PIP_Problem_Control_Parameter_Name { CUTTING_STRATEGY, PIVOT_ROW_-STRATEGY }

  Names of PIP problems' control parameters.

- enum PIP_Problem_Control_Parameter_Value {
  CUTTING_STRATEGY_FIRST, CUTTING_STRATEGY_DEEPEST, CUTTING_STRATEGY_-ALL, PIVOT_ROW_STRATEGY_FIRST,
  PIVOT_ROW_STRATEGY_MAX_COLUMN }

Possible values for PIP problems' control parameters.

- enum PIP_Problem_Status { UNFEASIBLE_PIP_PROBLEM, OPTIMIZED_PIP_PROBLEM }

  Possible outcomes of the PIP_Problem solver.

- enum Relation_Symbol {
  LESS_THAN, LESS_OR_EQUAL, EQUAL, GREATER_OR_EQUAL,
  GREATER_THAN }

  Relation symbols.

10.2.1 Detailed Description

The PPL Java interface package. All classes, interfaces and enums related to the Parma Polyhedra Library Java interface are included in this package.

10.3 Parma_Polyhedra_Library::Interfaces Namespace Reference

Namespaces

- namespace Java

10.4 Parma_Polyhedra_Library::Interfaces::Java Namespace Reference

Classes

- struct Java_ExceptionOccurred
- class timeout_exception
- class deterministic_timeout_exception
- class Java_Class_Cache

  A cache for global references to Java classes.

- struct Java_FMID_Cache

  A cache for field and method IDs of Java classes.

Functions

- void handle_exception (JNIEnv *env, const std::overflow_error &e)
- void handle_exception (JNIEnv *env, const std::invalid_argument &e)
- void handle_exception (JNIEnv *env, const std::logic_error &e)
- void handle_exception (JNIEnv *env, const std::length_error &e)
- void handle_exception (JNIEnv *env, const std::domain_error &e)
- void handle_exception (JNIEnv *env, const std::bad_alloc &)
- void handle_exception (JNIEnv *env, const std::exception &e)
- void handle_exception (JNIEnv *env, const timeout_exception &)
- void handle_exception (JNIEnv *env, const deterministic_timeout_exception &)
- void handle_exception (JNIEnv *env)

- void reset_timeout ()
- void reset_deterministic_timeout ()
- jobject build_java_poly_gen_relation (JNIEnv *env, Poly_Gen_Relation &r)

  Builds a Java parma_polyhedra_library::Poly_Gen_Relation from C++ Poly_Gen_Relation `r`.

- jobject build_java_poly_con_relation (JNIEnv *env, Poly_Con_Relation &r)

  Builds a Java parma_polyhedra_library::Poly_Con_Relation from C++ Poly_Con_Relation `r`.

- Congruence build_cxx_congruence (JNIEnv *env, jobject j_cg)

  Builds a C++ Congruence from Java parma_polyhedra_library::Congruence `j_cg`.

- PIP_Tree_Node::Artificial_Parameter build_cxx_artificial_parameter (JNIEnv *env, jobject j_ap)

  Builds a C++ Artificial_Parameter from Java parma_polyhedra_library::Artificial_Parameter `j_artificial_parameter`.

- jobject bool_to_j_boolean_class (JNIEnv *env, const bool value)

  Builds a Java Boolean from C++ bool `value`.

- jobject j_long_to_j_long_class (JNIEnv *env, jlong value)

  Builds a Java Long from Java long `value`.

- jlong j_long_class_to_j_long (JNIEnv *env, jobject j_long)

  Returns the Java long stored in Java Long `j_long`.

- jobject j_int_to_j_integer (JNIEnv *env, jint value)

  Builds a Java Integer from Java int `value`.

- jint j_integer_to_j_int (JNIEnv *env, jobject j_integer)

  Returns the Java int stored in Java Integer `j_integer`.

- Variables_Set build_cxx_variables_set (JNIEnv *env, jobject v_set)

  Builds a C++ Variables_Set from Java parma_polyhedra_library::Variables_Set `v_set`.

- jobject build_java_variables_set (JNIEnv *env, const Variables_Set &v_set)

  Builds a Java parma_polyhedra_library::Variables_Set from C++ Variables_Set `v_set`.

- Bounded_Integer_Type_Overflow build_cxx_bounded_overflow (JNIEnv *env, jobject j_bounded_overflow)

  Builds a C++ Bounded_Integer_Type_Overflow from Java parma_polyhedra_library::Bounded_Integer_Type_Overflow `j_bounded_overflow`.

- Bounded_Integer_Type_Representation build_cxx_bounded_rep (JNIEnv *env, jobject j_bounded_rep)

  Builds a C++ Bounded_Integer_Type_Representation from Java parma_polyhedra_library::Bounded_Integer_Type_Representation `j_bounded_rep`.

- Bounded_Integer_Type_Width build_cxx_bounded_width (JNIEnv *env, jobject j_bounded_width)

  Builds a C++ Bounded_Integer_Type_Width from Java parma_polyhedra_library::Bounded_Integer_Type_Width `j_bounded_width`.

- Relation_Symbol build_cxx_relsym (JNIEnv *env, jobject j_relsym)

  Builds a C++ Relation_Symbol from Java parma_polyhedra_library::Relation_Symbol `j_relsym`.

- Optimization_Mode build_cxx_optimization_mode (JNIEnv *env, jobject j_opt_mode)

  Builds a C++ Optimization_Mode from Java parma_polyhedra_library::Optimization_Mode `j_opt_mode`.

- jobject build_java_mip_status (JNIEnv *env, const MIP_Problem_Status &mip_status)

  Builds a Java parma_polyhedra_library::MIP_Problem_Status from C++ MIP_Problem_Status `mip_status`.

- jobject build_java_pip_status (JNIEnv *env, const PIP_Problem_Status &pip_status)

  Builds a Java parma_polyhedra_library::PIP_Problem_Status from C++ PIP_Problem_Status `pip_status`.

- jobject build_java_optimization_mode (JNIEnv *env, const Optimization_Mode &opt_mode)

  Builds a Java parma_polyhedra_library::Optimization_Mode from C++ Optimization_Mode `opt_mode`.

- MIP_Problem::Control_Parameter_Name build_cxx_control_parameter_name (JNIEnv *env, jobject j_cp_name)

  Builds a C++ MIP_Problem::Control_Parameter_Name from Java parma_polyhedra_library::Control_Parameter_Name `j_cp_name`.

- jobject build_java_control_parameter_name (JNIEnv *env, const MIP_Problem::Control_Parameter_Name &cp_name)

  Builds a Java parma_polyhedra_library::Control_Parameter_Name from C++ MIP_Problem::Control_Parameter_Name `cp_name`.

- MIP_Problem::Control_Parameter_Value build_cxx_control_parameter_value (JNIEnv *env, jobject j_cp_value)

  Builds a C++ MIP_Problem::Control_Parameter_Value from Java parma_polyhedra_library::Control_Parameter_Value `j_cp_value`.

- jobject build_java_control_parameter_value (JNIEnv *env, const MIP_Problem::Control_Parameter_Value &cp_value)

  Builds a Java parma_polyhedra_library::Control_Parameter_Value from C++ MIP_Problem::Control_Parameter_Value `cp_value`.

- PIP_Problem::Control_Parameter_Name build_cxx_pip_problem_control_parameter_name (JNIEnv *env, jobject j_cp_name)

  Builds a C++ PIP_Problem::Control_Parameter_Name from Java parma_polyhedra_library::PIP_Problem_Control_Parameter_Name `j_cp_name`.

- jobject build_pip_problem_java_control_parameter_name (JNIEnv *env, const PIP_Problem::Control_Parameter_Name &cp_name)
- PIP_Problem::Control_Parameter_Value build_cxx_pip_problem_control_parameter_value (JNIEnv *env, jobject j_cp_value)

  Builds a C++ PIP_Problem::Control_Parameter_Value from Java parma_polyhedra_library::PIP_Problem_Control_Parameter_Value `j_cp_value`.

- jobject build_java_pip_problem_control_parameter_value (JNIEnv *env, const PIP_Problem::Control_Parameter_Value &cp_value)

Builds a Java parma_polyhedra_library::Control_Parameter_Value from C++ PIP_Problem::Control_Parameter_Value `cp_value`.

- Constraint build_cxx_constraint (JNIEnv *env, jobject j_constraint)

  Builds a C++ Constraint from Java parma_polyhedra_library::Constraint `j_constraint`.

- Linear_Expression build_cxx_linear_expression (JNIEnv *env, jobject j_le)

  Builds a C++ Linear_Expression from Java parma_polyhedra_library::Linear_Expression `j_le`.

- Generator build_cxx_generator (JNIEnv *env, jobject j_g)

  Builds a C++ Generator from Java parma_polyhedra_library::Generator `j_g`.

- Grid_Generator build_cxx_grid_generator (JNIEnv *env, jobject j_g)

  Builds a C++ Grid_Generator from Java parma_polyhedra_library::Grid_Generator `j_g`.

- jobject build_java_linear_expression_coefficient (JNIEnv *env, const Coefficient &coeff)

  Builds a Java parma_polyhedra_library::Linear_Expression_Coefficient from C++ Coefficient `coeff`.

- void set_generator (JNIEnv *env, jobject dst, jobject src)

  Sets Java parma_polyhedra_library::Generator `dst` *to have the same value as* `src`.

- void set_pair_element (JNIEnv *env, jobject dst_pair, int arg, jobject src)

  Assigns `src` *to one of the fields of parma_polyhedra_library::Pair object* `dst_pair`.

- jobject get_pair_element (JNIEnv *env, int arg, jobject pair)

  Returns one of the fields of the parma_polyhedra_library::Pair object `pair`.

- jobject build_java_constraint (JNIEnv *env, const Constraint &c)

  Builds a Java parma_polyhedra_library::Constraint from C++ Constraint `c`.

- jobject build_java_congruence (JNIEnv *env, const Congruence &cg)

  Builds a Java parma_polyhedra_library::Congruence from C++ Congruence `cg`.

- jobject build_java_generator (JNIEnv *env, const Generator &g)

  Builds a Java parma_polyhedra_library::Generator from C++ Generator `g`.

- jobject build_java_grid_generator (JNIEnv *env, const Grid_Generator &g)

  Builds a Java parma_polyhedra_library::Grid_Generator from C++ Grid_Generator `g`.

- jobject build_java_constraint_system (JNIEnv *env, const Constraint_System &cs)

  Builds a Java parma_polyhedra_library::Constraint_System from C++ Constraint_System `cs`.

- jobject build_java_generator_system (JNIEnv *env, const Generator_System &gs)

  Builds a Java parma_polyhedra_library::Generator_System from C++ Generator_System `gs`.

- jobject build_java_grid_generator_system (JNIEnv *env, const Grid_Generator_System &gs)

  Builds a Java parma_polyhedra_library::Grid_Generator_System from C++ Grid_Generator_System `gs`.

- jobject build_java_congruence_system (JNIEnv *env, const Congruence_System &cgs)

  Builds a Java parma_polyhedra_library::Congruence_System from C++ Congruence_System `cgs`.

- jobject build_java_artificial_parameter (JNIEnv *env, const PIP_Tree_Node::Artificial_Parameter &ap)

  Builds a Java parma_polyhedra_library::Artificial_Parameter from C++ Artificial_Parameter `ap`.

- template<typename U , typename V >
  U jtype_to_unsigned (const V &value)

  Builds an unsigned C++ number from the Java native number `value`.

- bool is_java_marked (JNIEnv *env, jobject ppl_object)

  Returns `true` *if and only if the Java object* `ppl_object` *refers to a C++ object.*

- jobject build_java_pip_problem_control_parameter_name (JNIEnv *env, const PIP_Problem::Control_Parameter_Name &cp_name)

  Builds a Java parma_polyhedra_library::PIP_Problem_Control_Parameter_Name from C++ PIP_Problem::Control_Parameter_Name `cp_name`.

- Variable build_cxx_variable (JNIEnv *env, jobject j_var)

  Builds a C++ Variable from Java parma_polyhedra_library::Variable `j_var`.

- jobject build_java_variable (JNIEnv *env, const Variable &var)

  Builds a Java parma_polyhedra_library::Variable from C++ Variable `var`.

- Coefficient build_cxx_coeff (JNIEnv *env, jobject j_coeff)

  Builds a C++ Coefficient from Java parma_polyhedra_library::Coefficient `j_coeff`.

- jobject build_java_coeff (JNIEnv *env, const Coefficient &ppl_coeff)

  Builds a Java parma_polyhedra_library::Coefficient from C++ Coefficient `ppl_coeff`.

- Grid_Generator_System build_cxx_grid_generator_system (JNIEnv *env, jobject j_gs)

  Builds a C++ Grid_Generator_System from Java parma_polyhedra_library::Grid_Generator_System `j_gs`.

- Constraint_System build_cxx_constraint_system (JNIEnv *env, jobject j_cs)

  Builds a C++ Constraint_System from Java parma_polyhedra_library::Constraint_System `j_cs`.

- PIP_Tree_Node::Artificial_Parameter_Sequence build_cxx_artificial_parameter_sequence (JNIEnv *env, jobject j_aps)

  Builds a C++ Artificial_Parameter_Sequence from Java parma_polyhedra_library::Artificial_Parameter_Sequence `j_aps`.

- Generator_System build_cxx_generator_system (JNIEnv *env, jobject j_gs)

  Builds a C++ Generator_System from Java parma_polyhedra_library::Generator_System `j_gs`.

- Congruence_System build_cxx_congruence_system (JNIEnv *env, jobject j_cgs)

  Builds a C++ Congruence_System from Java parma_polyhedra_library::Congruence_System `j_cgs`.

- jobject build_java_artificial_parameter_sequence (JNIEnv *env, const PIP_Tree_Node::Artificial_Parameter_Sequence &aps)

  Builds a Java parma_polyhedra_library::Artificial_Parameter_Sequence from C++ Artificial_Parameter_Sequence `aps`.

- void set_coefficient (JNIEnv *env, jobject dst, jobject src)

  Sets Java Coefficient `dst` *to have the same value as* `src`.

- void set_by_reference (JNIEnv *env, jobject by_ref_dst, jobject src)

  Modifies parma_polyhedra_library::By_Reference object `by_ref_dst` *so that it references object* `src`.

- jobject get_by_reference (JNIEnv *env, jobject by_reference)

  Returns the object referenced by parma_polyhedra_library::By_Reference object `by_reference`.

- void * get_ptr (JNIEnv *env, jobject ppl_object)

  Returns a pointer to the C++ object wrapped by `ppl_object`.

- template<typename T >
  void set_ptr (JNIEnv *env, jobject ppl_object, const T *address, bool to_be_marked=false)

  Sets the pointer of the underlying C++ object in the Java object.

- template<typename R >
  jobject build_linear_expression (JNIEnv *env, const R &r)

  Builds a Java parma_polyhedra_library::Linear_Expression from the C++ constraint/congruence `r`.

- template<typename System , typename Elem_Builder >
  System build_cxx_system (JNIEnv *env, jobject j_iterable, Elem_Builder build_cxx_elem)

Variables

- Java_Class_Cache cached_classes

  The cached class references.

- Java_FMID_Cache cached_FMIDs

  The field and method ID cache.

10.4.1 Function Documentation

10.4.1.1 jobject Parma_Polyhedra_Library::Interfaces::Java::bool_to_j_boolean_class (JNIEnv * *env*, const bool *value*)

Builds a Java Boolean from C++ bool `value`.

10.4.1.2 PIP_Tree_Node::Artificial_Parameter Parma_Polyhedra_Library::Interfaces::Java::build_cxx_artificial_parameter (JNIEnv * *env*, jobject *j_artificial_parameter*)

Builds a C++ Artificial_Parameter from Java parma_polyhedra_library::Artificial_Parameter `j_artificial_parameter`.

Builds a C++ Artificial_Parameter from Java parma_polyhedra_library::Artificial_Parameter `j_art_-param`.

Referenced by Java_parma_1polyhedra_1library_Artificial_1Parameter_ascii_1dump(), and Java_parma_-1polyhedra_1library_Artificial_1Parameter_toString().

10.4.1.3 PIP_Tree_Node::Artificial_Parameter_Sequence Parma_Polyhedra_-Library::Interfaces::Java::build_cxx_artificial_parameter_sequence (JNIEnv * *env*, jobject *j_aps*)

Builds a C++ Artificial_Parameter_Sequence from Java parma_polyhedra_library::Artificial_Parameter_-Sequence `j_aps`.

10.4.1.4 Bounded_Integer_Type_Overflow Parma_Polyhedra_Library::Interfaces::Java::build_-cxx_bounded_overflow (JNIEnv * *env*, jobject *j_bounded_overflow*)

Builds a C++ Bounded_Integer_Type_Overflow from Java parma_polyhedra_library::Bounded_Integer_-Type_Overflow `j_bounded_overflow`.

10.4.1.5 Bounded_Integer_Type_Representation Parma_Polyhedra_-Library::Interfaces::Java::build_cxx_bounded_rep (JNIEnv * *env*, jobject *j_bounded_rep*)

Builds a C++ Bounded_Integer_Type_Representation from Java parma_polyhedra_library::Bounded_-Integer_Type_Representation `j_bounded_rep`.

10.4.1.6 Bounded_Integer_Type_Width Parma_Polyhedra_Library::Interfaces::Java::build_-cxx_bounded_width (JNIEnv * *env*, jobject *j_bounded_width*)

Builds a C++ Bounded_Integer_Type_Width from Java parma_polyhedra_library::Bounded_Integer_-Type_Width `j_bounded_width`.

10.4.1.7 Coefficient Parma_Polyhedra_Library::Interfaces::Java::build_cxx_coeff (JNIEnv * *env*, jobject *j_coeff*) `[inline]`

Builds a C++ Coefficient from Java parma_polyhedra_library::Coefficient `j_coeff`.

Definition at line 163 of file ppl_java_common.inlines.hh.

References cached_FMIDs, CHECK_EXCEPTION_THROW, CHECK_RESULT_THROW, and Parma_-Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Coefficient_toString_ID.

10.4.1.8 Congruence Parma_Polyhedra_Library::Interfaces::Java::build_cxx_congruence (JNIEnv * *env*, jobject *j_congruence*)

Builds a C++ Congruence from Java parma_polyhedra_library::Congruence `j_cg`.

Referenced by build_cxx_congruence_system(), Java_parma_1polyhedra_1library_Congruence_ascii_-1dump(), and Java_parma_1polyhedra_1library_Congruence_toString().

10.4.1.9 Congruence_System Parma_Polyhedra_Library::Interfaces::Java::build_cxx_-congruence_system (JNIEnv * *env*, jobject *j_cgs*) [inline]

Builds a C++ Congruence_System from Java parma_polyhedra_library::Congruence_System `j_cgs`.

Definition at line 216 of file ppl_java_common.inlines.hh.

References build_cxx_congruence().

Referenced by Java_parma_1polyhedra_1library_Congruence_1System_ascii_1dump(), and Java_-parma_1polyhedra_1library_Congruence_1System_toString().

10.4.1.10 Constraint Parma_Polyhedra_Library::Interfaces::Java::build_cxx_constraint (JNIEnv * *env*, jobject *j_constraint*)

Builds a C++ Constraint from Java parma_polyhedra_library::Constraint `j_constraint`.

Referenced by build_cxx_constraint_system(), Java_parma_1polyhedra_1library_Constraint_ascii_-1dump(), Java_parma_1polyhedra_1library_Constraint_toString(), Java_parma_1polyhedra_1library_-MIP_1Problem_add_1constraint(), and Java_parma_1polyhedra_1library_PIP_1Problem_add_-1constraint().

10.4.1.11 Constraint_System Parma_Polyhedra_Library::Interfaces::Java::build_cxx_-constraint_system (JNIEnv * *env*, jobject *j_cs*) [inline]

Builds a C++ Constraint_System from Java parma_polyhedra_library::Constraint_System `j_cs`.

Definition at line 222 of file ppl_java_common.inlines.hh.

References build_cxx_constraint().

Referenced by Java_parma_1polyhedra_1library_Constraint_1System_ascii_1dump(), Java_-parma_1polyhedra_1library_Constraint_1System_toString(), Java_parma_1polyhedra_1library_-MIP_1Problem_add_1constraints(), Java_parma_1polyhedra_1library_MIP_1Problem_build_1cpp_-1object__JLparma_1polyhedra_1library_Constraint_1System_2Lparma_1polyhedra_1library_Linear_-1Expression_2Lparma_1polyhedra_1library_Optimization_1Mode_2(), Java_parma_1polyhedra_-1library_PIP_1Problem_add_1constraints(), and Java_parma_1polyhedra_1library_PIP_1Problem_-build_1cpp_1object__JLparma_1polyhedra_1library_Constraint_1System_2Lparma_1polyhedra_-1library_Variables_1Set_2().

10.4.1.12 MIP_Problem::Control_Parameter_Name Parma_Polyhedra_Library::Interfaces::Java::build_cxx_control_parameter_name (JNIEnv * *env*, jobject *j_cp_name*)

Builds a C++ MIP_Problem::Control_Parameter_Name from Java parma_polyhedra_library::Control_Parameter_Name `j_cp_name`.

Referenced by Java_parma_1polyhedra_1library_MIP_1Problem_get_1control_1parameter().

10.4.1.13 MIP_Problem::Control_Parameter_Value Parma_Polyhedra_Library::Interfaces::Java::build_cxx_control_parameter_value (JNIEnv * *env*, jobject *j_cp_value*)

Builds a C++ MIP_Problem::Control_Parameter_Value from Java parma_polyhedra_library::Control_Parameter_Value `j_cp_value`.

Referenced by Java_parma_1polyhedra_1library_MIP_1Problem_set_1control_1parameter().

10.4.1.14 Generator Parma_Polyhedra_Library::Interfaces::Java::build_cxx_generator (JNIEnv * *env*, jobject *j_generator*)

Builds a C++ Generator from Java parma_polyhedra_library::Generator `j_g`.

Referenced by build_cxx_generator_system(), Java_parma_1polyhedra_1library_Generator_ascii_1dump(), Java_parma_1polyhedra_1library_Generator_toString(), and Java_parma_1polyhedra_1library_MIP_1Problem_evaluate_1objective_1function().

10.4.1.15 Generator_System Parma_Polyhedra_Library::Interfaces::Java::build_cxx_generator_system (JNIEnv * *env*, jobject *j_gs*) `[inline]`

Builds a C++ Generator_System from Java parma_polyhedra_library::Generator_System `j_gs`.

Definition at line 228 of file ppl_java_common.inlines.hh.

References build_cxx_generator().

Referenced by Java_parma_1polyhedra_1library_Generator_1System_ascii_1dump(), and Java_parma_1polyhedra_1library_Generator_1System_toString().

10.4.1.16 Grid_Generator Parma_Polyhedra_Library::Interfaces::Java::build_cxx_grid_generator (JNIEnv * *env*, jobject *j_grid_generator*)

Builds a C++ Grid_Generator from Java parma_polyhedra_library::Grid_Generator `j_g`.

Referenced by build_cxx_grid_generator_system(), Java_parma_1polyhedra_1library_Grid_1Generator_ascii_1dump(), and Java_parma_1polyhedra_1library_Grid_1Generator_toString().

10.4.1.17 Grid_Generator_System Parma_Polyhedra_Library::Interfaces::Java::build_cxx_grid_generator_system (JNIEnv * *env*, jobject *j_gs*) [inline]

Builds a C++ Grid_Generator_System from Java parma_polyhedra_library::Grid_Generator_System `j_gs`.

Definition at line 234 of file ppl_java_common.inlines.hh.

References build_cxx_grid_generator().

Referenced by Java_parma_1polyhedra_1library_Grid_1Generator_1System_ascii_1dump(), and Java_parma_1polyhedra_1library_Grid_1Generator_1System_toString().

10.4.1.18 Linear_Expression Parma_Polyhedra_Library::Interfaces::Java::build_cxx_linear_expression (JNIEnv * *env*, jobject *j_le*)

Builds a C++ Linear_Expression from Java parma_polyhedra_library::Linear_Expression `j_le`.

Referenced by Java_parma_1polyhedra_1library_Linear_1Expression_all_1homogeneous_1terms_1are_1zero(), Java_parma_1polyhedra_1library_Linear_1Expression_ascii_1dump(), Java_parma_1polyhedra_1library_Linear_1Expression_is_1zero(), Java_parma_1polyhedra_1library_Linear_1Expression_toString(), Java_parma_1polyhedra_1library_MIP_1Problem_build_1cpp_1object__JLparma_1polyhedra_1library_Constraint_1System_2Lparma_1polyhedra_1library_Linear_1Expression_2Lparma_1polyhedra_1library_Optimization_1Mode_2(), and Java_parma_1polyhedra_1library_MIP_1Problem_set_1objective_1function().

10.4.1.19 Optimization_Mode Parma_Polyhedra_Library::Interfaces::Java::build_cxx_optimization_mode (JNIEnv * *env*, jobject *j_opt_mode*)

Builds a C++ Optimization_Mode from Java parma_polyhedra_library::Optimization_Mode `j_opt_mode`.

Referenced by Java_parma_1polyhedra_1library_MIP_1Problem_build_1cpp_1object__JLparma_1polyhedra_1library_Constraint_1System_2Lparma_1polyhedra_1library_Linear_1Expression_2Lparma_1polyhedra_1library_Optimization_1Mode_2(), and Java_parma_1polyhedra_1library_MIP_1Problem_set_1optimization_1mode().

10.4.1.20 PIP_Problem::Control_Parameter_Name Parma_Polyhedra_Library::Interfaces::Java::build_cxx_pip_problem_control_parameter_name (JNIEnv * *env*, jobject *j_cp_name*)

Builds a C++ PIP_Problem::Control_Parameter_Name from Java parma_polyhedra_library::PIP_Problem_Control_Parameter_Name `j_cp_name`.

Referenced by Java_parma_1polyhedra_1library_PIP_1Problem_get_1pip_1problem_1control_1parameter().

10.4.1.21 PIP_Problem::Control_Parameter_Value Parma_Polyhedra_Library::Interfaces::Java::build_cxx_pip_problem_control_parameter_value (JNIEnv * *env*, jobject *j_cp_value*)

Builds a C++ PIP_Problem::Control_Parameter_Value from Java parma_polyhedra_library::PIP_Problem_Control_Parameter_Value `j_cp_value`.

Referenced by Java_parma_1polyhedra_1library_PIP_1Problem_set_1pip_1problem_1control_1parameter().

10.4.1.22 Relation_Symbol Parma_Polyhedra_Library::Interfaces::Java::build_cxx_relsym (JNIEnv * *env*, jobject *j_relsym*)

Builds a C++ Relation_Symbol from Java parma_polyhedra_library::Relation_Symbol `j_relsym`.

10.4.1.23 template<typename System , typename Elem_Builder > System Parma_Polyhedra_Library::Interfaces::Java::build_cxx_system (JNIEnv * *env*, jobject *j_iterable*, Elem_Builder *build_cxx_elem*) [inline]

Definition at line 192 of file ppl_java_common.inlines.hh.

References cached_FMIDs, CHECK_EXCEPTION_ASSERT, CHECK_EXCEPTION_THROW, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::System_Iterator_has_next_ID, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::System_iterator_ID, and Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::System_Iterator_next_ID.

10.4.1.24 Variable Parma_Polyhedra_Library::Interfaces::Java::build_cxx_variable (JNIEnv * *env*, jobject *j_var*) [inline]

Builds a C++ Variable from Java parma_polyhedra_library::Variable `j_var`.

Definition at line 149 of file ppl_java_common.inlines.hh.

References cached_FMIDs, and Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Variable_varid_ID.

Referenced by Java_parma_1polyhedra_1library_PIP_1Solution_1Node_parametric_1values().

10.4.1.25 Variables_Set Parma_Polyhedra_Library::Interfaces::Java::build_cxx_variables_set (JNIEnv * *env*, jobject *j_v_set*)

Builds a C++ Variables_Set from Java parma_polyhedra_library::Variables_Set `v_set`.

Referenced by Java_parma_1polyhedra_1library_MIP_1Problem_add_1to_1integer_1space_1dimensions(), Java_parma_1polyhedra_1library_PIP_1Problem_add_1to_1parameter_1space_1dimensions(), and Java_parma_1polyhedra_1library_PIP_1Problem_build_1cpp_1object__JLparma_1polyhedra_1library_Constraint_1System_2Lparma_1polyhedra_1library_Variables_1Set_2().

10.4.1.26 jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_artificial_parameter (JNIEnv * *env*, const PIP_Tree_Node::Artificial_Parameter & *art*)

Builds a Java parma_polyhedra_library::Artificial_Parameter from C++ Artificial_Parameter `ap`.

Referenced by Java_parma_1polyhedra_1library_PIP_1Tree_1Node_artificials().

10.4.1.27 jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_artificial_parameter_sequence (JNIEnv * *env*, const PIP_Tree_Node::Artificial_Parameter_Sequence & *aps*)

Builds a Java parma_polyhedra_library::Artificial_Parameter_Sequence from C++ Artificial_Parameter_Sequence `aps`.

10.4.1.28 jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_coeff (JNIEnv * *env*, const Coefficient & *ppl_coeff*) [inline]

Builds a Java parma_polyhedra_library::Coefficient from C++ Coefficient `ppl_coeff`.

Definition at line 177 of file ppl_java_common.inlines.hh.

References cached_classes, cached_FMIDs, CHECK_RESULT_THROW, Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Coefficient, and Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Coefficient_init_from_String_ID.

Referenced by build_linear_expression(), Java_parma_1polyhedra_1library_MIP_1Problem_evaluate_1objective_1function(), Java_parma_1polyhedra_1library_MIP_1Problem_objective_1function(), and Java_parma_1polyhedra_1library_MIP_1Problem_optimal_1value().

10.4.1.29 jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_congruence (JNIEnv * *env*, const Congruence & *cg*)

Builds a Java parma_polyhedra_library::Congruence from C++ Congruence `cg`.

10.4.1.30 jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_congruence_system (JNIEnv * *env*, const Congruence_System & *cgs*)

Builds a Java parma_polyhedra_library::Congruence_System from C++ Congruence_System `cgs`.

10.4.1.31 jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_constraint (JNIEnv * *env*, const Constraint & *c*)

Builds a Java parma_polyhedra_library::Constraint from C++ Constraint `c`.

Referenced by Java_parma_1polyhedra_1library_MIP_1Problem_constraints(), Java_parma_-1polyhedra_1library_PIP_1Problem_constraint_1at_1index(), and Java_parma_1polyhedra_1library_-PIP_1Problem_constraints().

10.4.1.32 jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_constraint_system (JNIEnv * *env*, const Constraint_System & *cs*)

Builds a Java parma_polyhedra_library::Constraint_System from C++ Constraint_System `cs`.

Referenced by Java_parma_1polyhedra_1library_PIP_1Tree_1Node_constraints().

10.4.1.33 jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_control_-parameter_name (JNIEnv * *env*, const MIP_Problem::Control_Parameter_Name & *cp_name*)

Builds a Java parma_polyhedra_library::Control_Parameter_Name from C++ MIP_Problem::Control_-Parameter_Name `cp_name`.

10.4.1.34 jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_control_-parameter_value (JNIEnv * *env*, const MIP_Problem::Control_Parameter_Value & *cp_value*)

Builds a Java parma_polyhedra_library::Control_Parameter_Value from C++ MIP_Problem::Control_-Parameter_Value `cp_value`.

Referenced by Java_parma_1polyhedra_1library_MIP_1Problem_get_1control_1parameter().

10.4.1.35 jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_generator (JNIEnv * *env*, const Generator & *g*)

Builds a Java parma_polyhedra_library::Generator from C++ Generator `g`.

Referenced by Java_parma_1polyhedra_1library_MIP_1Problem_feasible_1point(), and Java_parma_-1polyhedra_1library_MIP_1Problem_optimizing_1point().

10.4.1.36 jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_generator_system (JNIEnv * *env*, const Generator_System & *gs*)

Builds a Java parma_polyhedra_library::Generator_System from C++ Generator_System `gs`.

10.4.1.37 jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_grid_generator (JNIEnv * *env*, const Grid_Generator & *g*)

Builds a Java parma_polyhedra_library::Grid_Generator from C++ Grid_Generator `g`.

10.4.1.38 jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_grid_generator_system (JNIEnv * *env*, const Grid_Generator_System & *gs*)

Builds a Java parma_polyhedra_library::Grid_Generator_System from C++ Grid_Generator_System `gs`.

10.4.1.39 jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_linear_expression_coefficient (JNIEnv * *env*, const Coefficient & *c*)

Builds a Java parma_polyhedra_library::Linear_Expression_Coefficient from C++ Coefficient `coeff`.

10.4.1.40 jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_mip_status (JNIEnv * *env*, const MIP_Problem_Status & *mip_status*)

Builds a Java parma_polyhedra_library::MIP_Problem_Status from C++ MIP_Problem_Status `mip_status`.

Referenced by Java_parma_1polyhedra_1library_MIP_1Problem_solve().

10.4.1.41 jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_optimization_mode (JNIEnv * *env*, const Optimization_Mode & *opt_mode*)

Builds a Java parma_polyhedra_library::Optimization_Mode from C++ Optimization_Mode `opt_mode`.

Referenced by Java_parma_1polyhedra_1library_MIP_1Problem_optimization_1mode().

10.4.1.42 jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_pip_problem_control_parameter_name (JNIEnv * *env*, const PIP_Problem::Control_Parameter_Name & *cp_name*)

Builds a Java parma_polyhedra_library::PIP_Problem_Control_Parameter_Name from C++ PIP_Problem::Control_Parameter_Name `cp_name`.

10.4.1.43 jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_pip_problem_control_parameter_value (JNIEnv * *env*, const PIP_Problem::Control_Parameter_Value & *cp_value*)

Builds a Java parma_polyhedra_library::Control_Parameter_Value from C++ PIP_Problem::Control_Parameter_Value `cp_value`.

Referenced by Java_parma_1polyhedra_1library_PIP_1Problem_get_1pip_1problem_1control_-1parameter().

10.4.1.44 jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_pip_status (JNIEnv * *env*, const PIP_Problem_Status & *pip_status*)

Builds a Java parma_polyhedra_library::PIP_Problem_Status from C++ PIP_Problem_Status `pip_-status`.

Referenced by Java_parma_1polyhedra_1library_PIP_1Problem_solve().

10.4.1.45 jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_poly_con_relation (JNIEnv * *env*, Poly_Con_Relation & *r*)

Builds a Java parma_polyhedra_library::Poly_Con_Relation from C++ Poly_Con_Relation `r`.

10.4.1.46 jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_poly_gen_relation (JNIEnv * *env*, Poly_Gen_Relation & *r*)

Builds a Java parma_polyhedra_library::Poly_Gen_Relation from C++ Poly_Gen_Relation `r`.

10.4.1.47 jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_variable (JNIEnv * *env*, const Variable & *var*) `[inline]`

Builds a Java parma_polyhedra_library::Variable from C++ Variable `var`.

Definition at line 154 of file ppl_java_common.inlines.hh.

References cached_classes, cached_FMIDs, CHECK_RESULT_THROW, Parma_-Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Variable, and Parma_Polyhedra_-Library::Interfaces::Java::Java_FMID_Cache::Variable_init_ID.

Referenced by build_linear_expression().

10.4.1.48 jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_variables_set (JNIEnv * *env*, const Variables_Set & *v_set*)

Builds a Java parma_polyhedra_library::Variables_Set from C++ Variables_Set `v_set`.

Referenced by Java_parma_1polyhedra_1library_MIP_1Problem_integer_1space_1dimensions(), and Java_parma_1polyhedra_1library_PIP_1Problem_parameter_1space_1dimensions().

10.4.1.49 template<typename R > jobject Parma_Polyhedra_Library::Interfaces::Java::build_linear_expression (JNIEnv * *env*, const R & *r*) [inline]

Builds a Java parma_polyhedra_library::Linear_Expression from the C++ constraint/congruence `r`.

Definition at line 98 of file ppl_java_common.inlines.hh.

References build_java_coeff(), build_java_variable(), cached_classes, cached_FMIDs, CHECK_EXCEPTION_THROW, CHECK_RESULT_THROW, Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Linear_Expression_Coefficient, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Linear_Expression_Coefficient_init_ID, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Linear_Expression_sum_ID, Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Linear_Expression_Times, and Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Linear_Expression_Times_init_from_coeff_var_ID.

Referenced by Java_parma_1polyhedra_1library_MIP_1Problem_objective_1function(), and Java_parma_1polyhedra_1library_PIP_1Solution_1Node_parametric_1values().

10.4.1.50 jobject Parma_Polyhedra_Library::Interfaces::Java::build_pip_problem_java_control_parameter_name (JNIEnv * *env*, const PIP_Problem::Control_Parameter_Name & *cp_name*)

Definition at line 743 of file ppl_java_common.cc.

10.4.1.51 jobject Parma_Polyhedra_Library::Interfaces::Java::get_by_reference (JNIEnv * *env*, jobject *by_reference*) [inline]

Returns the object referenced by parma_polyhedra_library::By_Reference object `by_reference`.

Definition at line 92 of file ppl_java_common.inlines.hh.

References Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::By_Reference_obj_ID, and cached_FMIDs.

10.4.1.52 jobject Parma_Polyhedra_Library::Interfaces::Java::get_pair_element (JNIEnv * *env*, int *arg*, jobject *pair*)

Returns one of the fields of the parma_polyhedra_library::Pair object `pair`.

If `arg` is 0, the first element of `pair` is returned; if `arg` is 1, the second element of `pair` is returned.

Exceptions

std::runtime_error Thrown if `arg` is neither 0 nor 1.

10.4.1.53 void * Parma_Polyhedra_Library::Interfaces::Java::get_ptr (JNIEnv * *env*, jobject *ppl_object*) [inline]

Returns a pointer to the C++ object wrapped by `ppl_object`.

Definition at line 60 of file ppl_java_common.inlines.hh.

References cached_FMIDs, and Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::PPL_-Object_ptr_ID.

Referenced by Java_parma_1polyhedra_1library_MIP_1Problem_add_1constraint(), Java_parma_-1polyhedra_1library_MIP_1Problem_add_1constraints(), Java_parma_1polyhedra_1library_MIP_-1Problem_add_1space_1dimensions_1and_1embed(), Java_parma_1polyhedra_1library_MIP_-1Problem_add_1to_1integer_1space_1dimensions(), Java_parma_1polyhedra_1library_MIP_1Problem_-ascii_1dump(), Java_parma_1polyhedra_1library_MIP_1Problem_build_1cpp_1object__Lparma_-1polyhedra_1library_MIP_1Problem_2(), Java_parma_1polyhedra_1library_MIP_1Problem_clear(), Java_parma_1polyhedra_1library_MIP_1Problem_constraints(), Java_parma_1polyhedra_1library_-MIP_1Problem_evaluate_1objective_1function(), Java_parma_1polyhedra_1library_MIP_1Problem_-feasible_1point(), Java_parma_1polyhedra_1library_MIP_1Problem_finalize(), Java_parma_-1polyhedra_1library_MIP_1Problem_free(), Java_parma_1polyhedra_1library_MIP_1Problem_-get_1control_1parameter(), Java_parma_1polyhedra_1library_MIP_1Problem_integer_1space_-1dimensions(), Java_parma_1polyhedra_1library_MIP_1Problem_is_1satisfiable(), Java_parma_-1polyhedra_1library_MIP_1Problem_max_1space_1dimension(), Java_parma_1polyhedra_1library_-MIP_1Problem_objective_1function(), Java_parma_1polyhedra_1library_MIP_1Problem_OK(), Java_parma_1polyhedra_1library_MIP_1Problem_optimal_1value(), Java_parma_1polyhedra_-1library_MIP_1Problem_optimization_1mode(), Java_parma_1polyhedra_1library_MIP_1Problem_-optimizing_1point(), Java_parma_1polyhedra_1library_MIP_1Problem_set_1control_1parameter(), Java_parma_1polyhedra_1library_MIP_1Problem_set_1objective_1function(), Java_parma_1polyhedra_-1library_MIP_1Problem_set_1optimization_1mode(), Java_parma_1polyhedra_1library_MIP_-1Problem_solve(), Java_parma_1polyhedra_1library_MIP_1Problem_space_1dimension(), Java_-parma_1polyhedra_1library_MIP_1Problem_toString(), Java_parma_1polyhedra_1library_MIP_-1Problem_total_1memory_1in_1bytes(), Java_parma_1polyhedra_1library_Partial_1Function_finalize(), Java_parma_1polyhedra_1library_Partial_1Function_free(), Java_parma_1polyhedra_1library_Partial_-1Function_has_1empty_1codomain(), Java_parma_1polyhedra_1library_Partial_1Function_insert(), Java_parma_1polyhedra_1library_Partial_1Function_maps(), Java_parma_1polyhedra_1library_Partial_-1Function_max_1in_1codomain(), Java_parma_1polyhedra_1library_PIP_1_problem_finalize(), Java_-parma_1polyhedra_1library_PIP_1Decision_1Node_child_1node(), Java_parma_1polyhedra_1library_-PIP_1Problem_add_1constraint(), Java_parma_1polyhedra_1library_PIP_1Problem_add_1constraints(), Java_parma_1polyhedra_1library_PIP_1Problem_add_1space_1dimensions_1and_1embed(), Java_-parma_1polyhedra_1library_PIP_1Problem_add_1to_1parameter_1space_1dimensions(), Java_parma_-1polyhedra_1library_PIP_1Problem_ascii_1dump(), Java_parma_1polyhedra_1library_PIP_1Problem_-build_1cpp_1object__Lparma_1polyhedra_1library_PIP_1Problem_2(), Java_parma_1polyhedra_-1library_PIP_1Problem_constraint_1at_1index(), Java_parma_1polyhedra_1library_PIP_1Problem_-constraints(), Java_parma_1polyhedra_1library_PIP_1Problem_external_1memory_1in_1bytes(), Java_-parma_1polyhedra_1library_PIP_1Problem_free(), Java_parma_1polyhedra_1library_PIP_1Problem_-get_1big_1parameter_1dimension(), Java_parma_1polyhedra_1library_PIP_1Problem_get_1pip_-1problem_1control_1parameter(), Java_parma_1polyhedra_1library_PIP_1Problem_is_1satisfiable(), Java_parma_1polyhedra_1library_PIP_1Problem_max_1space_1dimension(), Java_parma_1polyhedra_-1library_PIP_1Problem_number_1of_1constraints(), Java_parma_1polyhedra_1library_PIP_1Problem_-number_1of_1parameter_1space_1dimensions(), Java_parma_1polyhedra_1library_PIP_1Problem_OK(), Java_parma_1polyhedra_1library_PIP_1Problem_optimizing_1solution(), Java_parma_1polyhedra_-1library_PIP_1Problem_parameter_1space_1dimensions(), Java_parma_1polyhedra_1library_PIP_-1Problem_set_1big_1parameter_1dimension(), Java_parma_1polyhedra_1library_PIP_1Problem_set_-1pip_1problem_1control_1parameter(), Java_parma_1polyhedra_1library_PIP_1Problem_solution(),

Java_parma_1polyhedra_1library_PIP_1Problem_solve(), Java_parma_1polyhedra_1library_PIP_-1Problem_space_1dimension(), Java_parma_1polyhedra_1library_PIP_1Problem_toString(), Java_-parma_1polyhedra_1library_PIP_1Problem_total_1memory_1in_1bytes(), Java_parma_1polyhedra_-1library_PIP_1Solution_1Node_parametric_1values(), Java_parma_1polyhedra_1library_PIP_1Tree_-1Node_artificials(), Java_parma_1polyhedra_1library_PIP_1Tree_1Node_as_1decision(), Java_parma_-1polyhedra_1library_PIP_1Tree_1Node_as_1solution(), Java_parma_1polyhedra_1library_PIP_1Tree_-1Node_constraints(), Java_parma_1polyhedra_1library_PIP_1Tree_1Node_finalize(), Java_parma_-1polyhedra_1library_PIP_1Tree_1Node_free(), Java_parma_1polyhedra_1library_PIP_1Tree_1Node_-number_1of_1artificials(), Java_parma_1polyhedra_1library_PIP_1Tree_1Node_OK(), and Java_parma_-1polyhedra_1library_PIP_1Tree_1Node_toString().

10.4.1.54 void Parma_Polyhedra_Library::Interfaces::Java::handle_exception (JNIEnv * *env*)

10.4.1.55 void Parma_Polyhedra_Library::Interfaces::Java::handle_exception (JNIEnv * *env*, const deterministic_timeout_exception &)

10.4.1.56 void Parma_Polyhedra_Library::Interfaces::Java::handle_exception (JNIEnv * *env*, const timeout_exception &)

10.4.1.57 void Parma_Polyhedra_Library::Interfaces::Java::handle_exception (JNIEnv * *env*, const std::exception & *e*)

10.4.1.58 void Parma_Polyhedra_Library::Interfaces::Java::handle_exception (JNIEnv * *env*, const std::bad_alloc &)

10.4.1.59 void Parma_Polyhedra_Library::Interfaces::Java::handle_exception (JNIEnv * *env*, const std::domain_error & *e*)

10.4.1.60 void Parma_Polyhedra_Library::Interfaces::Java::handle_exception (JNIEnv * *env*, const std::length_error & *e*)

10.4.1.61 void Parma_Polyhedra_Library::Interfaces::Java::handle_exception (JNIEnv * *env*, const std::logic_error & *e*)

10.4.1.62 void Parma_Polyhedra_Library::Interfaces::Java::handle_exception (JNIEnv * *env*, const std::invalid_argument & *e*)

10.4.1.63 void Parma_Polyhedra_Library::Interfaces::Java::handle_exception (JNIEnv * *env*, const std::overflow_error & *e*)

10.4.1.64 bool Parma_Polyhedra_Library::Interfaces::Java::is_java_marked (JNIEnv * *env*, jobject *ppl_object*) [inline]

Returns `true` if and only if the Java object `ppl_object` refers to a C++ object.

Definition at line 69 of file ppl_java_common.inlines.hh.

References cached_FMIDs, and Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::PPL_-Object_ptr_ID.

Referenced by Java_parma_1polyhedra_1library_MIP_1Problem_finalize(), Java_parma_1polyhedra_-1library_MIP_1Problem_free(), Java_parma_1polyhedra_1library_Partial_1Function_finalize(), Java_parma_1polyhedra_1library_Partial_1Function_free(), Java_parma_1polyhedra_1library_PIP_-1_problem_finalize(), Java_parma_1polyhedra_1library_PIP_1Problem_free(), Java_parma_1polyhedra_-1library_PIP_1Tree_1Node_finalize(), and Java_parma_1polyhedra_1library_PIP_1Tree_1Node_free().

10.4.1.65 jobject Parma_Polyhedra_Library::Interfaces::Java::j_int_to_j_integer (JNIEnv * *env*, jint *jint_value*)

Builds a Java Integer from Java int `value`.

10.4.1.66 jint Parma_Polyhedra_Library::Interfaces::Java::j_integer_to_j_int (JNIEnv * *env*, jobject *j_integer*)

Returns the Java int stored in Java Integer `j_integer`.

10.4.1.67 jlong Parma_Polyhedra_Library::Interfaces::Java::j_long_class_to_j_long (JNIEnv * *env*, jobject *j_long*)

Returns the Java long stored in Java Long `j_long`.

10.4.1.68 jobject Parma_Polyhedra_Library::Interfaces::Java::j_long_to_j_long_class (JNIEnv * *env*, jlong *jlong_value*)

Builds a Java Long from Java long `value`.

10.4.1.69 template<typename U , typename V > U Parma_Polyhedra_Library::Interfaces::Java::jtype_to_unsigned (const V & *value*) [inline]

Builds an unsigned C++ number from the Java native number `value`.

Parameters

value The Java native number of type V to be converted.

Exceptions

std::invalid_argument Thrown if `value` is negative.

Definition at line 37 of file ppl_java_common.inlines.hh.

10.4.1.70 void Parma_Polyhedra_Library::Interfaces::Java::reset_deterministic_timeout ()

Referenced by Java_parma_1polyhedra_1library_Parma_1Polyhedra_1Library_reset_1deterministic_1timeout(), and Java_parma_1polyhedra_1library_Parma_1Polyhedra_1Library_set_1deterministic_1timeout().

10.4.1.71 void Parma_Polyhedra_Library::Interfaces::Java::reset_timeout ()

Referenced by Java_parma_1polyhedra_1library_Parma_1Polyhedra_1Library_reset_1timeout(), and Java_parma_1polyhedra_1library_Parma_1Polyhedra_1Library_set_1timeout().

10.4.1.72 void Parma_Polyhedra_Library::Interfaces::Java::set_by_reference (JNIEnv * *env*, jobject *by_ref_dst*, jobject *src*) [inline]

Modifies parma_polyhedra_library::By_Reference object `by_ref_dst` so that it references object `src`.

Definition at line 85 of file ppl_java_common.inlines.hh.

References Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::By_Reference_obj_ID, and cached_FMIDs.

10.4.1.73 void Parma_Polyhedra_Library::Interfaces::Java::set_coefficient (JNIEnv * *env*, jobject *dst*, jobject *src*) [inline]

Sets Java Coefficient `dst` to have the same value as `src`.

Definition at line 78 of file ppl_java_common.inlines.hh.

References cached_FMIDs, and Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Coefficient_value_ID.

Referenced by Java_parma_1polyhedra_1library_MIP_1Problem_evaluate_1objective_1function(), and Java_parma_1polyhedra_1library_MIP_1Problem_optimal_1value().

10.4.1.74 void Parma_Polyhedra_Library::Interfaces::Java::set_generator (JNIEnv * *env*, jobject *dst*, jobject *src*)

Sets Java parma_polyhedra_library::Generator `dst` to have the same value as `src`.

10.4.1.75 void Parma_Polyhedra_Library::Interfaces::Java::set_pair_element (JNIEnv * *env*, jobject *dst_pair*, int *arg*, jobject *src*)

Assigns `src` to one of the fields of parma_polyhedra_library::Pair object `dst_pair`.

If `arg` is 0, the first element of `dst_pair` is overwritten; if `arg` is 1, the second element of `dst_pair` is overwritten.

Exceptions

std::runtime_error Thrown if `arg` is neither 0 nor 1.

10.4.1.76 template<typename T > void Parma_Polyhedra_Library::Interfaces::Java::set_ptr (JNIEnv * *env*, jobject *ppl_object*, const T * *address*, bool *to_be_marked* = false) [inline]

Sets the pointer of the underlying C++ object in the Java object.

Definition at line 51 of file ppl_java_common.inlines.hh.

References cached_FMIDs, and Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::PPL_-Object_ptr_ID.

Referenced by Java_parma_1polyhedra_1library_MIP_1Problem_build_1cpp_1object__J(), Java_-parma_1polyhedra_1library_MIP_1Problem_build_1cpp_1object__JLparma_1polyhedra_1library_-Constraint_1System_2Lparma_1polyhedra_1library_Linear_1Expression_2Lparma_1polyhedra_-1library_Optimization_1Mode_2(), Java_parma_1polyhedra_1library_MIP_1Problem_build_1cpp_-1object__Lparma_1polyhedra_1library_MIP_1Problem_2(), Java_parma_1polyhedra_1library_-MIP_1Problem_free(), Java_parma_1polyhedra_1library_Partial_1Function_build_1cpp_1object(), Java_parma_1polyhedra_1library_Partial_1Function_free(), Java_parma_1polyhedra_1library_PIP_-1Decision_1Node_child_1node(), Java_parma_1polyhedra_1library_PIP_1Problem_build_1cpp_-

1object__J(), Java_parma_1polyhedra_1library_PIP_1Problem_build_1cpp_1object__JLparma_-1polyhedra_1library_Constraint_1System_2Lparma_1polyhedra_1library_Variables_1Set_2(), Java_-parma_1polyhedra_1library_PIP_1Problem_build_1cpp_1object__Lparma_1polyhedra_1library_PIP_-1Problem_2(), Java_parma_1polyhedra_1library_PIP_1Problem_free(), Java_parma_1polyhedra_-1library_PIP_1Problem_optimizing_1solution(), Java_parma_1polyhedra_1library_PIP_1Problem_-solution(), Java_parma_1polyhedra_1library_PIP_1Tree_1Node_as_1decision(), Java_parma_-1polyhedra_1library_PIP_1Tree_1Node_as_1solution(), and Java_parma_1polyhedra_1library_PIP_-1Tree_1Node_free().

10.4.2 Variable Documentation

10.4.2.1 Java_Class_Cache Parma_Polyhedra_Library::Interfaces::Java::cached_classes

The cached class references.

Definition at line 28 of file ppl_java_common.cc.

Referenced by build_java_coeff(), build_java_variable(), build_linear_expression(), Java_parma_-1polyhedra_1library_Coefficient_initIDs(), Java_parma_1polyhedra_1library_Constraint_1System_-initIDs(), Java_parma_1polyhedra_1library_MIP_1Problem_constraints(), Java_parma_1polyhedra_-1library_MIP_1Problem_objective_1function(), Java_parma_1polyhedra_1library_Parma_1Polyhedra_-1Library_finalize_1library(), Java_parma_1polyhedra_1library_Parma_1Polyhedra_1Library_initialize_-1library(), Java_parma_1polyhedra_1library_PIP_1Problem_constraints(), Java_parma_1polyhedra_-1library_PIP_1Tree_1Node_artificials(), and Java_parma_1polyhedra_1library_PIP_1Tree_1Node_-constraints().

10.4.2.2 Java_FMID_Cache Parma_Polyhedra_Library::Interfaces::Java::cached_FMIDs

The field and method ID cache.

Definition at line 29 of file ppl_java_common.cc.

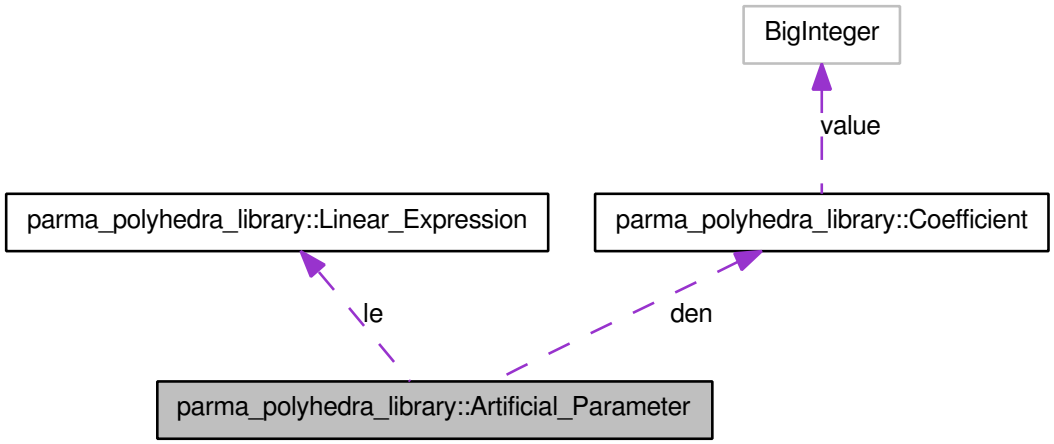
Referenced by build_cxx_coeff(), build_cxx_system(), build_cxx_variable(), build_java_coeff(), build_java_variable(), build_linear_expression(), get_by_reference(), get_ptr(), is_java_marked(), Java_parma_1polyhedra_1library_Artificial_1Parameter_1Sequence_initIDs(), Java_parma_1polyhedra_-1library_Artificial_1Parameter_initIDs(), Java_parma_1polyhedra_1library_Bounded_1Integer_1Type_-1Overflow_initIDs(), Java_parma_1polyhedra_1library_Bounded_1Integer_1Type_1Representation_-initIDs(), Java_parma_1polyhedra_1library_Bounded_1Integer_1Type_1Width_initIDs(), Java_parma_-1polyhedra_1library_By_1Reference_initIDs(), Java_parma_1polyhedra_1library_Coefficient_initIDs(), Java_parma_1polyhedra_1library_Complexity_1Class_initIDs(), Java_parma_1polyhedra_1library_-Congruence_1System_initIDs(), Java_parma_1polyhedra_1library_Congruence_initIDs(), Java_parma_-1polyhedra_1library_Constraint_1System_initIDs(), Java_parma_1polyhedra_1library_Constraint_-initIDs(), Java_parma_1polyhedra_1library_Degenerate_1Element_initIDs(), Java_parma_1polyhedra_-1library_Generator_1System_initIDs(), Java_parma_1polyhedra_1library_Generator_1Type_initIDs(), Java_parma_1polyhedra_1library_Generator_initIDs(), Java_parma_1polyhedra_1library_Grid_-1Generator_1System_initIDs(), Java_parma_1polyhedra_1library_Grid_1Generator_1Type_initIDs(), Java_parma_1polyhedra_1library_Grid_1Generator_initIDs(), Java_parma_1polyhedra_1library_-Linear_1Expression_1Coefficient_initIDs(), Java_parma_1polyhedra_1library_Linear_1Expression_-1Difference_initIDs(), Java_parma_1polyhedra_1library_Linear_1Expression_1Sum_initIDs(), Java_-parma_1polyhedra_1library_Linear_1Expression_1Times_initIDs(), Java_parma_1polyhedra_1library_-Linear_1Expression_1Unary_1Minus_initIDs(), Java_parma_1polyhedra_1library_Linear_1Expression_-

1Variable_initIDs(), Java_parma_1polyhedra_1library_Linear_1Expression_initIDs(), Java_parma_-1polyhedra_1library_MIP_1Problem_1Status_initIDs(), Java_parma_1polyhedra_1library_MIP_-1Problem_constraints(), Java_parma_1polyhedra_1library_MIP_1Problem_objective_1function(), Java_-parma_1polyhedra_1library_Optimization_1Mode_initIDs(), Java_parma_1polyhedra_1library_Pair_-initIDs(), Java_parma_1polyhedra_1library_PIP_1Problem_1Status_initIDs(), Java_parma_1polyhedra_-1library_PIP_1Problem_constraints(), Java_parma_1polyhedra_1library_PIP_1Tree_1Node_artificials(), Java_parma_1polyhedra_1library_PIP_1Tree_1Node_constraints(), Java_parma_1polyhedra_1library_-Poly_1Con_1Relation_initIDs(), Java_parma_1polyhedra_1library_Poly_1Gen_1Relation_initIDs(), Java_parma_1polyhedra_1library_PPL_1Object_initIDs(), Java_parma_1polyhedra_1library_Relation_-1Symbol_initIDs(), Java_parma_1polyhedra_1library_Variable_initIDs(), Java_parma_1polyhedra_-1library_Variables_1Set_initIDs(), set_by_reference(), set_coefficient(), and set_ptr().

11 Class Documentation

11.1 parma_polyhedra_library::Artificial_Parameter Class Reference

Collaboration diagram for parma_polyhedra_library::Artificial_Parameter:



Public Member Functions

- Artificial_Parameter (Linear_Expression e, Coefficient d)

  Builds an artificial parameter from a linear expression and a denominator.

- Linear_Expression linear_expression ()

  Returns the linear expression in artificial parameter `this`.

- Coefficient denominator ()

  Returns the denominator in artificial parameter `this`.

- native String ascii_dump ()

  Returns an ascii formatted internal representation of `this`.

- native String toString ()

  Returns a string representation of `this`.

Static Package Functions

- [static initializer]

Static Private Member Functions

- static native void initIDs ()

Private Attributes

- Linear_Expression le

  The value of the linear expression of `this`.

- Coefficient den

  The value of the denominatir of `this`.

11.1.1 Detailed Description

An Artificial_Parameter object represents the result of the integer division of a Linear_Expression (on the other parameters, including the previously-defined artificials) by an integer denominator (a Coefficient object). The dimensions of the artificial parameters (if any) in a tree node have consecutive indices starting from `dim+1`, where the value of `dim` is computed as follows:

- for the tree root node, `dim` is the space dimension of the PIP_Problem;
- for any other node of the tree, it is recusrively obtained by adding the value of `dim` computed for the parent node to the number of artificial parameters defined in the parent node.

> Since the numbering of dimensions for artificial parameters follows the rule above, the addition of new problem variables and/or new problem parameters to an already solved PIP_Problem object (as done when incrementally solving a problem) will result in the systematic renumbering of all the existing artificial parameters.

Definition at line 49 of file Artificial_Parameter.java.

11.1.2 Constructor & Destructor Documentation

11.1.2.1 parma_polyhedra_library::Artificial_Parameter::Artificial_Parameter (Linear_Expression *e*, Coefficient *d*) `[inline]`

Builds an artificial parameter from a linear expression and a denominator.

Definition at line 61 of file Artificial_Parameter.java.

References parma_polyhedra_library::Linear_Expression::clone(), den, and le.

11.1.3 Member Function Documentation

11.1.3.1 parma_polyhedra_library::Artificial_Parameter::[static initializer] () [inline, static, package]

11.1.3.2 native String parma_polyhedra_library::Artificial_Parameter::ascii_dump ()

Returns an ascii formatted internal representation of `this`.

11.1.3.3 Coefficient parma_polyhedra_library::Artificial_Parameter::denominator () [inline]

Returns the denominator in artificial parameter `this`.

Definition at line 76 of file Artificial_Parameter.java.

References den.

11.1.3.4 static native void parma_polyhedra_library::Artificial_Parameter::initIDs () [static, private]

11.1.3.5 Linear_Expression parma_polyhedra_library::Artificial_Parameter::linear_expression () [inline]

Returns the linear expression in artificial parameter `this`.

Definition at line 69 of file Artificial_Parameter.java.

References le.

11.1.3.6 native String parma_polyhedra_library::Artificial_Parameter::toString ()

Returns a string representation of `this`.

11.1.4 Member Data Documentation

11.1.4.1 Coefficient parma_polyhedra_library::Artificial_Parameter::den [private]

The value of the denominatir of `this`.

Definition at line 55 of file Artificial_Parameter.java.

Referenced by Artificial_Parameter(), and denominator().

11.1.4.2 Linear_Expression parma_polyhedra_library::Artificial_Parameter::le [private]

The value of the linear expression of `this`.

Definition at line 52 of file Artificial_Parameter.java.

Referenced by Artificial_Parameter(), and linear_expression().

The documentation for this class was generated from the following file:

- Artificial_Parameter.java

11.2 parma_polyhedra_library::Artificial_Parameter_Sequence Class Reference

A sequence of artificial parameters.

Public Member Functions

- Artificial_Parameter_Sequence ()
  Default constructor: builds an empty sequence of artificial parameters.

Static Package Functions

- [static initializer]

Static Private Member Functions

- static native void initIDs ()

11.2.1 Detailed Description

A sequence of artificial parameters. An object of the class Artificial_Parameter_Sequence is a sequence of artificial parameters.

Definition at line 34 of file Artificial_Parameter_Sequence.java.

11.2.2 Constructor & Destructor Documentation

11.2.2.1 parma_polyhedra_library::Artificial_Parameter_Sequence::Artificial_Parameter_Sequence () [inline]

Default constructor: builds an empty sequence of artificial parameters.

Definition at line 38 of file Artificial_Parameter_Sequence.java.

11.2.3 Member Function Documentation

11.2.3.1 parma_polyhedra_library::Artificial_Parameter_Sequence::[static initializer] () [inline, static, package]

11.2.3.2 static native void parma_polyhedra_library::Artificial_Parameter_Sequence::initIDs () [static, private]

The documentation for this class was generated from the following file:

- Artificial_Parameter_Sequence.java

11.3 parma_polyhedra_library::By_Reference< T > Class Reference

An utility class implementing mutable and non-mutable call-by-reference.

Public Member Functions

- By_Reference (T object_value)

  Builds an object encapsulating `object_value`.

- void set (T y)

  Set an object to value `object_value`.

- T get ()

  Returns the value held by `this`.

Static Package Functions

- [static initializer]

Package Attributes

- T obj

  Stores the object.

Static Private Member Functions

- static native void initIDs ()

11.3.1 Detailed Description

An utility class implementing mutable and non-mutable call-by-reference.

Definition at line 28 of file By_Reference.java.

11.3.2 Constructor & Destructor Documentation

11.3.2.1 parma_polyhedra_library::By_Reference< T >::By_Reference (T *object_value*) [inline]

Builds an object encapsulating `object_value`.

Definition at line 33 of file By_Reference.java.

11.3.3 Member Function Documentation

11.3.3.1 parma_polyhedra_library::By_Reference< T >::[static initializer] () [inline, static, package]

11.3.3.2 T parma_polyhedra_library::By_Reference< T >::get () [inline]

Returns the value held by `this`.

Definition at line 43 of file By_Reference.java.

11.3.3.3 static native void parma_polyhedra_library::By_Reference< T >::initIDs () [static, private]

11.3.3.4 void parma_polyhedra_library::By_Reference< T >::set (T *y*) [inline]

Set an object to value `object_value`.

Definition at line 38 of file By_Reference.java.

11.3.4 Member Data Documentation

11.3.4.1 T parma_polyhedra_library::By_Reference< T >::obj [package]

Stores the object.

Definition at line 30 of file By_Reference.java.

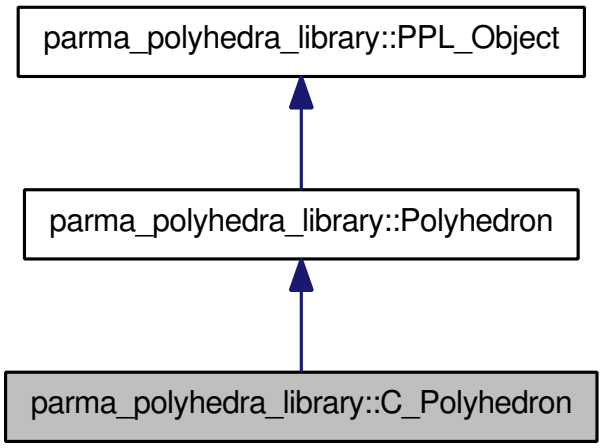
The documentation for this class was generated from the following file:

- By_Reference.java

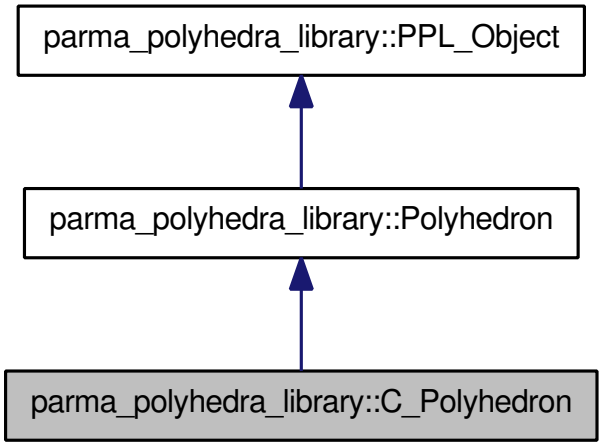
11.4 parma_polyhedra_library::C_Polyhedron Class Reference

A topologically closed convex polyhedron.

Inheritance diagram for parma_polyhedra_library::C_Polyhedron:



Collaboration diagram for parma_polyhedra_library::C_Polyhedron:



Public Member Functions

Standard Constructors and Destructor

- C_Polyhedron (long d, Degenerate_Element kind)
  Builds a new C polyhedron of dimension d.
- C_Polyhedron (C_Polyhedron y)

Builds a new C polyhedron that is copy of `y`.

- C_Polyhedron (C_Polyhedron y, Complexity_Class complexity)
  Builds a new C polyhedron that is a copy of `ph`.
- C_Polyhedron (Constraint_System cs)
  Builds a new C polyhedron from the system of constraints `cs`.
- C_Polyhedron (Congruence_System cgs)
  Builds a new C polyhedron from the system of congruences `cgs`.
- native void free ()
  Releases all resources managed by `this`*, also resetting it to a null reference.*

Constructors Behaving as Conversion Operators

Besides the conversions listed here below, the library also provides conversion operators that build a semantic geometric description starting from ***any*** *other semantic geometric description (e.g.,* `Grid(C_Polyhedron y)`*,* `C_Polyhedron(BD_Shape_mpq_class y)`*, etc.). Clearly, the conversion operators are only available if both the source and the target semantic geometric descriptions have been enabled when configuring the library. The conversions also taking as argument a complexity class sometimes provide non-trivial precision/efficiency trade-offs.*

- C_Polyhedron (NNC_Polyhedron y)
  Builds a C polyhedron that is a copy of the topological closure of the NNC polyhedron `y`.
- C_Polyhedron (NNC_Polyhedron y, Complexity_Class complexity)
  Builds a C polyhedron that is a copy of the topological closure of the NNC polyhedron `y`.
- C_Polyhedron (Generator_System gs)
  Builds a new C polyhedron from the system of generators `gs`.

Other Methods

- native boolean upper_bound_assign_if_exact (C_Polyhedron y)
  If the upper bound of `this` *and* `y` *is exact it is assigned to* `this` *and* `true` *is returned; otherwise* `false` *is returned.*

Static Public Member Functions

- static native Pair< C_Polyhedron, Pointset_Powerset_NNC_Polyhedron > linear_partition (C_Polyhedron p, C_Polyhedron q)
  Partitions `q` *with respect to* `p`.

Protected Member Functions

- native void finalize ()
  Releases all resources managed by `this`.

11.4.1 Detailed Description

A topologically closed convex polyhedron.

Definition at line 1059 of file Fake_Class_for_Doxygen.java.

11.4.2 Constructor & Destructor Documentation

11.4.2.1 parma_polyhedra_library::C_Polyhedron::C_Polyhedron (long *d*, Degenerate_Element *kind*)

Builds a new C polyhedron of dimension `d`.

If `kind` is `EMPTY`, the newly created polyhedron will be empty; otherwise, it will be a universe polyhedron.

11.4.2.2 parma_polyhedra_library::C_Polyhedron::C_Polyhedron (C_Polyhedron *y*)

Builds a new C polyhedron that is copy of `y`.

11.4.2.3 parma_polyhedra_library::C_Polyhedron::C_Polyhedron (C_Polyhedron *y*, Complexity_Class *complexity*)

Builds a new C polyhedron that is a copy of `ph`.

The complexity argument is ignored.

11.4.2.4 parma_polyhedra_library::C_Polyhedron::C_Polyhedron (Constraint_System *cs*)

Builds a new C polyhedron from the system of constraints `cs`.

The new polyhedron will inherit the space dimension of `cs`.

11.4.2.5 parma_polyhedra_library::C_Polyhedron::C_Polyhedron (Congruence_System *cgs*)

Builds a new C polyhedron from the system of congruences `cgs`.

The new polyhedron will inherit the space dimension of `cgs`.

11.4.2.6 parma_polyhedra_library::C_Polyhedron::C_Polyhedron (NNC_Polyhedron *y*)

Builds a C polyhedron that is a copy of the topological closure of the NNC polyhedron `y`.

11.4.2.7 parma_polyhedra_library::C_Polyhedron::C_Polyhedron (NNC_Polyhedron *y*, Complexity_Class *complexity*)

Builds a C polyhedron that is a copy of the topological closure of the NNC polyhedron `y`.

The complexity argument is ignored, since the exact constructor has polynomial complexity.

11.4.2.8 parma_polyhedra_library::C_Polyhedron::C_Polyhedron (Generator_System *gs*)

Builds a new C polyhedron from the system of generators `gs`.

The new polyhedron will inherit the space dimension of `gs`.

11.4.3 Member Function Documentation

11.4.3.1 native void parma_polyhedra_library::C_Polyhedron::finalize () `[protected]`

Releases all resources managed by `this`.

11.4.3.2 native void parma_polyhedra_library::C_Polyhedron::free ()

Releases all resources managed by `this`, also resetting it to a null reference.

11.4.3.3 static native Pair<C_Polyhedron, Pointset_Powerset_NNC_Polyhedron> parma_polyhedra_library::C_Polyhedron::linear_partition (C_Polyhedron *p*, C_Polyhedron *q*) `[static]`

Partitions `q` with respect to `p`.

Let `p` and `q` be two polyhedra. The function returns a pair object `r` such that

- `r.first` is the intersection of `p` and `q`;
- `r.second` has the property that all its elements are pairwise disjoint and disjoint from `p`;
- the set-theoretical union of `r.first` with all the elements of `r.second` gives `q` (i.e., `r` is the representation of a partition of `q`).

11.4.3.4 native boolean parma_polyhedra_library::C_Polyhedron::upper_bound_assign_if_exact (C_Polyhedron *y*)

If the upper bound of `this` and `y` is exact it is assigned to `this` and `true` is returned; otherwise `false` is returned.

Exceptions

Invalid_Argument_Exception Thrown if `this` and `y` are dimension-incompatible.

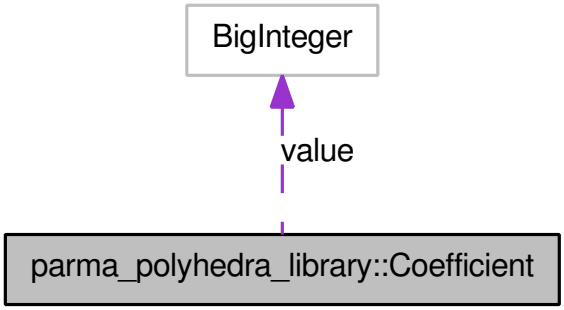
The documentation for this class was generated from the following file:

- Fake_Class_for_Doxygen.java

11.5 parma_polyhedra_library::Coefficient Class Reference

A PPL coefficient.

Collaboration diagram for parma_polyhedra_library::Coefficient:



Public Member Functions

- Coefficient (int i)

  Builds a coefficient valued `i`.

- Coefficient (long l)

  Builds a coefficient valued `l`.

- Coefficient (BigInteger bi)

  Builds a coefficient valued `bi`.

- Coefficient (String s)

  Builds a coefficient from the decimal representation in `s`.

- Coefficient (Coefficient c)

  Builds a copy of `c`.

- String toString ()

  Returns a String representation of `this`.

- BigInteger getBigInteger ()

  Returns the value held by `this`.

Static Public Member Functions

- static native int bits ()

  Returns the number of bits of PPL coefficients; 0 if unbounded.

Static Package Functions

- [static initializer]

Static Private Member Functions

- static native void initIDs ()

Private Attributes

- BigInteger value

  Holds the value of `this`.

11.5.1 Detailed Description

A PPL coefficient. Objects of type Coefficient are used to implement the integral valued coefficients occurring in linear expressions, constraints, generators and so on.

Definition at line 34 of file Coefficient.java.

11.5.2 Constructor & Destructor Documentation

11.5.2.1 parma_polyhedra_library::Coefficient::Coefficient (int *i*) [inline]

Builds a coefficient valued `i`.

Definition at line 40 of file Coefficient.java.

References value.

11.5.2.2 parma_polyhedra_library::Coefficient::Coefficient (long *l*) [inline]

Builds a coefficient valued `l`.

Definition at line 45 of file Coefficient.java.

References value.

11.5.2.3 parma_polyhedra_library::Coefficient::Coefficient (BigInteger *bi*) [inline]

Builds a coefficient valued `bi`.

Definition at line 50 of file Coefficient.java.

References value.

11.5.2.4 parma_polyhedra_library::Coefficient::Coefficient (String *s*) [inline]

Builds a coefficient from the decimal representation in `s`.

Exceptions

java.lang.NumberFormatException Thrown if `s` does not contain a valid decimal representation.

Definition at line 59 of file Coefficient.java.

References value.

11.5.2.5 parma_polyhedra_library::Coefficient::Coefficient (Coefficient *c*) [inline]

Builds a copy of `c`.

Definition at line 64 of file Coefficient.java.

References value.

11.5.3 Member Function Documentation

11.5.3.1 parma_polyhedra_library::Coefficient::[static initializer] () [inline, static, package]

11.5.3.2 static native int parma_polyhedra_library::Coefficient::bits () [static]

Returns the number of bits of PPL coefficients; 0 if unbounded.

11.5.3.3 BigInteger parma_polyhedra_library::Coefficient::getBigInteger () [inline]

Returns the value held by `this`.

Definition at line 74 of file Coefficient.java.

References value.

Referenced by parma_polyhedra_library::Generator::closure_point(), and parma_polyhedra_-library::Generator::point().

11.5.3.4 static native void parma_polyhedra_library::Coefficient::initIDs () [static, private]

11.5.3.5 String parma_polyhedra_library::Coefficient::toString () [inline]

Returns a String representation of `this`.

Definition at line 69 of file Coefficient.java.

References value.

11.5.4 Member Data Documentation

11.5.4.1 BigInteger parma_polyhedra_library::Coefficient::value [private]

Holds the value of `this`.

Definition at line 37 of file Coefficient.java.

Referenced by Coefficient(), getBigInteger(), and toString().

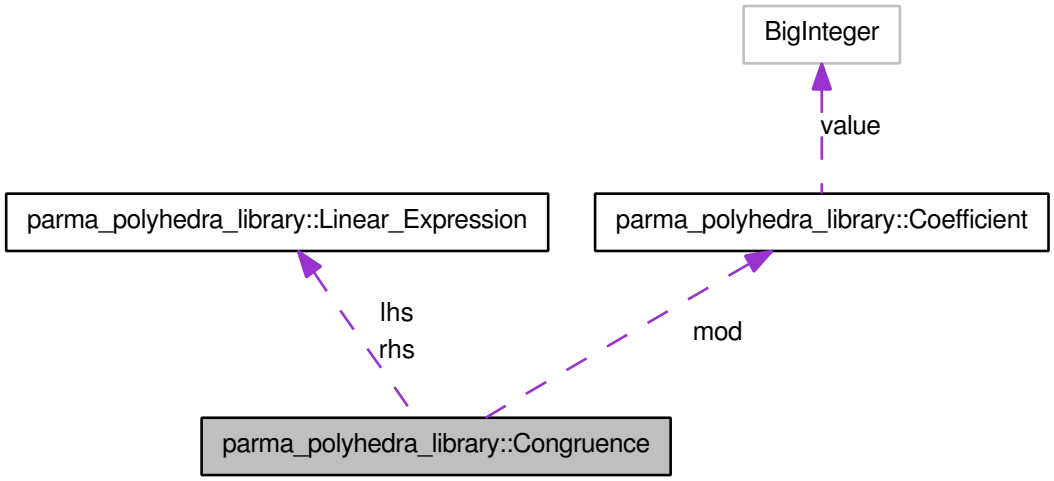
The documentation for this class was generated from the following file:

- Coefficient.java

11.6 parma_polyhedra_library::Congruence Class Reference

A linear congruence.

Collaboration diagram for parma_polyhedra_library::Congruence:



Public Member Functions

- Congruence (Linear_Expression e1, Linear_Expression e2, Coefficient m)

  Returns the congruence `e1` $=$ `e2` $\pmod{\texttt{m}}$.

- Linear_Expression left_hand_side ()

  Returns the left hand side of `this`.

- Linear_Expression right_hand_side ()

  Returns the right hand side of `this`.

- Coefficient modulus ()

  Returns the relation symbol of `this`.

- native String ascii_dump ()

  Returns an ascii formatted internal representation of `this`.

- native String toString ()

  Returns a string representation of `this`.

Protected Attributes

- Coefficient mod

  The modulus of the congruence.

Static Package Functions

- [static initializer]

Package Attributes

- Linear_Expression lhs

  The value of the left hand side of `this`.

- Linear_Expression rhs

  The value of the right hand side of `this`.

Static Private Member Functions

- static native void initIDs ()

11.6.1 Detailed Description

A linear congruence. An object of the class Congruence is an object represeting a congruence:

- $\mathrm{cg} = \sum_{i=0}^{n-1} a_i x_i + b = 0 \pmod{m}$

where n is the dimension of the space, a_i is the integer coefficient of variable x_i, b is the integer inhomogeneous term and m is the integer modulus; if $m = 0$, then cg represents the equality congruence $\sum_{i=0}^{n-1} a_i x_i + b = 0$ and, if $m \neq 0$, then the congruence cg is said to be a proper congruence.

Definition at line 42 of file Congruence.java.

11.6.2 Constructor & Destructor Documentation

11.6.2.1 parma_polyhedra_library::Congruence::Congruence (Linear_Expression *e1*, Linear_Expression *e2*, Coefficient *m*) `[inline]`

Returns the congruence `e1` $=$ `e2` $\pmod{\texttt{m}}$.

Definition at line 57 of file Congruence.java.

References parma_polyhedra_library::Linear_Expression::clone(), lhs, mod, and rhs.

11.6.3 Member Function Documentation

11.6.3.1 parma_polyhedra_library::Congruence::[static initializer] () `[inline, static, package]`

11.6.3.2 native String parma_polyhedra_library::Congruence::ascii_dump ()

Returns an ascii formatted internal representation of `this`.

11.6.3.3 static native void parma_polyhedra_library::Congruence::initIDs () [static, private]

11.6.3.4 Linear_Expression parma_polyhedra_library::Congruence::left_hand_side () [inline]

Returns the left hand side of `this`.

Definition at line 65 of file Congruence.java.

References lhs.

11.6.3.5 Coefficient parma_polyhedra_library::Congruence::modulus () [inline]

Returns the relation symbol of `this`.

Definition at line 75 of file Congruence.java.

References mod.

11.6.3.6 Linear_Expression parma_polyhedra_library::Congruence::right_hand_side () [inline]

Returns the right hand side of `this`.

Definition at line 70 of file Congruence.java.

References rhs.

11.6.3.7 native String parma_polyhedra_library::Congruence::toString ()

Returns a string representation of `this`.

11.6.4 Member Data Documentation

11.6.4.1 Linear_Expression parma_polyhedra_library::Congruence::lhs [package]

The value of the left hand side of `this`.

Definition at line 48 of file Congruence.java.

Referenced by Congruence(), and left_hand_side().

11.6.4.2 Coefficient parma_polyhedra_library::Congruence::mod [protected]

The modulus of the congruence.

Definition at line 45 of file Congruence.java.

Referenced by Congruence(), and modulus().

11.6.4.3 Linear_Expression parma_polyhedra_library::Congruence::rhs [package]

The value of the right hand side of `this`.

Definition at line 51 of file Congruence.java.

Referenced by Congruence(), and right_hand_side().

The documentation for this class was generated from the following file:

- Congruence.java

11.7 parma_polyhedra_library::Congruence_System Class Reference

A system of congruences.

Public Member Functions

- Congruence_System ()
  Default constructor: builds an empty system of congruences.
- native String ascii_dump ()
  Returns an ascii formatted internal representation of `this`.
- native String toString ()
  Returns a string representation of `this`.

Static Package Functions

- [static initializer]

Static Private Member Functions

- static native void initIDs ()

11.7.1 Detailed Description

A system of congruences. An object of the class Congruence_System is a system of congruences, i.e., a multiset of objects of the class Congruence.

Definition at line 34 of file Congruence_System.java.

11.7.2 Constructor & Destructor Documentation

11.7.2.1 parma_polyhedra_library::Congruence_System::Congruence_System () [inline]

Default constructor: builds an empty system of congruences.

Definition at line 36 of file Congruence_System.java.

11.7.3 Member Function Documentation

11.7.3.1 parma_polyhedra_library::Congruence_System::[static initializer] () [inline, static, package]

11.7.3.2 native String parma_polyhedra_library::Congruence_System::ascii_dump ()

Returns an ascii formatted internal representation of `this`.

11.7.3.3 static native void parma_polyhedra_library::Congruence_System::initIDs () [static, private]

11.7.3.4 native String parma_polyhedra_library::Congruence_System::toString ()

Returns a string representation of `this`.

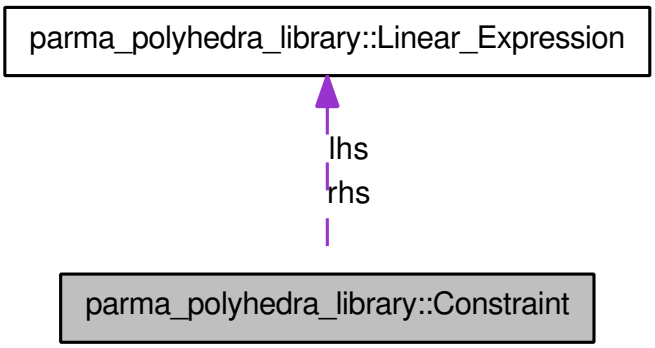
The documentation for this class was generated from the following file:

- Congruence_System.java

11.8 parma_polyhedra_library::Constraint Class Reference

A linear equality or inequality.

Collaboration diagram for parma_polyhedra_library::Constraint:



Public Member Functions

- Constraint (Linear_Expression le1, Relation_Symbol rel_sym, Linear_Expression le2)

  Builds a constraint from two linear expressions with a specified relation symbol.

- Linear_Expression left_hand_side ()

  Returns the left hand side of `this`.

- Linear_Expression right_hand_side ()

  Returns the right hand side of `this`.

- Relation_Symbol kind ()

  Returns the relation symbol of `this`.

- native String ascii_dump ()

  Returns an ascii formatted internal representation of `this`.

- native String toString ()

  Returns a string representation of `this`.

Static Package Functions

- [static initializer]

Static Private Member Functions

- static native void initIDs ()

Private Attributes

- Linear_Expression lhs

  The value of the left hand side of `this`.

- Linear_Expression rhs

  The value of the right hand side of `this`.

- Relation_Symbol kind

  The relation symbol of `this`.

11.8.1 Detailed Description

A linear equality or inequality. An object of the class Constraint is either:

- a linear equality;
- a non-strict linear inequality;
- a strict linear inequality.

Definition at line 36 of file Constraint.java.

11.8.2 Constructor & Destructor Documentation

11.8.2.1 parma_polyhedra_library::Constraint::Constraint (Linear_Expression *le1*, Relation_Symbol *rel_sym*, Linear_Expression *le2*) `[inline]`

Builds a constraint from two linear expressions with a specified relation symbol.

Definition at line 51 of file Constraint.java.

References parma_polyhedra_library::Linear_Expression::clone(), kind(), lhs, and rhs.

11.8.3 Member Function Documentation

11.8.3.1 parma_polyhedra_library::Constraint::[static initializer] () `[inline, static, package]`

11.8.3.2 native String parma_polyhedra_library::Constraint::ascii_dump ()

Returns an ascii formatted internal representation of `this`.

11.8.3.3 static native void parma_polyhedra_library::Constraint::initIDs () `[static, private]`

11.8.3.4 Relation_Symbol parma_polyhedra_library::Constraint::kind () [inline]

Returns the relation symbol of `this`.

Definition at line 69 of file Constraint.java.

Referenced by Constraint().

11.8.3.5 Linear_Expression parma_polyhedra_library::Constraint::left_hand_side () [inline]

Returns the left hand side of `this`.

Definition at line 59 of file Constraint.java.

References lhs.

11.8.3.6 Linear_Expression parma_polyhedra_library::Constraint::right_hand_side () [inline]

Returns the right hand side of `this`.

Definition at line 64 of file Constraint.java.

References rhs.

11.8.3.7 native String parma_polyhedra_library::Constraint::toString ()

Returns a string representation of `this`.

11.8.4 Member Data Documentation

11.8.4.1 Relation_Symbol parma_polyhedra_library::Constraint::kind [private]

The relation symbol of `this`.

Definition at line 45 of file Constraint.java.

11.8.4.2 Linear_Expression parma_polyhedra_library::Constraint::lhs [private]

The value of the left hand side of `this`.

Definition at line 39 of file Constraint.java.

Referenced by Constraint(), and left_hand_side().

11.8.4.3 Linear_Expression parma_polyhedra_library::Constraint::rhs [private]

The value of the right hand side of `this`.

Definition at line 42 of file Constraint.java.

Referenced by Constraint(), and right_hand_side().

The documentation for this class was generated from the following file:

- Constraint.java

11.9 parma_polyhedra_library::Constraint_System Class Reference

A system of constraints.

Public Member Functions

- Constraint_System ()
  Default constructor: builds an empty system of constraints.
- native String ascii_dump ()
  Returns an ascii formatted internal representation of `this`.
- native String toString ()
  Returns a string representation of `this`.

Static Package Functions

- [static initializer]

Static Private Member Functions

- static native void initIDs ()

11.9.1 Detailed Description

A system of constraints. An object of the class Constraint_System is a system of constraints, i.e., a multiset of objects of the class Constraint.

Definition at line 34 of file Constraint_System.java.

11.9.2 Constructor & Destructor Documentation

11.9.2.1 parma_polyhedra_library::Constraint_System::Constraint_System () [inline]

Default constructor: builds an empty system of constraints.

Definition at line 37 of file Constraint_System.java.

11.9.3 Member Function Documentation

11.9.3.1 parma_polyhedra_library::Constraint_System::[static initializer] () [inline, static, package]

11.9.3.2 native String parma_polyhedra_library::Constraint_System::ascii_dump ()

Returns an ascii formatted internal representation of `this`.

11.9.3.3 static native void parma_polyhedra_library::Constraint_System::initIDs () [static, private]

11.9.3.4 native String parma_polyhedra_library::Constraint_System::toString ()

Returns a string representation of `this`.

The documentation for this class was generated from the following file:

- Constraint_System.java

11.10 Parma_Polyhedra_Library::Interfaces::Java::deterministic_timeout_exception Class Reference

`#include <ppl_java_common.defs.hh>`

Public Member Functions

- void throw_me () const
- int priority () const

11.10.1 Detailed Description

Definition at line 133 of file ppl_java_common.defs.hh.

11.10.2 Member Function Documentation

11.10.2.1 int Parma_Polyhedra_Library::Interfaces::Java::deterministic_timeout_exception::priority () const [inline]

Definition at line 139 of file ppl_java_common.defs.hh.

11.10.2.2 void Parma_Polyhedra_Library::Interfaces::Java::deterministic_timeout_exception::throw_me () const [inline]

Definition at line 136 of file ppl_java_common.defs.hh.

The documentation for this class was generated from the following file:

- ppl_java_common.defs.hh

11.11 parma_polyhedra_library::Domain_Error_Exception Class Reference

Exceptions caused by domain errors.

Public Member Functions

- Domain_Error_Exception (String s)

  Constructor.

11.11.1 Detailed Description

Exceptions caused by domain errors.

Definition at line 28 of file Domain_Error_Exception.java.

11.11.2 Constructor & Destructor Documentation

11.11.2.1 parma_polyhedra_library::Domain_Error_Exception::Domain_Error_Exception (String *s*) [inline]

Constructor.

Definition at line 30 of file Domain_Error_Exception.java.

The documentation for this class was generated from the following file:

- Domain_Error_Exception.java

11.12 parma_polyhedra_library::Fake_Class_For_Doxygen Class Reference

11.12.1 Detailed Description

Definition at line 27 of file Fake_Class_for_Doxygen.java.

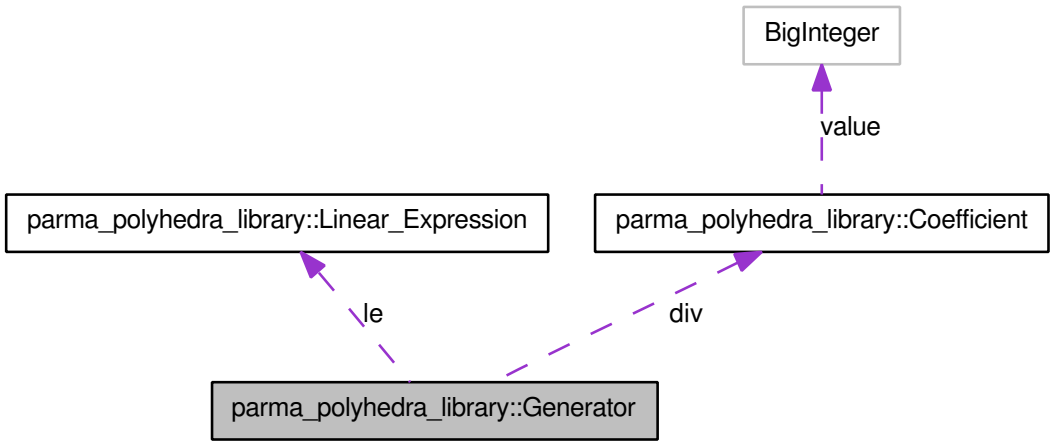
The documentation for this class was generated from the following file:

- Fake_Class_for_Doxygen.java

11.13 parma_polyhedra_library::Generator Class Reference

A line, ray, point or closure point.

Collaboration diagram for parma_polyhedra_library::Generator:



Public Member Functions

- Generator_Type type ()

  Returns the generator type.

- Linear_Expression linear_expression ()

  Returns the linear expression in `this`.

- Coefficient divisor ()

  If `this` *is either a point or a closure point, returns its divisor.*

- native String ascii_dump ()

  Returns an ascii formatted internal representation of `this`.

- native String toString ()

  Returns a string representation of `this`.

Static Public Member Functions

- static Generator closure_point (Linear_Expression e, Coefficient d)

  Returns the closure point at `e` / `d`.

- static Generator line (Linear_Expression e)

  Returns the line of direction `e`.

- static Generator point (Linear_Expression e, Coefficient d)

  Returns the point at `e` / `d`.

- static Generator ray (Linear_Expression e)

  Returns the ray of direction `e`.

Static Package Functions

- [static initializer]

Private Member Functions

- Generator (Linear_Expression e, Generator_Type g_type)

  Builds a generator of type `g_type`*, stealing the coefficients from* `e`.

Static Private Member Functions

- static native void initIDs ()

Private Attributes

- Generator_Type gt

  The generator type.

- Linear_Expression le

  The linear expression.

- Coefficient div

  The divisor (valid if the generator is a point or a closure point).

11.13.1 Detailed Description

A line, ray, point or closure point. An object of the class Generator is one of the following:

- a line;
- a ray;
- a point;
- a closure point.

Definition at line 40 of file Generator.java.

11.13.2 Constructor & Destructor Documentation

11.13.2.1 parma_polyhedra_library::Generator::Generator (Linear_Expression *e*, Generator_Type *g_type*) [inline, private]

Builds a generator of type `g_type`, stealing the coefficients from `e`.

Definition at line 54 of file Generator.java.

References parma_polyhedra_library::Linear_Expression::clone(), gt, and le.

Referenced by closure_point(), line(), point(), and ray().

11.13.3 Member Function Documentation

11.13.3.1 parma_polyhedra_library::Generator::[static initializer] () [inline, static, package]

11.13.3.2 native String parma_polyhedra_library::Generator::ascii_dump ()

Returns an ascii formatted internal representation of `this`.

11.13.3.3 static Generator parma_polyhedra_library::Generator::closure_point (Linear_Expression *e*, Coefficient *d*) [inline, static]

Returns the closure point at `e` / `d`.

Exceptions

RuntimeErrorException Thrown if `d` is zero.

Definition at line 64 of file Generator.java.

References div, Generator(), and parma_polyhedra_library::Coefficient::getBigInteger().

11.13.3.4 Coefficient parma_polyhedra_library::Generator::divisor () [inline]

If `this` is either a point or a closure point, returns its divisor.

Exceptions

RuntimeErrorException Thrown if `this` is neither a point nor a closure point.

Definition at line 129 of file Generator.java.

References div, and gt.

11.13.3.5 static native void parma_polyhedra_library::Generator::initIDs () [static, private]

11.13.3.6 static Generator parma_polyhedra_library::Generator::line (Linear_Expression *e*) [inline, static]

Returns the line of direction `e`.

Exceptions

RuntimeErrorException Thrown if the homogeneous part of `e` represents the origin of the vector space.

Definition at line 82 of file Generator.java.

References Generator().

11.13.3.7 Linear_Expression parma_polyhedra_library::Generator::linear_expression () [inline]

Returns the linear expression in `this`.

Definition at line 120 of file Generator.java.

References le.

11.13.3.8 static Generator parma_polyhedra_library::Generator::point (Linear_Expression *e*, Coefficient *d*) [inline, static]

Returns the point at `e` / `d`.

Exceptions

RuntimeErrorException Thrown if `d` is zero.

Definition at line 91 of file Generator.java.

References div, Generator(), and parma_polyhedra_library::Coefficient::getBigInteger().

11.13.3.9 static Generator parma_polyhedra_library::Generator::ray (Linear_Expression *e*) [inline, static]

Returns the ray of direction `e`.

Exceptions

RuntimeErrorException Thrown if the homogeneous part of `e` represents the origin of the vector space.

Definition at line 110 of file Generator.java.

References Generator().

11.13.3.10 native String parma_polyhedra_library::Generator::toString ()

Returns a string representation of `this`.

11.13.3.11 Generator_Type parma_polyhedra_library::Generator::type () [inline]

Returns the generator type.

Definition at line 115 of file Generator.java.

References gt.

11.13.4 Member Data Documentation

11.13.4.1 Coefficient parma_polyhedra_library::Generator::div [private]

The divisor (valid if the generator is a point or a closure point).

Definition at line 48 of file Generator.java.

Referenced by closure_point(), divisor(), and point().

11.13.4.2 Generator_Type parma_polyhedra_library::Generator::gt [private]

The generator type.

Definition at line 42 of file Generator.java.

Referenced by divisor(), Generator(), and type().

11.13.4.3 Linear_Expression parma_polyhedra_library::Generator::le [private]

The linear expression.

Definition at line 45 of file Generator.java.

Referenced by Generator(), and linear_expression().

The documentation for this class was generated from the following file:

- Generator.java

11.14 parma_polyhedra_library::Generator_System Class Reference

A system of generators.

Public Member Functions

- Generator_System ()

  Default constructor: builds an empty system of generators.

- native String ascii_dump ()

  Returns an ascii formatted internal representation of `this`.

- native String toString ()

  Returns a string representation of `this`.

Static Package Functions

- [static initializer]

Static Private Member Functions

- static native void initIDs ()

11.14.1 Detailed Description

A system of generators. An object of the class Generator_System is a system of generators, i.e., a multiset of objects of the class Generator (lines, rays, points and closure points).

Definition at line 35 of file Generator_System.java.

11.14.2 Constructor & Destructor Documentation

11.14.2.1 parma_polyhedra_library::Generator_System::Generator_System () [inline]

Default constructor: builds an empty system of generators.

Definition at line 38 of file Generator_System.java.

11.14.3 Member Function Documentation

11.14.3.1 parma_polyhedra_library::Generator_System::[static initializer] () [inline, static, package]

11.14.3.2 native String parma_polyhedra_library::Generator_System::ascii_dump ()

Returns an ascii formatted internal representation of `this`.

11.14.3.3 static native void parma_polyhedra_library::Generator_System::initIDs () [static, private]

11.14.3.4 native String parma_polyhedra_library::Generator_System::toString ()

Returns a string representation of `this`.

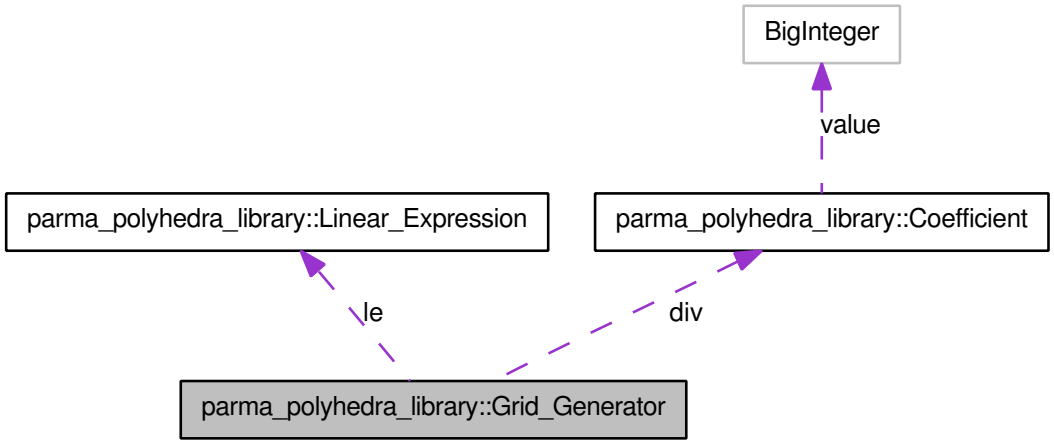
The documentation for this class was generated from the following file:

- Generator_System.java

11.15 parma_polyhedra_library::Grid_Generator Class Reference

A grid line, parameter or grid point.

Collaboration diagram for parma_polyhedra_library::Grid_Generator:



Public Member Functions

- Grid_Generator_Type type ()

  Returns the generator type.

- Linear_Expression linear_expression ()

  Returns the linear expression in `this`.

- Coefficient divisor ()

If `this` is either a grid point or a parameter, returns its divisor.

- native String ascii_dump ()

  Returns an ascii formatted internal representation of `this`.

- native String toString ()

  Returns a string representation of `this`.

Static Public Member Functions

- static Grid_Generator grid_line (Linear_Expression e)

  Returns the line of direction `e`.

- static Grid_Generator parameter (Linear_Expression e, Coefficient d)

  Returns the parameter at `e` / `d`.

- static Grid_Generator grid_point (Linear_Expression e, Coefficient d)

  Returns the point at `e` / `d`.

Static Package Functions

- [static initializer]

Private Member Functions

- Grid_Generator (Linear_Expression e, Coefficient d, Grid_Generator_Type generator_type)

Static Private Member Functions

- static native void initIDs ()

Private Attributes

- Grid_Generator_Type gt

  The grid generator type.

- Linear_Expression le

  The linear expression.

- Coefficient div

  The coefficient used if the grid generator is a parameter or a grid point a parameter.

11.15.1 Detailed Description

A grid line, parameter or grid point. An object of the class Grid_Generator is one of the following:

- a grid_line;
- a parameter;
- a grid_point.

Definition at line 37 of file Grid_Generator.java.

11.15.2 Constructor & Destructor Documentation

11.15.2.1 parma_polyhedra_library::Grid_Generator::Grid_Generator (Linear_Expression *e*, Coefficient *d*, Grid_Generator_Type *generator_type*) [inline, private]

Definition at line 51 of file Grid_Generator.java.

References parma_polyhedra_library::Linear_Expression::clone(), div, gt, and le.

Referenced by grid_line(), grid_point(), and parameter().

11.15.3 Member Function Documentation

11.15.3.1 parma_polyhedra_library::Grid_Generator::[static initializer] () [inline, static, package]

11.15.3.2 native String parma_polyhedra_library::Grid_Generator::ascii_dump ()

Returns an ascii formatted internal representation of `this`.

11.15.3.3 Coefficient parma_polyhedra_library::Grid_Generator::divisor () [inline]

If `this` is either a grid point or a parameter, returns its divisor.

Exceptions

RuntimeErrorException Thrown if `this` is a line.

Definition at line 104 of file Grid_Generator.java.

References div, and gt.

11.15.3.4 static Grid_Generator parma_polyhedra_library::Grid_Generator::grid_line (Linear_Expression *e*) [inline, static]

Returns the line of direction `e`.

Exceptions

RuntimeErrorException Thrown if the homogeneous part of `e` represents the origin of the vector space.

Definition at line 64 of file Grid_Generator.java.

References Grid_Generator().

11.15.3.5 static Grid_Generator parma_polyhedra_library::Grid_Generator::grid_point (Linear_Expression *e*, Coefficient *d*) [inline, static]

Returns the point at `e` / `d`.

Exceptions

RuntimeErrorException Thrown if `d` is zero.

Definition at line 84 of file Grid_Generator.java.

References Grid_Generator().

11.15.3.6 static native void parma_polyhedra_library::Grid_Generator::initIDs () [static, private]

11.15.3.7 Linear_Expression parma_polyhedra_library::Grid_Generator::linear_expression () [inline]

Returns the linear expression in `this`.

Definition at line 95 of file Grid_Generator.java.

References le.

11.15.3.8 static Grid_Generator parma_polyhedra_library::Grid_Generator::parameter (Linear_Expression *e*, Coefficient *d*) [inline, static]

Returns the parameter at `e` / `d`.

Exceptions

RuntimeErrorException Thrown if `d` is zero.

Definition at line 74 of file Grid_Generator.java.

References Grid_Generator().

11.15.3.9 native String parma_polyhedra_library::Grid_Generator::toString ()

Returns a string representation of `this`.

11.15.3.10 Grid_Generator_Type parma_polyhedra_library::Grid_Generator::type () [inline]

Returns the generator type.

Definition at line 90 of file Grid_Generator.java.

References gt.

11.15.4 Member Data Documentation

11.15.4.1 Coefficient parma_polyhedra_library::Grid_Generator::div [private]

The coefficient used if the grid generator is a parameter or a grid point a parameter.

Definition at line 49 of file Grid_Generator.java.

Referenced by divisor(), and Grid_Generator().

11.15.4.2 Grid_Generator_Type parma_polyhedra_library::Grid_Generator::gt [private]

The grid generator type.

Definition at line 40 of file Grid_Generator.java.

Referenced by divisor(), Grid_Generator(), and type().

11.15.4.3 Linear_Expression parma_polyhedra_library::Grid_Generator::le [private]

The linear expression.

Definition at line 43 of file Grid_Generator.java.

Referenced by Grid_Generator(), and linear_expression().

The documentation for this class was generated from the following file:

- Grid_Generator.java

11.16 parma_polyhedra_library::Grid_Generator_System Class Reference

A system of grid generators.

Public Member Functions

- Grid_Generator_System ()

  Default constructor: builds an empty system of grid generators.

- native String ascii_dump ()

  Returns an ascii formatted internal representation of `this`.

- native String toString ()

  Returns a string representation of `this`.

Static Package Functions

- [static initializer]

Static Private Member Functions

- static native void initIDs ()

11.16.1 Detailed Description

A system of grid generators. An object of the class Grid_Generator_System is a system of grid generators, i.e., a multiset of objects of the class Grid_Generator.

Definition at line 35 of file Grid_Generator_System.java.

11.16.2 Constructor & Destructor Documentation

11.16.2.1 parma_polyhedra_library::Grid_Generator_System::Grid_Generator_System () [inline]

Default constructor: builds an empty system of grid generators.

Definition at line 38 of file Grid_Generator_System.java.

11.16.3 Member Function Documentation

11.16.3.1 parma_polyhedra_library::Grid_Generator_System::[static initializer] () [inline, static, package]

11.16.3.2 native String parma_polyhedra_library::Grid_Generator_System::ascii_dump ()

Returns an ascii formatted internal representation of `this`.

11.16.3.3 static native void parma_polyhedra_library::Grid_Generator_System::initIDs () [static, private]

11.16.3.4 native String parma_polyhedra_library::Grid_Generator_System::toString ()

Returns a string representation of `this`.

The documentation for this class was generated from the following file:

- Grid_Generator_System.java

11.17 parma_polyhedra_library::Invalid_Argument_Exception Class Reference

Exceptions caused by invalid arguments.

Public Member Functions

- Invalid_Argument_Exception (String s)

  Constructor.

11.17.1 Detailed Description

Exceptions caused by invalid arguments.

Definition at line 29 of file Invalid_Argument_Exception.java.

11.17.2 Constructor & Destructor Documentation

11.17.2.1 parma_polyhedra_library::Invalid_Argument_Exception::Invalid_Argument_Exception (String *s*) [inline]

Constructor.

Definition at line 31 of file Invalid_Argument_Exception.java.

The documentation for this class was generated from the following file:

- Invalid_Argument_Exception.java

11.18 parma_polyhedra_library::IO Class Reference

A class collecting I/O functions.

Static Public Member Functions

- static native String wrap_string (String str, int indent_depth, int preferred_first_line_length, int preferred_line_length)

  Utility function for the wrapping of lines of text.

11.18.1 Detailed Description

A class collecting I/O functions.

Definition at line 28 of file IO.java.

11.18.2 Member Function Documentation

11.18.2.1 static native String parma_polyhedra_library::IO::wrap_string (String *str*, int *indent_depth*, int *preferred_first_line_length*, int *preferred_line_length*) [static]

Utility function for the wrapping of lines of text.

Parameters

str The source string holding the lines to wrap.

indent_depth The indentation depth.

preferred_first_line_length The preferred length for the first line of text.

preferred_line_length The preferred length for all the lines but the first one.

Returns

The wrapped string.

The documentation for this class was generated from the following file:

- IO.java

11.19 Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache Class Reference

A cache for global references to Java classes.

`#include <ppl_java_common.defs.hh>`

Public Member Functions

- Java_Class_Cache ()

  Default constructor.

- void init_cache (JNIEnv *env)

  Initializes all cache fields.

- void clear_cache (JNIEnv *env)

  Resets all fields to `NULL`.

Public Attributes

- jclass Boolean
- jclass Integer
- jclass Long
- jclass Iterator
- jclass Artificial_Parameter
- jclass Artificial_Parameter_Sequence
- jclass Bounded_Integer_Type_Overflow
- jclass Bounded_Integer_Type_Representation
- jclass Bounded_Integer_Type_Width
- jclass By_Reference
- jclass Coefficient
- jclass Congruence
- jclass Constraint
- jclass Generator
- jclass Grid_Generator
- jclass Generator_Type
- jclass Grid_Generator_Type
- jclass Constraint_System
- jclass Congruence_System
- jclass Generator_System
- jclass Grid_Generator_System
- jclass Linear_Expression
- jclass Linear_Expression_Coefficient
- jclass Linear_Expression_Difference
- jclass Linear_Expression_Sum
- jclass Linear_Expression_Times
- jclass Linear_Expression_Unary_Minus
- jclass Linear_Expression_Variable
- jclass MIP_Problem_Status
- jclass Optimization_Mode
- jclass Pair
- jclass PIP_Problem_Control_Parameter_Name
- jclass PIP_Problem_Control_Parameter_Value
- jclass PIP_Problem_Status
- jclass Poly_Con_Relation
- jclass Poly_Gen_Relation

- jclass PPL_Object
- jclass Relation_Symbol
- jclass Variable
- jclass Variables_Set

Private Member Functions

- void init_cache (JNIEnv *env, jclass &field, const char *name)

  Sets `field` to a global reference to Java class called `name`.

- void clear_cache (JNIEnv *env, jclass &field)

  Resets `field` to `NULL`, deleting the global reference (if any).

- Java_Class_Cache (const Java_Class_Cache &)
- Java_Class_Cache & operator= (const Java_Class_Cache &)

11.19.1 Detailed Description

A cache for global references to Java classes. The cache is loaded by `Parma_Polyhedra_Library.initialize_library();` It is cleared by `Parma_Polyhedra_Library.finalize_library()`.

Definition at line 193 of file ppl_java_common.defs.hh.

11.19.2 Constructor & Destructor Documentation

11.19.2.1 Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Java_Class_Cache ()

Default constructor.

Definition at line 31 of file ppl_java_common.cc.

References Artificial_Parameter, Artificial_Parameter_Sequence, Boolean, Bounded_Integer_Type_Overflow, Bounded_Integer_Type_Representation, Bounded_Integer_Type_Width, By_Reference, Coefficient, Congruence, Congruence_System, Constraint, Constraint_System, Generator, Generator_System, Generator_Type, Grid_Generator, Grid_Generator_System, Grid_Generator_Type, Integer, Iterator, Linear_Expression, Linear_Expression_Coefficient, Linear_Expression_Difference, Linear_Expression_Sum, Linear_Expression_Times, Linear_Expression_Unary_Minus, Linear_Expression_Variable, Long, MIP_Problem_Status, Optimization_Mode, Pair, PIP_Problem_Control_Parameter_Name, PIP_Problem_Control_Parameter_Value, PIP_Problem_Status, Poly_Con_Relation, Poly_Gen_Relation, PPL_Object, and Relation_Symbol.

11.19.2.2 Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Java_Class_Cache (const Java_Class_Cache &) [private]

11.19.3 Member Function Documentation

11.19.3.1 void Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::clear_cache (JNIEnv * *env*, jclass & *field*) [private]

Resets `field` to `NULL`, deleting the global reference (if any).

Definition at line 157 of file ppl_java_common.cc.

11.19.3.2 void Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::clear_cache (JNIEnv * *env*)

Resets all fields to `NULL`.

Definition at line 166 of file ppl_java_common.cc.

Referenced by Java_parma_1polyhedra_1library_Parma_1Polyhedra_1Library_finalize_1library().

11.19.3.3 void Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::init_cache (JNIEnv * *env*, jclass & *field*, const char * *name*) [private]

Sets `field` to a global reference to Java class called `name`.

Definition at line 77 of file ppl_java_common.cc.

11.19.3.4 void Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::init_cache (JNIEnv * *env*)

Initializes all cache fields.

Definition at line 89 of file ppl_java_common.cc.

Referenced by Java_parma_1polyhedra_1library_Parma_1Polyhedra_1Library_initialize_1library().

11.19.3.5 Java_Class_Cache& Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::operator= (const Java_Class_Cache &) [private]

11.19.4 Member Data Documentation

11.19.4.1 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Artificial_Parameter

Definition at line 201 of file ppl_java_common.defs.hh.

Referenced by Java_Class_Cache().

11.19.4.2 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Artificial_Parameter_Sequence

Definition at line 202 of file ppl_java_common.defs.hh.

Referenced by Java_Class_Cache(), and Java_parma_1polyhedra_1library_PIP_1Tree_1Node_artificials().

11.19.4.3 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Boolean

Definition at line 196 of file ppl_java_common.defs.hh.

Referenced by Java_Class_Cache(), and Java_parma_1polyhedra_1library_Coefficient_initIDs().

11.19.4.4 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Bounded_Integer_Type_Overflow

Definition at line 203 of file ppl_java_common.defs.hh.

Referenced by Java_Class_Cache().

11.19.4.5 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Bounded_Integer_Type_Representation

Definition at line 204 of file ppl_java_common.defs.hh.

Referenced by Java_Class_Cache().

11.19.4.6 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Bounded_Integer_Type_Width

Definition at line 205 of file ppl_java_common.defs.hh.

Referenced by Java_Class_Cache().

11.19.4.7 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::By_Reference

Definition at line 206 of file ppl_java_common.defs.hh.

Referenced by Java_Class_Cache().

11.19.4.8 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Coefficient

Definition at line 207 of file ppl_java_common.defs.hh.

Referenced by Parma_Polyhedra_Library::Interfaces::Java::build_java_coeff(), and Java_Class_Cache().

11.19.4.9 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Congruence

Definition at line 208 of file ppl_java_common.defs.hh.

Referenced by Java_Class_Cache().

11.19.4.10 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Congruence_System

Definition at line 215 of file ppl_java_common.defs.hh.

Referenced by Java_Class_Cache().

11.19.4.11 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Constraint

Definition at line 209 of file ppl_java_common.defs.hh.

Referenced by Java_Class_Cache().

11.19.4.12 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Constraint_System

Definition at line 214 of file ppl_java_common.defs.hh.

Referenced by Java_Class_Cache(), Java_parma_1polyhedra_1library_MIP_1Problem_constraints(), Java_parma_1polyhedra_1library_PIP_1Problem_constraints(), and Java_parma_1polyhedra_1library_PIP_1Tree_1Node_constraints().

11.19.4.13 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Generator

Definition at line 210 of file ppl_java_common.defs.hh.

Referenced by Java_Class_Cache().

11.19.4.14 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Generator_System

Definition at line 216 of file ppl_java_common.defs.hh.

Referenced by Java_Class_Cache().

11.19.4.15 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Generator_Type

Definition at line 212 of file ppl_java_common.defs.hh.

Referenced by Java_Class_Cache().

11.19.4.16 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Grid_Generator

Definition at line 211 of file ppl_java_common.defs.hh.

Referenced by Java_Class_Cache().

11.19.4.17 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Grid_Generator_System

Definition at line 217 of file ppl_java_common.defs.hh.

Referenced by Java_Class_Cache().

11.19.4.18 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Grid_Generator_Type

Definition at line 213 of file ppl_java_common.defs.hh.

Referenced by Java_Class_Cache().

11.19.4.19 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Integer

Definition at line 197 of file ppl_java_common.defs.hh.

Referenced by Java_Class_Cache(), and Java_parma_1polyhedra_1library_Coefficient_initIDs().

11.19.4.20 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Iterator

Definition at line 199 of file ppl_java_common.defs.hh.

Referenced by Java_Class_Cache(), and Java_parma_1polyhedra_1library_Constraint_1System_initIDs().

11.19.4.21 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Linear_-Expression

Definition at line 218 of file ppl_java_common.defs.hh.

Referenced by Java_Class_Cache().

11.19.4.22 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Linear_-Expression_Coefficient

Definition at line 219 of file ppl_java_common.defs.hh.

Referenced by Parma_Polyhedra_Library::Interfaces::Java::build_linear_expression(), Java_Class_-Cache(), and Java_parma_1polyhedra_1library_MIP_1Problem_objective_1function().

11.19.4.23 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Linear_-Expression_Difference

Definition at line 220 of file ppl_java_common.defs.hh.

Referenced by Java_Class_Cache().

11.19.4.24 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Linear_-Expression_Sum

Definition at line 221 of file ppl_java_common.defs.hh.

Referenced by Java_Class_Cache().

11.19.4.25 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Linear_-Expression_Times

Definition at line 222 of file ppl_java_common.defs.hh.

Referenced by Parma_Polyhedra_Library::Interfaces::Java::build_linear_expression(), and Java_Class_-Cache().

11.19.4.26 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Linear_-Expression_Unary_Minus

Definition at line 223 of file ppl_java_common.defs.hh.

Referenced by Java_Class_Cache().

11.19.4.27 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Linear_-Expression_Variable

Definition at line 224 of file ppl_java_common.defs.hh.

Referenced by Java_Class_Cache().

11.19.4.28 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Long

Definition at line 198 of file ppl_java_common.defs.hh.

Referenced by Java_Class_Cache(), and Java_parma_1polyhedra_1library_Coefficient_initIDs().

11.19.4.29 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::MIP_-Problem_Status

Definition at line 225 of file ppl_java_common.defs.hh.

Referenced by Java_Class_Cache().

11.19.4.30 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_-Cache::Optimization_Mode

Definition at line 226 of file ppl_java_common.defs.hh.

Referenced by Java_Class_Cache().

11.19.4.31 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Pair

Definition at line 227 of file ppl_java_common.defs.hh.

Referenced by Java_Class_Cache().

11.19.4.32 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::PIP_-Problem_Control_Parameter_Name

Definition at line 228 of file ppl_java_common.defs.hh.

Referenced by Java_Class_Cache().

11.19.4.33 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::PIP_-Problem_Control_Parameter_Value

Definition at line 229 of file ppl_java_common.defs.hh.

Referenced by Java_Class_Cache().

11.19.4.34 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::PIP_-Problem_Status

Definition at line 230 of file ppl_java_common.defs.hh.

Referenced by Java_Class_Cache().

11.19.4.35 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Poly_Con_-Relation

Definition at line 231 of file ppl_java_common.defs.hh.

Referenced by Java_Class_Cache().

11.19.4.36 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Poly_Gen_-Relation

Definition at line 232 of file ppl_java_common.defs.hh.

Referenced by Java_Class_Cache().

11.19.4.37 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::PPL_Object

Definition at line 233 of file ppl_java_common.defs.hh.

Referenced by Java_Class_Cache().

11.19.4.38 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Relation_-Symbol

Definition at line 234 of file ppl_java_common.defs.hh.

Referenced by Java_Class_Cache().

11.19.4.39 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Variable

Definition at line 235 of file ppl_java_common.defs.hh.

Referenced by Parma_Polyhedra_Library::Interfaces::Java::build_java_variable().

11.19.4.40 jclass Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Variables_Set

Definition at line 236 of file ppl_java_common.defs.hh.

The documentation for this class was generated from the following files:

- ppl_java_common.defs.hh
- ppl_java_common.cc

11.20 Parma_Polyhedra_Library::Interfaces::Java::Java_ExceptionOccurred Struct Reference

`#include <ppl_java_common.defs.hh>`

11.20.1 Detailed Description

Definition at line 119 of file ppl_java_common.defs.hh.

The documentation for this struct was generated from the following file:

- ppl_java_common.defs.hh

11.21 Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache Struct Reference

A cache for field and method IDs of Java classes.

`#include <ppl_java_common.defs.hh>`

Public Attributes

- jmethodID Boolean_valueOf_ID
- jmethodID Boolean_boolValue_ID
- jmethodID Integer_valueOf_ID
- jmethodID Integer_intValue_ID
- jmethodID Long_valueOf_ID
- jmethodID Long_longValue_ID
- jfieldID Artificial_Parameter_le_ID
- jfieldID Artificial_Parameter_den_ID
- jmethodID Artificial_Parameter_init_ID
- jmethodID Artificial_Parameter_Sequence_init_ID
- jmethodID Artificial_Parameter_Sequence_add_ID
- jfieldID Bounded_Integer_Type_Overflow_OVERFLOW_WRAPS_ID
- jfieldID Bounded_Integer_Type_Overflow_OVERFLOW_UNDEFINED_ID
- jfieldID Bounded_Integer_Type_Overflow_OVERFLOW_IMPOSSIBLE_ID
- jmethodID Bounded_Integer_Type_Overflow_ordinal_ID
- jfieldID Bounded_Integer_Type_Representation_UNSIGNED_ID
- jfieldID Bounded_Integer_Type_Representation_SIGNED_2_COMPLEMENT_ID
- jmethodID Bounded_Integer_Type_Representation_ordinal_ID

- jfieldID Bounded_Integer_Type_Width_BITS_8_ID
- jfieldID Bounded_Integer_Type_Width_BITS_16_ID
- jfieldID Bounded_Integer_Type_Width_BITS_32_ID
- jfieldID Bounded_Integer_Type_Width_BITS_64_ID
- jfieldID Bounded_Integer_Type_Width_BITS_128_ID
- jmethodID Bounded_Integer_Type_Width_ordinal_ID
- jfieldID By_Reference_obj_ID
- jmethodID By_Reference_init_ID
- jfieldID Coefficient_value_ID
- jmethodID Coefficient_init_from_String_ID
- jmethodID Coefficient_toString_ID
- jmethodID Complexity_Class_ordinal_ID
- jfieldID Congruence_mod_ID
- jfieldID Congruence_lhs_ID
- jfieldID Congruence_rhs_ID
- jmethodID Congruence_init_ID
- jfieldID Constraint_lhs_ID
- jfieldID Constraint_rhs_ID
- jfieldID Constraint_kind_ID
- jmethodID Constraint_init_ID
- jmethodID Degenerate_Element_ordinal_ID
- jfieldID Generator_gt_ID
- jfieldID Generator_le_ID
- jfieldID Generator_div_ID
- jmethodID Generator_line_ID
- jmethodID Generator_ray_ID
- jmethodID Generator_point_ID
- jmethodID Generator_closure_point_ID
- jfieldID Grid_Generator_gt_ID
- jfieldID Grid_Generator_le_ID
- jfieldID Grid_Generator_div_ID
- jmethodID Grid_Generator_grid_line_ID
- jmethodID Grid_Generator_parameter_ID
- jmethodID Grid_Generator_grid_point_ID
- jmethodID Generator_Type_ordinal_ID
- jmethodID Grid_Generator_Type_ordinal_ID
- jmethodID Constraint_System_init_ID
- jmethodID Constraint_System_add_ID
- jmethodID Congruence_System_init_ID
- jmethodID Congruence_System_add_ID
- jmethodID Generator_System_init_ID
- jmethodID Generator_System_add_ID
- jmethodID Grid_Generator_System_init_ID
- jmethodID Grid_Generator_System_add_ID
- jmethodID System_iterator_ID
- jmethodID System_Iterator_has_next_ID
- jmethodID System_Iterator_next_ID
- jmethodID Linear_Expression_sum_ID
- jmethodID Linear_Expression_times_ID
- jfieldID Linear_Expression_Coefficient_coeff_ID

11.21.1 Detailed Description

A cache for field and method IDs of Java classes. The IDs for fields and methods of PPL Java classes are automatically by the static initializer of the corresponding Java class. The static initializers of some PPL Java class also stores the IDs for fields and methods of non-PPL classes (e.g., Boolean, Long, etc.).

Definition at line 267 of file ppl_java_common.defs.hh.

11.21.2 Member Data Documentation

11.21.2.1 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Artificial_Parameter_den_ID

Definition at line 279 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Artificial_1Parameter_initIDs().

11.21.2.2 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Artificial_Parameter_init_ID

Definition at line 280 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Artificial_1Parameter_initIDs().

11.21.2.3 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Artificial_Parameter_le_ID

Definition at line 278 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Artificial_1Parameter_initIDs().

11.21.2.4 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Artificial_Parameter_Sequence_add_ID

Definition at line 282 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Artificial_1Parameter_1Sequence_initIDs(), and Java_parma_1polyhedra_1library_PIP_1Tree_1Node_artificials().

11.21.2.5 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Artificial_Parameter_Sequence_init_ID

Definition at line 281 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Artificial_1Parameter_1Sequence_initIDs(), and Java_parma_1polyhedra_1library_PIP_1Tree_1Node_artificials().

11.21.2.6 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Boolean_boolValue_ID

Definition at line 270 of file ppl_java_common.defs.hh.

11.21.2.7 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Boolean_valueOf_ID

Definition at line 269 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Coefficient_initIDs().

11.21.2.8 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Bounded_Integer_Type_Overflow_ordinal_ID

Definition at line 287 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Bounded_1Integer_1Type_1Overflow_initIDs().

11.21.2.9 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Bounded_-Integer_Type_Overflow_OVERFLOW_IMPOSSIBLE_ID

Definition at line 286 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Bounded_1Integer_1Type_1Overflow_initIDs().

11.21.2.10 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Bounded_-Integer_Type_Overflow_OVERFLOW_UNDEFINED_ID

Definition at line 285 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Bounded_1Integer_1Type_1Overflow_initIDs().

11.21.2.11 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Bounded_-Integer_Type_Overflow_OVERFLOW_WRAPS_ID

Definition at line 284 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Bounded_1Integer_1Type_1Overflow_initIDs().

11.21.2.12 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Bounded_Integer_Type_Representation_ordinal_ID

Definition at line 291 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Bounded_1Integer_1Type_1Representation_initIDs().

11.21.2.13 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Bounded_-Integer_Type_Representation_SIGNED_2_COMPLEMENT_ID

Definition at line 290 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Bounded_1Integer_1Type_1Representation_initIDs().

11.21.2.14 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Bounded_-Integer_Type_Representation_UNSIGNED_ID

Definition at line 289 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Bounded_1Integer_1Type_1Representation_initIDs().

11.21.2.15 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Bounded_-Integer_Type_Width_BITS_128_ID

Definition at line 297 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Bounded_1Integer_1Type_1Width_initIDs().

11.21.2.16 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Bounded_-Integer_Type_Width_BITS_16_ID

Definition at line 294 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Bounded_1Integer_1Type_1Width_initIDs().

11.21.2.17 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Bounded_-Integer_Type_Width_BITS_32_ID

Definition at line 295 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Bounded_1Integer_1Type_1Width_initIDs().

11.21.2.18 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Bounded_Integer_Type_Width_BITS_64_ID

Definition at line 296 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Bounded_1Integer_1Type_1Width_initIDs().

11.21.2.19 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Bounded_Integer_Type_Width_BITS_8_ID

Definition at line 293 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Bounded_1Integer_1Type_1Width_initIDs().

11.21.2.20 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Bounded_Integer_Type_Width_ordinal_ID

Definition at line 298 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Bounded_1Integer_1Type_1Width_initIDs().

11.21.2.21 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::By_Reference_init_ID

Definition at line 301 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_By_1Reference_initIDs().

11.21.2.22 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::By_Reference_obj_ID

Definition at line 300 of file ppl_java_common.defs.hh.

Referenced by Parma_Polyhedra_Library::Interfaces::Java::get_by_reference(), Java_parma_1polyhedra_1library_By_1Reference_initIDs(), and Parma_Polyhedra_Library::Interfaces::Java::set_by_reference().

11.21.2.23 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Coefficient_init_from_String_ID

Definition at line 304 of file ppl_java_common.defs.hh.

Referenced by Parma_Polyhedra_Library::Interfaces::Java::build_java_coeff(), and Java_parma_1polyhedra_1library_Coefficient_initIDs().

11.21.2.24 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Coefficient_toString_ID

Definition at line 305 of file ppl_java_common.defs.hh.

Referenced by Parma_Polyhedra_Library::Interfaces::Java::build_cxx_coeff(), and Java_parma_-1polyhedra_1library_Coefficient_initIDs().

11.21.2.25 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Coefficient_value_ID

Definition at line 303 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Coefficient_initIDs(), and Parma_Polyhedra_-Library::Interfaces::Java::set_coefficient().

11.21.2.26 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Complexity_Class_ordinal_ID

Definition at line 307 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Complexity_1Class_initIDs().

11.21.2.27 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Congruence_init_ID

Definition at line 312 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Congruence_initIDs().

11.21.2.28 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Congruence_lhs_ID

Definition at line 310 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Congruence_initIDs().

11.21.2.29 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Congruence_mod_ID

Definition at line 309 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Congruence_initIDs().

11.21.2.30 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Congruence_rhs_ID

Definition at line 311 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Congruence_initIDs().

11.21.2.31 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Congruence_System_add_ID

Definition at line 342 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Congruence_1System_initIDs().

11.21.2.32 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Congruence_System_init_ID

Definition at line 341 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Congruence_1System_initIDs().

11.21.2.33 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Constraint_init_ID

Definition at line 317 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Constraint_initIDs().

11.21.2.34 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Constraint_kind_ID

Definition at line 316 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Constraint_initIDs().

11.21.2.35 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Constraint_lhs_ID

Definition at line 314 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Constraint_initIDs().

11.21.2.36 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Constraint_rhs_ID

Definition at line 315 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Constraint_initIDs().

11.21.2.37 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Constraint_System_add_ID

Definition at line 340 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Constraint_1System_initIDs(), Java_parma_-1polyhedra_1library_MIP_1Problem_constraints(), and Java_parma_1polyhedra_1library_PIP_-1Problem_constraints().

11.21.2.38 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Constraint_System_init_ID

Definition at line 339 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Constraint_1System_initIDs(), Java_parma_-1polyhedra_1library_MIP_1Problem_constraints(), Java_parma_1polyhedra_1library_PIP_1Problem_-constraints(), and Java_parma_1polyhedra_1library_PIP_1Tree_1Node_constraints().

11.21.2.39 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Degenerate_Element_ordinal_ID

Definition at line 319 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Degenerate_1Element_initIDs().

11.21.2.40 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Generator_closure_point_ID

Definition at line 327 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Generator_initIDs().

11.21.2.41 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Generator_div_ID

Definition at line 323 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Generator_initIDs().

11.21.2.42 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Generator_gt_ID

Definition at line 321 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Generator_initIDs().

11.21.2.43 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Generator_le_ID

Definition at line 322 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Generator_initIDs().

11.21.2.44 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Generator_line_ID

Definition at line 324 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Generator_initIDs().

11.21.2.45 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Generator_point_ID

Definition at line 326 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Generator_initIDs().

11.21.2.46 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Generator_ray_ID

Definition at line 325 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Generator_initIDs().

11.21.2.47 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Generator_System_add_ID

Definition at line 344 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Generator_1System_initIDs().

11.21.2.48 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Generator_System_init_ID

Definition at line 343 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Generator_1System_initIDs().

11.21.2.49 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Generator_Type_ordinal_ID

Definition at line 336 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Generator_1Type_initIDs().

11.21.2.50 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Grid_-Generator_div_ID

Definition at line 331 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Grid_1Generator_initIDs().

11.21.2.51 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Grid_-Generator_grid_line_ID

Definition at line 332 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Grid_1Generator_initIDs().

11.21.2.52 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Grid_-Generator_grid_point_ID

Definition at line 334 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Grid_1Generator_initIDs().

11.21.2.53 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Grid_-Generator_gt_ID

Definition at line 329 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Grid_1Generator_initIDs().

11.21.2.54 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Grid_Generator_le_ID

Definition at line 330 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Grid_1Generator_initIDs().

11.21.2.55 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Grid_Generator_parameter_ID

Definition at line 333 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Grid_1Generator_initIDs().

11.21.2.56 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Grid_Generator_System_add_ID

Definition at line 346 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Grid_1Generator_1System_initIDs().

11.21.2.57 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Grid_Generator_System_init_ID

Definition at line 345 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Grid_1Generator_1System_initIDs().

11.21.2.58 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Grid_Generator_Type_ordinal_ID

Definition at line 337 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Grid_1Generator_1Type_initIDs().

11.21.2.59 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Integer_intValue_ID

Definition at line 272 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Coefficient_initIDs().

11.21.2.60 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Integer_valueOf_ID

Definition at line 271 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Coefficient_initIDs().

11.21.2.61 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Linear_-Expression_Coefficient_coeff_ID

Definition at line 355 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Linear_1Expression_1Coefficient_initIDs().

11.21.2.62 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Linear_Expression_Coefficient_init_ID

Definition at line 356 of file ppl_java_common.defs.hh.

Referenced by Parma_Polyhedra_Library::Interfaces::Java::build_linear_expression(), Java_parma_-1polyhedra_1library_Linear_1Expression_1Coefficient_initIDs(), and Java_parma_1polyhedra_1library_-MIP_1Problem_objective_1function().

11.21.2.63 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Linear_-Expression_Difference_lhs_ID

Definition at line 357 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Linear_1Expression_1Difference_initIDs().

11.21.2.64 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Linear_-Expression_Difference_rhs_ID

Definition at line 358 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Linear_1Expression_1Difference_initIDs().

11.21.2.65 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Linear_Expression_sum_ID

Definition at line 352 of file ppl_java_common.defs.hh.

Referenced by Parma_Polyhedra_Library::Interfaces::Java::build_linear_expression(), Java_parma_-1polyhedra_1library_Linear_1Expression_initIDs(), and Java_parma_1polyhedra_1library_MIP_-1Problem_objective_1function().

11.21.2.66 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Linear_-Expression_Sum_lhs_ID

Definition at line 359 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Linear_1Expression_1Sum_initIDs().

11.21.2.67 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Linear_-Expression_Sum_rhs_ID

Definition at line 360 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Linear_1Expression_1Sum_initIDs().

11.21.2.68 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Linear_-Expression_Times_coeff_ID

Definition at line 361 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Linear_1Expression_1Times_initIDs().

11.21.2.69 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Linear_Expression_times_ID

Definition at line 353 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Linear_1Expression_initIDs().

11.21.2.70 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Linear_Expression_Times_init_from_coeff_var_ID

Definition at line 363 of file ppl_java_common.defs.hh.

Referenced by Parma_Polyhedra_Library::Interfaces::Java::build_linear_expression(), and Java_parma_-1polyhedra_1library_Linear_1Expression_1Times_initIDs().

11.21.2.71 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Linear_-Expression_Times_lin_expr_ID

Definition at line 362 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Linear_1Expression_1Times_initIDs().

11.21.2.72 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Linear_Expression_Unary_Minus_arg_ID

Definition at line 364 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Linear_1Expression_1Unary_1Minus_initIDs().

11.21.2.73 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Linear_Expression_Variable_init_ID

Definition at line 365 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Linear_1Expression_1Variable_initIDs().

11.21.2.74 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Linear_Expression_Variable_var_id_ID

Definition at line 366 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Linear_1Expression_1Variable_initIDs().

11.21.2.75 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Long_longValue_ID

Definition at line 274 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Coefficient_initIDs().

11.21.2.76 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Long_valueOf_ID

Definition at line 273 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Coefficient_initIDs().

11.21.2.77 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::MIP_Problem_Status_OPTIMIZED_MIP_PROBLEM_ID

Definition at line 370 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_MIP_1Problem_1Status_initIDs().

11.21.2.78 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::MIP_-Problem_Status_ordinal_ID

Definition at line 371 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_MIP_1Problem_1Status_initIDs().

11.21.2.79 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::MIP_-Problem_Status_UNBOUNDED_MIP_PROBLEM_ID

Definition at line 369 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_MIP_1Problem_1Status_initIDs().

11.21.2.80 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::MIP_-Problem_Status_UNFEASIBLE_MIP_PROBLEM_ID

Definition at line 368 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_MIP_1Problem_1Status_initIDs().

11.21.2.81 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Optimization_Mode_MAXIMIZATION_ID

Definition at line 373 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Optimization_1Mode_initIDs().

11.21.2.82 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Optimization_Mode_MINIMIZATION_ID

Definition at line 374 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Optimization_1Mode_initIDs().

11.21.2.83 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Optimization_Mode_ordinal_ID

Definition at line 375 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Optimization_1Mode_initIDs().

11.21.2.84 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Pair_first_-ID

Definition at line 392 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Pair_initIDs().

11.21.2.85 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Pair_-second_ID

Definition at line 393 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Pair_initIDs().

11.21.2.86 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::PIP_-Problem_Control_Parameter_Name_CUTTING_STRATEGY_ID

Definition at line 377 of file ppl_java_common.defs.hh.

11.21.2.87 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::PIP_-Problem_Control_Parameter_Name_ordinal_ID

Definition at line 379 of file ppl_java_common.defs.hh.

11.21.2.88 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::PIP_-Problem_Control_Parameter_Name_PIVOT_ROW_STRATEGY

Definition at line 378 of file ppl_java_common.defs.hh.

11.21.2.89 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::PIP_-Problem_Control_Parameter_Value_CUTTING_STRATEGY_ALL_ID

Definition at line 383 of file ppl_java_common.defs.hh.

11.21.2.90 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::PIP_-Problem_Control_Parameter_Value_CUTTING_STRATEGY_DEEPEST_ID

Definition at line 382 of file ppl_java_common.defs.hh.

11.21.2.91 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::PIP_-Problem_Control_Parameter_Value_CUTTING_STRATEGY_FIRST_ID

Definition at line 381 of file ppl_java_common.defs.hh.

11.21.2.92 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::PIP_-Problem_Control_Parameter_Value_ordinal_ID

Definition at line 386 of file ppl_java_common.defs.hh.

11.21.2.93 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::PIP_-Problem_Control_Parameter_Value_PIVOT_ROW_STRATEGY_FIRST_ID

Definition at line 384 of file ppl_java_common.defs.hh.

11.21.2.94 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::PIP_-Problem_Control_Parameter_Value_PIVOT_ROW_STRATEGY_MAX_COLUMN_-ID

Definition at line 385 of file ppl_java_common.defs.hh.

11.21.2.95 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::PIP_-Problem_Status_OPTIMIZED_PIP_PROBLEM_ID

Definition at line 389 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_PIP_1Problem_1Status_initIDs().

11.21.2.96 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::PIP_-Problem_Status_ordinal_ID

Definition at line 390 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_PIP_1Problem_1Status_initIDs().

11.21.2.97 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::PIP_-Problem_Status_UNFEASIBLE_PIP_PROBLEM_ID

Definition at line 388 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_PIP_1Problem_1Status_initIDs().

11.21.2.98 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Poly_Con_Relation_init_ID

Definition at line 395 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Poly_1Con_1Relation_initIDs().

11.21.2.99 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Poly_Gen_Relation_init_ID

Definition at line 396 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Poly_1Gen_1Relation_initIDs().

11.21.2.100 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::PPL_Object_ptr_ID

Definition at line 398 of file ppl_java_common.defs.hh.

Referenced by Parma_Polyhedra_Library::Interfaces::Java::get_ptr(), Parma_Polyhedra_Library::Interfaces::Java::is_java_marked(), Java_parma_1polyhedra_1library_PPL_1Object_initIDs(), and Parma_Polyhedra_Library::Interfaces::Java::set_ptr().

11.21.2.101 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Relation_Symbol_EQUAL_ID

Definition at line 400 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Relation_1Symbol_initIDs().

11.21.2.102 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Relation_Symbol_GREATER_OR_EQUAL_ID

Definition at line 401 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Relation_1Symbol_initIDs().

11.21.2.103 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Relation_Symbol_GREATER_THAN_ID

Definition at line 402 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Relation_1Symbol_initIDs().

11.21.2.104 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Relation_Symbol_ordinal_ID

Definition at line 403 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Relation_1Symbol_initIDs().

11.21.2.105 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::System_Iterator_has_next_ID

Definition at line 349 of file ppl_java_common.defs.hh.

Referenced by Parma_Polyhedra_Library::Interfaces::Java::build_cxx_system(), and Java_parma_-1polyhedra_1library_Constraint_1System_initIDs().

11.21.2.106 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::System_iterator_ID

Definition at line 348 of file ppl_java_common.defs.hh.

Referenced by Parma_Polyhedra_Library::Interfaces::Java::build_cxx_system(), and Java_parma_-1polyhedra_1library_Constraint_1System_initIDs().

11.21.2.107 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::System_Iterator_next_ID

Definition at line 350 of file ppl_java_common.defs.hh.

Referenced by Parma_Polyhedra_Library::Interfaces::Java::build_cxx_system(), and Java_parma_-1polyhedra_1library_Constraint_1System_initIDs().

11.21.2.108 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Variable_init_ID

Definition at line 406 of file ppl_java_common.defs.hh.

Referenced by Parma_Polyhedra_Library::Interfaces::Java::build_java_variable(), and Java_parma_-1polyhedra_1library_Variable_initIDs().

11.21.2.109 jfieldID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Variable_varid_ID

Definition at line 405 of file ppl_java_common.defs.hh.

Referenced by Parma_Polyhedra_Library::Interfaces::Java::build_cxx_variable(), and Java_parma_-1polyhedra_1library_Variable_initIDs().

11.21.2.110 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Variables_Set_add_ID

Definition at line 409 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Variables_1Set_initIDs().

11.21.2.111 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Variables_Set_init_ID

Definition at line 408 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Variables_1Set_initIDs().

11.21.2.112 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Variables_Set_Iterator_has_next_ID

Definition at line 412 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Variables_1Set_initIDs().

11.21.2.113 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Variables_Set_iterator_ID

Definition at line 410 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Variables_1Set_initIDs().

11.21.2.114 jmethodID Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Variables_Set_Iterator_next_ID

Definition at line 413 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Variables_1Set_initIDs().

The documentation for this struct was generated from the following file:

- ppl_java_common.defs.hh

11.22 parma_polyhedra_library::Length_Error_Exception Class Reference

Exceptions caused by too big length/size values.

Public Member Functions

- Length_Error_Exception (String s)

  Constructor.

11.22.1 Detailed Description

Exceptions caused by too big length/size values.

Definition at line 28 of file Length_Error_Exception.java.

11.22.2 Constructor & Destructor Documentation

11.22.2.1 parma_polyhedra_library::Length_Error_Exception::Length_Error_Exception (String *s*) `[inline]`

Constructor.

Definition at line 30 of file Length_Error_Exception.java.

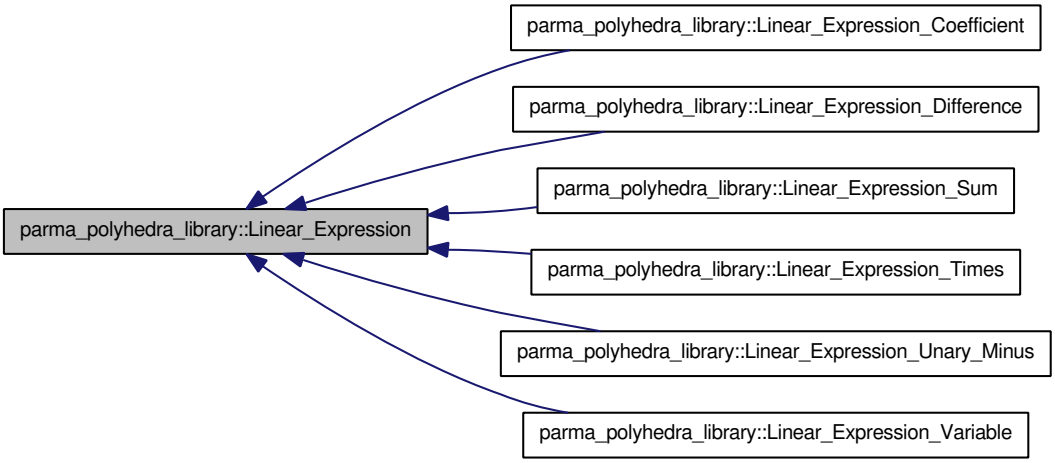
The documentation for this class was generated from the following file:

- Length_Error_Exception.java

11.23 parma_polyhedra_library::Linear_Expression Class Reference

A linear expression.

Inheritance diagram for parma_polyhedra_library::Linear_Expression:



Public Member Functions

- Linear_Expression sum (Linear_Expression y)

Returns the sum of `this` *and* `y`.

- Linear_Expression subtract (Linear_Expression y)

  Returns the difference of `this` *and* `y`.

- Linear_Expression times (Coefficient c)

  Returns the product of `this` *times* `c`.

- Linear_Expression unary_minus ()

  Returns the negation of `this`.

- abstract Linear_Expression clone ()

  Returns a copy of the linear expression.

- native String ascii_dump ()

  Returns an ascii formatted internal representation of `this`.

- native String toString ()

  Returns a string representation of `this`.

- native boolean is_zero ()

  Returns `true` *if and only if* `*this` *is* 0.

- native boolean all_homogeneous_terms_are_zero ()

  Returns `true` *if and only if all the homogeneous terms of* `*this` *are* 0.

Static Package Functions

- [static initializer]

Static Private Member Functions

- static native void initIDs ()

11.23.1 Detailed Description

A linear expression. An object of the class Linear_Expression represents a linear expression that can be built from a Linear_Expression_Variable, Linear_Expression_Coefficient, Linear_Expression_Sum, Linear_Expression_Difference, Linear_Expression_Unary_Minus.

Definition at line 36 of file Linear_Expression.java.

11.23.2 Member Function Documentation

11.23.2.1 parma_polyhedra_library::Linear_Expression::[static initializer] () `[inline, static, package]`

Reimplemented in parma_polyhedra_library::Linear_Expression_Coefficient, parma_polyhedra_library::Linear_Expression_Difference, parma_polyhedra_library::Linear_Expression_Sum, parma_polyhedra_library::Linear_Expression_Times, parma_polyhedra_library::Linear_Expression_Unary_Minus, and parma_polyhedra_library::Linear_Expression_Variable.

11.23.2.2 native boolean parma_polyhedra_library::Linear_Expression::all_homogeneous_terms_are_zero ()

Returns `true` if and only if all the homogeneous terms of `*this` are 0.

11.23.2.3 native String parma_polyhedra_library::Linear_Expression::ascii_dump ()

Returns an ascii formatted internal representation of `this`.

11.23.2.4 abstract Linear_Expression parma_polyhedra_library::Linear_Expression::clone () [pure virtual]

Returns a copy of the linear expression.

Implemented in parma_polyhedra_library::Linear_Expression_Coefficient, parma_polyhedra_library::Linear_Expression_Difference, parma_polyhedra_library::Linear_Expression_Sum, parma_polyhedra_library::Linear_Expression_Times, parma_polyhedra_library::Linear_Expression_Unary_Minus, and parma_polyhedra_library::Linear_Expression_Variable.

Referenced by parma_polyhedra_library::Artificial_Parameter::Artificial_Parameter(), parma_polyhedra_library::Congruence::Congruence(), parma_polyhedra_library::Constraint::Constraint(), parma_polyhedra_library::Generator::Generator(), parma_polyhedra_library::Grid_Generator::Grid_Generator(), parma_polyhedra_library::Linear_Expression_Difference::Linear_Expression_Difference(), parma_polyhedra_library::Linear_Expression_Sum::Linear_Expression_Sum(), parma_polyhedra_library::Linear_Expression_Times::Linear_Expression_Times(), and parma_polyhedra_library::Linear_Expression_Unary_Minus::Linear_Expression_Unary_Minus().

11.23.2.5 static native void parma_polyhedra_library::Linear_Expression::initIDs () [static, private]

Reimplemented in parma_polyhedra_library::Linear_Expression_Coefficient, parma_polyhedra_library::Linear_Expression_Difference, parma_polyhedra_library::Linear_Expression_Sum, parma_polyhedra_library::Linear_Expression_Times, parma_polyhedra_library::Linear_Expression_Unary_Minus, and parma_polyhedra_library::Linear_Expression_Variable.

11.23.2.6 native boolean parma_polyhedra_library::Linear_Expression::is_zero ()

Returns `true` if and only if `*this` is 0.

11.23.2.7 Linear_Expression parma_polyhedra_library::Linear_Expression::subtract (Linear_Expression *y*) [inline]

Returns the difference of `this` and `y`.

Definition at line 44 of file Linear_Expression.java.

11.23.2.8 Linear_Expression parma_polyhedra_library::Linear_Expression::sum (Linear_Expression *y*) [inline]

Returns the sum of `this` and `y`.

Definition at line 39 of file Linear_Expression.java.

11.23.2.9 Linear_Expression parma_polyhedra_library::Linear_Expression::times (Coefficient *c*) [inline]

Returns the product of `this` times `c`.

Definition at line 49 of file Linear_Expression.java.

11.23.2.10 native String parma_polyhedra_library::Linear_Expression::toString ()

Returns a string representation of `this`.

11.23.2.11 Linear_Expression parma_polyhedra_library::Linear_Expression::unary_minus () [inline]

Returns the negation of `this`.

Definition at line 54 of file Linear_Expression.java.

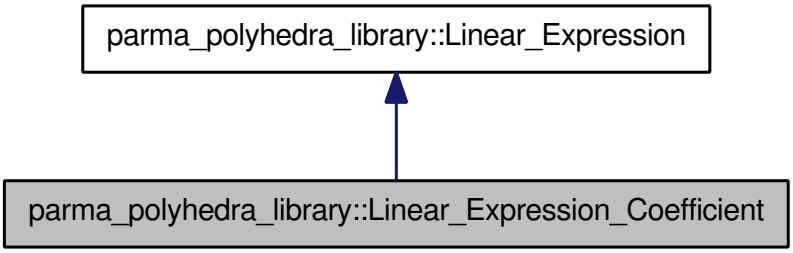
The documentation for this class was generated from the following file:

- Linear_Expression.java

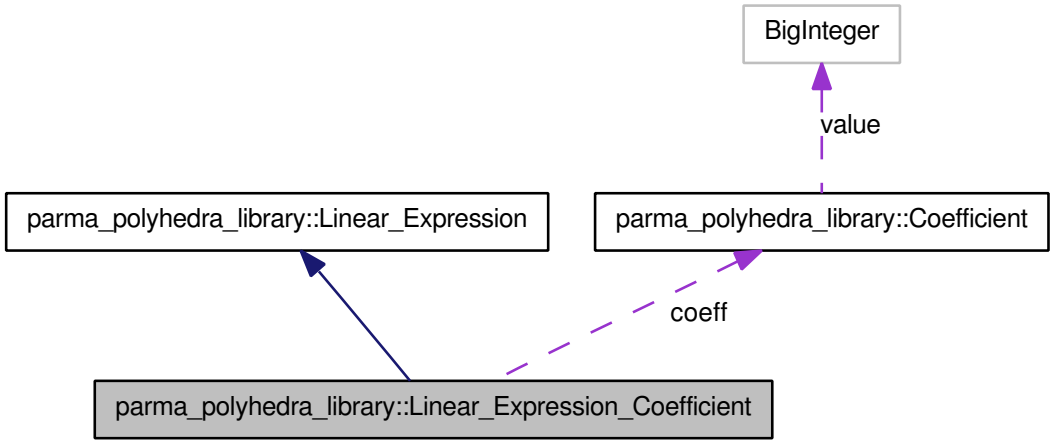
11.24 parma_polyhedra_library::Linear_Expression_Coefficient Class Reference

A linear expression built from a coefficient.

Inheritance diagram for parma_polyhedra_library::Linear_Expression_Coefficient:



Collaboration diagram for parma_polyhedra_library::Linear_Expression_Coefficient:



Public Member Functions

- Linear_Expression_Coefficient (Coefficient c)

  Builds the object corresponding to a copy of the coefficient c.

- Coefficient argument ()

  Returns coefficient representing the linear expression.

- Linear_Expression_Coefficient clone ()

  Builds a copy of this.

Protected Attributes

- Coefficient coeff

  The coefficient representing the linear expression.

Static Package Functions

- [static initializer]

Static Private Member Functions

- static native void initIDs ()

11.24.1 Detailed Description

A linear expression built from a coefficient.

Definition at line 28 of file Linear_Expression_Coefficient.java.

11.24.2 Constructor & Destructor Documentation

11.24.2.1 parma_polyhedra_library::Linear_Expression_Coefficient::Linear_Expression_Coefficient (Coefficient *c*) [inline]

Builds the object corresponding to a copy of the coefficient `c`.

Definition at line 35 of file Linear_Expression_Coefficient.java.

References coeff.

Referenced by clone().

11.24.3 Member Function Documentation

11.24.3.1 parma_polyhedra_library::Linear_Expression_Coefficient::[static initializer] () [inline, static, package]

Reimplemented from parma_polyhedra_library::Linear_Expression.

11.24.3.2 Coefficient parma_polyhedra_library::Linear_Expression_Coefficient::argument () [inline]

Returns coefficient representing the linear expression.

Definition at line 40 of file Linear_Expression_Coefficient.java.

References coeff.

11.24.3.3 Linear_Expression_Coefficient parma_polyhedra_library::Linear_Expression_Coefficient::clone () [inline, virtual]

Builds a copy of this.

Implements parma_polyhedra_library::Linear_Expression.

Definition at line 45 of file Linear_Expression_Coefficient.java.

References coeff, and Linear_Expression_Coefficient().

11.24.3.4 static native void parma_polyhedra_library::Linear_Expression_Coefficient::initIDs () [static, private]

Reimplemented from parma_polyhedra_library::Linear_Expression.

11.24.4 Member Data Documentation

11.24.4.1 Coefficient parma_polyhedra_library::Linear_Expression_Coefficient::coeff [protected]

The coefficient representing the linear expression.

Definition at line 32 of file Linear_Expression_Coefficient.java.

Referenced by argument(), clone(), and Linear_Expression_Coefficient().

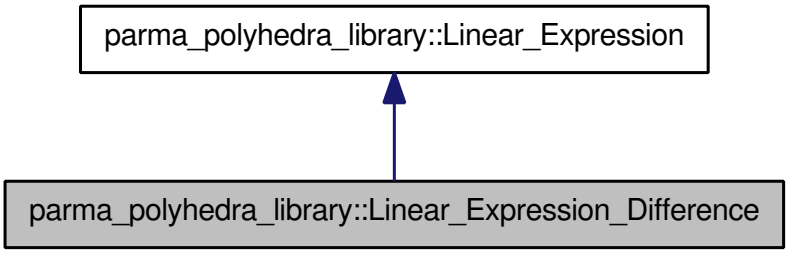
The documentation for this class was generated from the following file:

- Linear_Expression_Coefficient.java

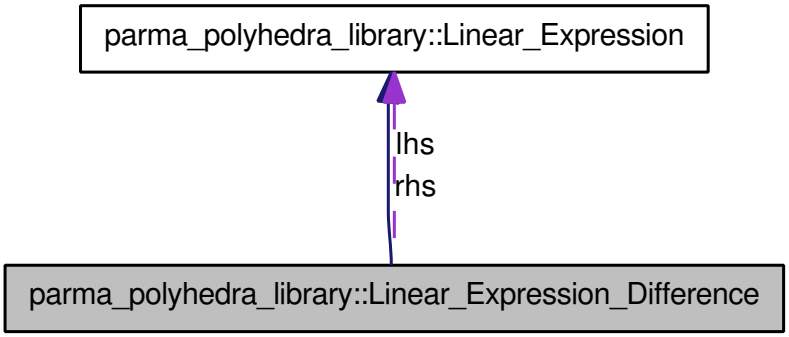
11.25 parma_polyhedra_library::Linear_Expression_Difference Class Reference

The difference of two linear expressions.

Inheritance diagram for parma_polyhedra_library::Linear_Expression_Difference:



Collaboration diagram for parma_polyhedra_library::Linear_Expression_Difference:



Public Member Functions

- Linear_Expression_Difference (Linear_Expression x, Linear_Expression y)
  Builds an object that represents the difference of the copy `x` and `y`.
- Linear_Expression left_hand_side ()
  Returns the left hand side of `this`.
- Linear_Expression right_hand_side ()
  Returns the left hand side of `this`.
- Linear_Expression_Difference clone ()
  Builds a copy of this.

Protected Attributes

- Linear_Expression lhs
  The value of the left hand side of `this`.
- Linear_Expression rhs
  The value of the right hand side of `this`.

Static Package Functions

- [static initializer]

Static Private Member Functions

- static native void initIDs ()

11.25.1 Detailed Description

The difference of two linear expressions.

Definition at line 28 of file Linear_Expression_Difference.java.

11.25.2 Constructor & Destructor Documentation

11.25.2.1 parma_polyhedra_library::Linear_Expression_Difference::Linear_Expression_Difference (Linear_Expression *x*, Linear_Expression *y*) [inline]

Builds an object that represents the difference of the copy `x` and `y`.

Definition at line 41 of file Linear_Expression_Difference.java.

References parma_polyhedra_library::Linear_Expression::clone(), lhs, and rhs.

Referenced by clone().

11.25.3 Member Function Documentation

11.25.3.1 parma_polyhedra_library::Linear_Expression_Difference::[static initializer] () [inline, static, package]

Reimplemented from parma_polyhedra_library::Linear_Expression.

11.25.3.2 Linear_Expression_Difference parma_polyhedra_library::Linear_Expression_Difference::clone () [inline, virtual]

Builds a copy of this.

Implements parma_polyhedra_library::Linear_Expression.

Definition at line 58 of file Linear_Expression_Difference.java.

References lhs, Linear_Expression_Difference(), and rhs.

11.25.3.3 static native void parma_polyhedra_library::Linear_Expression_Difference::initIDs () [static, private]

Reimplemented from parma_polyhedra_library::Linear_Expression.

11.25.3.4 Linear_Expression parma_polyhedra_library::Linear_Expression_Difference::left_hand_side () [inline]

Returns the left hand side of `this`.

Definition at line 48 of file Linear_Expression_Difference.java.

References lhs.

11.25.3.5 Linear_Expression parma_polyhedra_library::Linear_Expression_Difference::right_hand_side () [inline]

Returns the left hand side of `this`.

Definition at line 53 of file Linear_Expression_Difference.java.

References rhs.

11.25.4 Member Data Documentation

11.25.4.1 Linear_Expression parma_polyhedra_library::Linear_Expression_Difference::lhs [protected]

The value of the left hand side of `this`.

Definition at line 32 of file Linear_Expression_Difference.java.

Referenced by clone(), left_hand_side(), and Linear_Expression_Difference().

11.25.4.2 Linear_Expression parma_polyhedra_library::Linear_Expression_Difference::rhs [protected]

The value of the right hand side of `this`.

Definition at line 35 of file Linear_Expression_Difference.java.

Referenced by clone(), Linear_Expression_Difference(), and right_hand_side().

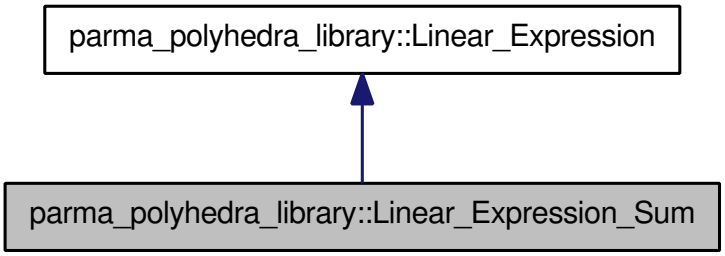
The documentation for this class was generated from the following file:

- Linear_Expression_Difference.java

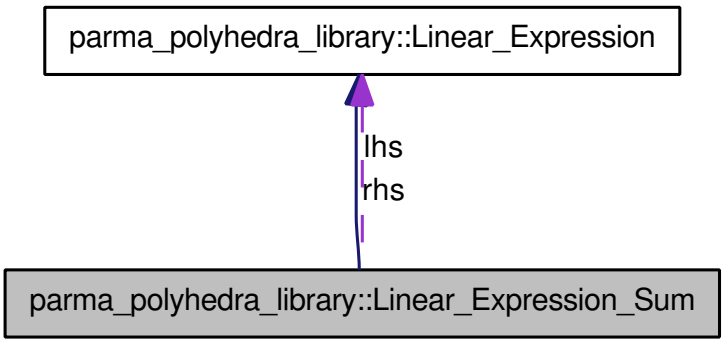
11.26 parma_polyhedra_library::Linear_Expression_Sum Class Reference

The sum of two linear expressions.

Inheritance diagram for parma_polyhedra_library::Linear_Expression_Sum:



Collaboration diagram for parma_polyhedra_library::Linear_Expression_Sum:



Public Member Functions

- Linear_Expression_Sum (Linear_Expression x, Linear_Expression y)

  Builds an object that represents the sum of the copy of `x` *and* `y`.

- Linear_Expression left_hand_side ()

  Returns the left hand side of `this`.

- Linear_Expression right_hand_side ()

  Returns the right hand side of `this`.

- Linear_Expression_Sum clone ()

  Builds a copy of this.

Protected Attributes

- Linear_Expression lhs

  The value of the left hand side of `this`.

- Linear_Expression rhs

  The value of the right hand side of `this`.

Static Package Functions

- [static initializer]

Static Private Member Functions

- static native void initIDs ()

11.26.1 Detailed Description

The sum of two linear expressions.

Definition at line 28 of file Linear_Expression_Sum.java.

11.26.2 Constructor & Destructor Documentation

11.26.2.1 parma_polyhedra_library::Linear_Expression_Sum::Linear_Expression_Sum (Linear_Expression *x*, Linear_Expression *y*) [inline]

Builds an object that represents the sum of the copy of `x` and `y`.

Definition at line 38 of file Linear_Expression_Sum.java.

References parma_polyhedra_library::Linear_Expression::clone(), lhs, and rhs.

Referenced by clone().

11.26.3 Member Function Documentation

11.26.3.1 parma_polyhedra_library::Linear_Expression_Sum::[static initializer] () [inline, static, package]

Reimplemented from parma_polyhedra_library::Linear_Expression.

11.26.3.2 Linear_Expression_Sum parma_polyhedra_library::Linear_Expression_Sum::clone () [inline, virtual]

Builds a copy of this.

Implements parma_polyhedra_library::Linear_Expression.

Definition at line 54 of file Linear_Expression_Sum.java.

References lhs, Linear_Expression_Sum(), and rhs.

11.26.3.3 static native void parma_polyhedra_library::Linear_Expression_Sum::initIDs () [static, private]

Reimplemented from parma_polyhedra_library::Linear_Expression.

11.26.3.4 Linear_Expression parma_polyhedra_library::Linear_Expression_Sum::left_hand_side () [inline]

Returns the left hand side of `this`.

Definition at line 44 of file Linear_Expression_Sum.java.

References lhs.

11.26.3.5 Linear_Expression parma_polyhedra_library::Linear_Expression_Sum::right_hand_side () [inline]

Returns the right hand side of `this`.

Definition at line 49 of file Linear_Expression_Sum.java.

References rhs.

11.26.4 Member Data Documentation

11.26.4.1 Linear_Expression parma_polyhedra_library::Linear_Expression_Sum::lhs [protected]

The value of the left hand side of `this`.

Definition at line 32 of file Linear_Expression_Sum.java.

Referenced by clone(), left_hand_side(), and Linear_Expression_Sum().

11.26.4.2 Linear_Expression parma_polyhedra_library::Linear_Expression_Sum::rhs [protected]

The value of the right hand side of `this`.

Definition at line 35 of file Linear_Expression_Sum.java.

Referenced by clone(), Linear_Expression_Sum(), and right_hand_side().

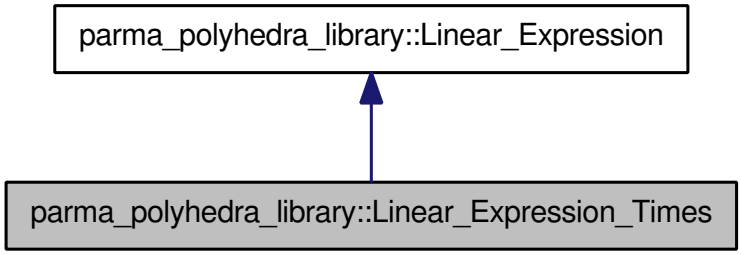
The documentation for this class was generated from the following file:

- Linear_Expression_Sum.java

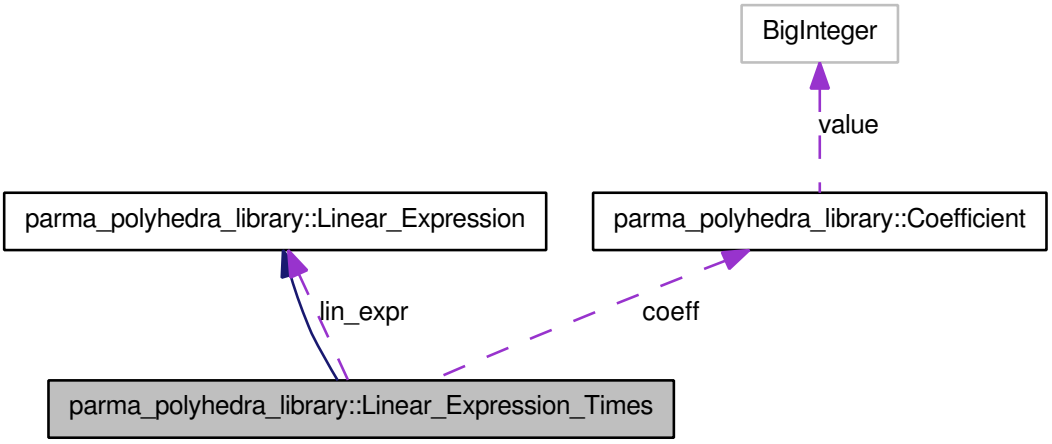
11.27 parma_polyhedra_library::Linear_Expression_Times Class Reference

The product of a linear expression and a coefficient.

Inheritance diagram for parma_polyhedra_library::Linear_Expression_Times:



Collaboration diagram for parma_polyhedra_library::Linear_Expression_Times:



Public Member Functions

- Linear_Expression_Times (Coefficient c, Variable v)

  Builds an object cloning the input arguments.

- Linear_Expression_Times (Coefficient c, Linear_Expression l)

  Builds an object cloning the input arguments.

- Linear_Expression_Times (Linear_Expression l, Coefficient c)

  Builds an object cloning the input arguments.

- Coefficient coefficient ()

  Returns the coefficient of `this`.

- Linear_Expression linear_expression ()

  Returns the linear expression subobject of `this`.

- Linear_Expression_Times clone ()

  Builds a copy of this.

Protected Attributes

- Coefficient coeff

  The value of the coefficient.

- Linear_Expression lin_expr

  The value of the inner linear expression.

Static Package Functions

- [static initializer]

Static Private Member Functions

- static native void initIDs ()

11.27.1 Detailed Description

The product of a linear expression and a coefficient.

Definition at line 28 of file Linear_Expression_Times.java.

11.27.2 Constructor & Destructor Documentation

11.27.2.1 parma_polyhedra_library::Linear_Expression_Times::Linear_Expression_Times (Coefficient *c*, Variable *v*) `[inline]`

Builds an object cloning the input arguments.

Definition at line 38 of file Linear_Expression_Times.java.

References coeff, and lin_expr.

Referenced by clone().

11.27.2.2 parma_polyhedra_library::Linear_Expression_Times::Linear_Expression_Times (Coefficient *c*, Linear_Expression *l*) `[inline]`

Builds an object cloning the input arguments.

Definition at line 44 of file Linear_Expression_Times.java.

References parma_polyhedra_library::Linear_Expression::clone(), coeff, and lin_expr.

11.27.2.3 parma_polyhedra_library::Linear_Expression_Times::Linear_Expression_Times (Linear_Expression *l*, Coefficient *c*) `[inline]`

Builds an object cloning the input arguments.

Definition at line 50 of file Linear_Expression_Times.java.

References parma_polyhedra_library::Linear_Expression::clone(), coeff, and lin_expr.

11.27.3 Member Function Documentation

11.27.3.1 parma_polyhedra_library::Linear_Expression_Times::[static initializer] () [inline, static, package]

Reimplemented from parma_polyhedra_library::Linear_Expression.

11.27.3.2 Linear_Expression_Times parma_polyhedra_library::Linear_Expression_Times::clone () [inline, virtual]

Builds a copy of this.

Implements parma_polyhedra_library::Linear_Expression.

Definition at line 66 of file Linear_Expression_Times.java.

References coeff, lin_expr, and Linear_Expression_Times().

11.27.3.3 Coefficient parma_polyhedra_library::Linear_Expression_Times::coefficient () [inline]

Returns the coefficient of `this`.

Definition at line 56 of file Linear_Expression_Times.java.

References coeff.

11.27.3.4 static native void parma_polyhedra_library::Linear_Expression_Times::initIDs () [static, private]

Reimplemented from parma_polyhedra_library::Linear_Expression.

11.27.3.5 Linear_Expression parma_polyhedra_library::Linear_Expression_Times::linear_expression () [inline]

Returns the linear expression subobject of `this`.

Definition at line 61 of file Linear_Expression_Times.java.

References lin_expr.

11.27.4 Member Data Documentation

11.27.4.1 Coefficient parma_polyhedra_library::Linear_Expression_Times::coeff [protected]

The value of the coefficient.

Definition at line 32 of file Linear_Expression_Times.java.

Referenced by clone(), coefficient(), and Linear_Expression_Times().

11.27.4.2 Linear_Expression parma_polyhedra_library::Linear_Expression_Times::lin_expr [protected]

The value of the inner linear expression.

Definition at line 35 of file Linear_Expression_Times.java.

Referenced by clone(), linear_expression(), and Linear_Expression_Times().

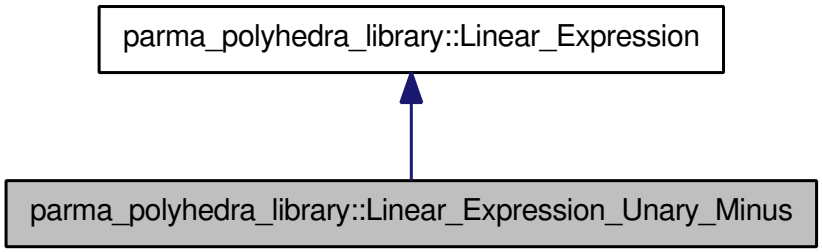
The documentation for this class was generated from the following file:

- Linear_Expression_Times.java

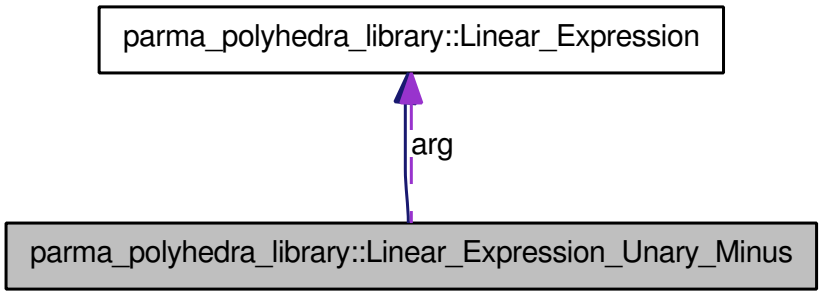
11.28 parma_polyhedra_library::Linear_Expression_Unary_Minus Class Reference

The negation of a linear expression.

Inheritance diagram for parma_polyhedra_library::Linear_Expression_Unary_Minus:



Collaboration diagram for parma_polyhedra_library::Linear_Expression_Unary_Minus:



Public Member Functions

- Linear_Expression_Unary_Minus (Linear_Expression x)

  Builds an object that represents the negation of the copy `x`*.*

- Linear_Expression argument ()

  Returns the value that `this` *negates.*

- Linear_Expression_Unary_Minus clone ()

  Builds a copy of this.

Protected Attributes

- Linear_Expression arg

  The value that `this` *negates.*

Static Package Functions

- [static initializer]

Static Private Member Functions

- static native void initIDs ()

11.28.1 Detailed Description

The negation of a linear expression.

Definition at line 28 of file Linear_Expression_Unary_Minus.java.

11.28.2 Constructor & Destructor Documentation

11.28.2.1 parma_polyhedra_library::Linear_Expression_Unary_Minus::Linear_Expression_Unary_Minus (Linear_Expression *x*) [inline]

Builds an object that represents the negation of the copy `x`.

Definition at line 35 of file Linear_Expression_Unary_Minus.java.

References arg, and parma_polyhedra_library::Linear_Expression::clone().

Referenced by clone().

11.28.3 Member Function Documentation

11.28.3.1 parma_polyhedra_library::Linear_Expression_Unary_Minus::[static initializer] () [inline, static, package]

Reimplemented from parma_polyhedra_library::Linear_Expression.

11.28.3.2 Linear_Expression parma_polyhedra_library::Linear_Expression_Unary_Minus::argument () [inline]

Returns the value that `this` negates.

Definition at line 40 of file Linear_Expression_Unary_Minus.java.

References arg.

11.28.3.3 Linear_Expression_Unary_Minus parma_polyhedra_library::Linear_Expression_Unary_Minus::clone () [inline, virtual]

Builds a copy of this.

Implements parma_polyhedra_library::Linear_Expression.

Definition at line 45 of file Linear_Expression_Unary_Minus.java.

References arg, and Linear_Expression_Unary_Minus().

11.28.3.4 static native void parma_polyhedra_library::Linear_Expression_Unary_Minus::initIDs () [static, private]

Reimplemented from parma_polyhedra_library::Linear_Expression.

11.28.4 Member Data Documentation

11.28.4.1 Linear_Expression parma_polyhedra_library::Linear_Expression_Unary_Minus::arg [protected]

The value that `this` negates.

Definition at line 32 of file Linear_Expression_Unary_Minus.java.

Referenced by argument(), clone(), and Linear_Expression_Unary_Minus().

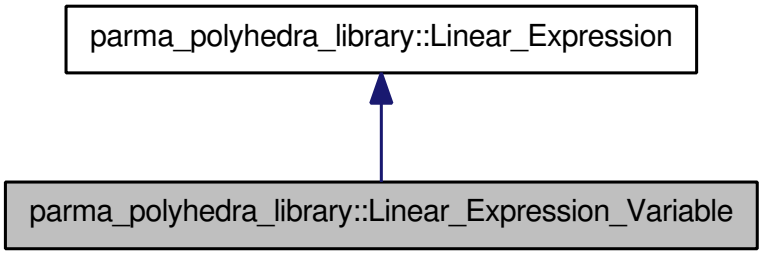
The documentation for this class was generated from the following file:

- Linear_Expression_Unary_Minus.java

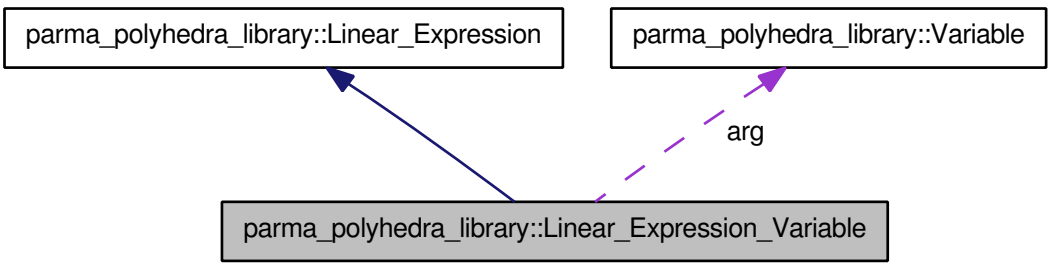
11.29 parma_polyhedra_library::Linear_Expression_Variable Class Reference

A linear expression built from a variable.

Inheritance diagram for parma_polyhedra_library::Linear_Expression_Variable:



Collaboration diagram for parma_polyhedra_library::Linear_Expression_Variable:



Public Member Functions

- Linear_Expression_Variable (Variable v)

  Builds the object associated to the copy of `v`.

- Variable argument ()

  Returns the variable representing the linear expression.

- Linear_Expression_Variable clone ()

  Builds a copy of this.

Protected Attributes

- Variable arg

Static Package Functions

- [static initializer]

Private Member Functions

- int var_id ()

Static Private Member Functions

- static native void initIDs ()

11.29.1 Detailed Description

A linear expression built from a variable.

Definition at line 28 of file Linear_Expression_Variable.java.

11.29.2 Constructor & Destructor Documentation

11.29.2.1 parma_polyhedra_library::Linear_Expression_Variable::Linear_Expression_Variable (Variable *v*) [inline]

Builds the object associated to the copy of `v`.

Definition at line 34 of file Linear_Expression_Variable.java.

References arg, and parma_polyhedra_library::Variable::id().

Referenced by clone().

11.29.3 Member Function Documentation

11.29.3.1 parma_polyhedra_library::Linear_Expression_Variable::[static initializer] () [inline, static, package]

Reimplemented from parma_polyhedra_library::Linear_Expression.

11.29.3.2 Variable parma_polyhedra_library::Linear_Expression_Variable::argument () [inline]

Returns the variable representing the linear expression.

Definition at line 39 of file Linear_Expression_Variable.java.

References arg.

11.29.3.3 Linear_Expression_Variable parma_polyhedra_library::Linear_Expression_-Variable::clone () [inline, virtual]

Builds a copy of this.

Implements parma_polyhedra_library::Linear_Expression.

Definition at line 44 of file Linear_Expression_Variable.java.

References arg, and Linear_Expression_Variable().

11.29.3.4 static native void parma_polyhedra_library::Linear_Expression_Variable::initIDs () [static, private]

Reimplemented from parma_polyhedra_library::Linear_Expression.

11.29.3.5 int parma_polyhedra_library::Linear_Expression_Variable::var_id () [inline, private]

Definition at line 48 of file Linear_Expression_Variable.java.

References arg, and parma_polyhedra_library::Variable::id().

11.29.4 Member Data Documentation

11.29.4.1 Variable parma_polyhedra_library::Linear_Expression_Variable::arg [protected]

Definition at line 31 of file Linear_Expression_Variable.java.

Referenced by argument(), clone(), Linear_Expression_Variable(), and var_id().

The documentation for this class was generated from the following file:

- Linear_Expression_Variable.java

11.30 parma_polyhedra_library::Logic_Error_Exception Class Reference

Exceptions due to errors in low-level routines.

Public Member Functions

- Logic_Error_Exception (String s)

  Constructor.

11.30.1 Detailed Description

Exceptions due to errors in low-level routines. These exceptions may be generated, for instance, by the inability of querying/controlling the FPU behavior with respect to rounding modes.

Definition at line 31 of file Logic_Error_Exception.java.

11.30.2 Constructor & Destructor Documentation

11.30.2.1 parma_polyhedra_library::Logic_Error_Exception::Logic_Error_Exception (String *s*) `[inline]`

Constructor.

Definition at line 34 of file Logic_Error_Exception.java.

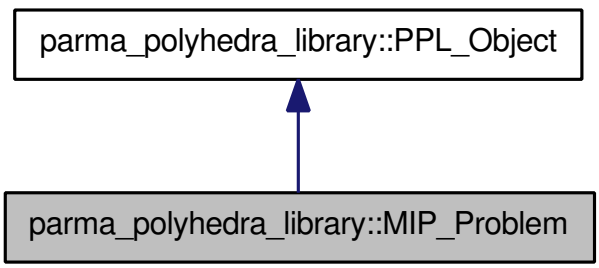
The documentation for this class was generated from the following file:

- Logic_Error_Exception.java

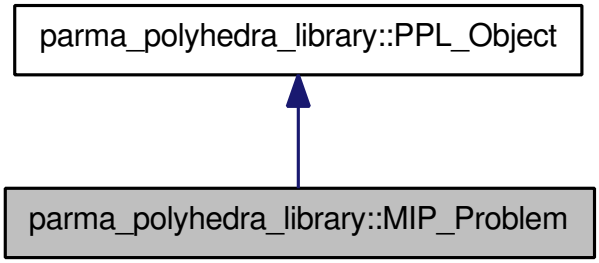
11.31 parma_polyhedra_library::MIP_Problem Class Reference

A Mixed Integer (linear) Programming problem.

Inheritance diagram for parma_polyhedra_library::MIP_Problem:



Collaboration diagram for parma_polyhedra_library::MIP_Problem:



Public Member Functions

Functions that Do Not Modify the MIP_Problem

- native long max_space_dimension ()
  Returns the maximum space dimension an MIP_Problem can handle.
- native long space_dimension ()
  Returns the space dimension of the MIP problem.
- native Variables_Set integer_space_dimensions ()
  Returns a set containing all the variables' indexes constrained to be integral.
- native Constraint_System constraints ()
  Returns the constraints .
- native Linear_Expression objective_function ()
  Returns the objective function.
- native Optimization_Mode optimization_mode ()
  Returns the optimization mode.
- native String ascii_dump ()
  Returns an ascii formatted internal representation of `this`.
- native String toString ()
  Returns a string representation of `this`.
- native long total_memory_in_bytes ()
  Returns the total size in bytes of the memory occupied by the underlying C++ object.
- native boolean OK ()
  Checks if all the invariants are satisfied.

Functions that May Modify the MIP_Problem

- native void clear ()
  Resets `this` to be equal to the trivial MIP problem.
- native void add_space_dimensions_and_embed (long m)

Adds `m` new space dimensions and embeds the old MIP problem in the new vector space.

- native void add_to_integer_space_dimensions (Variables_Set i_vars)
  Sets the variables whose indexes are in set `i_vars` to be integer space dimensions.
- native void add_constraint (Constraint c)
  Adds a copy of constraint `c` to the MIP problem.
- native void add_constraints (Constraint_System cs)
  Adds a copy of the constraints in `cs` to the MIP problem.
- native void set_objective_function (Linear_Expression obj)
  Sets the objective function to `obj`.
- native void set_optimization_mode (Optimization_Mode mode)
  Sets the optimization mode to `mode`.

Computing the Solution of the MIP_Problem

- native boolean is_satisfiable ()
  Checks satisfiability of `*this`.
- native MIP_Problem_Status solve ()
  Optimizes the MIP problem.
- native void evaluate_objective_function (Generator evaluating_point, Coefficient num, Coefficient den)
  Sets `num` and `den` so that $\frac{num}{den}$ is the result of evaluating the objective function on `evaluating_point`.
- native Generator feasible_point ()
  Returns a feasible point for `*this`, if it exists.
- native Generator optimizing_point ()
  Returns an optimal point for `this`, if it exists.
- native void optimal_value (Coefficient num, Coefficient den)
  Sets `num` and `den` so that $\frac{num}{den}$ is the solution of the optimization problem.

Querying/Setting Control Parameters

- native Control_Parameter_Value get_control_parameter (Control_Parameter_Name name)
  Returns the value of control parameter `name`.
- native void set_control_parameter (Control_Parameter_Value value)
  Sets control parameter `value`.

Private Member Functions

- native void build_cpp_object (long dim)
  Builds the underlying C++ object.

- native void build_cpp_object (long dim, Constraint_System cs, Linear_Expression obj, Optimization_Mode mode)

  Builds the underlying C++ object.

- native void build_cpp_object (MIP_Problem y)

  Builds the underlying C++ object.

Constructors and Destructor

- MIP_Problem (long dim)

  Builds a trivial MIP problem.

- MIP_Problem (long dim, Constraint_System cs, Linear_Expression obj, Optimization_Mode mode)

  Builds an MIP problem having space dimension `dim` *from the constraint system* `cs`*, the objective function* `obj` *and optimization mode* `mode`*.*

- MIP_Problem (MIP_Problem y)

  Builds a copy of `y`*.*

- native void free ()

  Releases all resources managed by `this`*, also resetting it to a null reference.*

- native void finalize ()

  Releases all resources managed by `this`*.*

11.31.1 Detailed Description

A Mixed Integer (linear) Programming problem. An object of this class encodes a mixed integer (linear) programming problem. The MIP problem is specified by providing:

- the dimension of the vector space;
- the feasible region, by means of a finite set of linear equality and non-strict inequality constraints;
- the subset of the unknown variables that range over the integers (the other variables implicitly ranging over the reals);
- the objective function, described by a Linear_Expression;
- the optimization mode (either maximization or minimization).

The class provides support for the (incremental) solution of the MIP problem based on variations of the revised simplex method and on branch-and-bound techniques. The result of the resolution process is expressed in terms of an enumeration, encoding the feasibility and the unboundedness of the optimization problem. The class supports simple feasibility tests (i.e., no optimization), as well as the extraction of an optimal (resp., feasible) point, provided the MIP_Problem is optimizable (resp., feasible).

By exploiting the incremental nature of the solver, it is possible to reuse part of the computational work already done when solving variants of a given MIP_Problem: currently, incremental resolution supports the

addition of space dimensions, the addition of constraints, the change of objective function and the change of optimization mode.

Definition at line 57 of file MIP_Problem.java.

11.31.2 Constructor & Destructor Documentation

11.31.2.1 parma_polyhedra_library::MIP_Problem::MIP_Problem (long *dim*) [inline]

Builds a trivial MIP problem.

A trivial MIP problem requires to maximize the objective function 0 on a vector space under no constraints at all: the origin of the vector space is an optimal solution.

Parameters

dim The dimension of the vector space enclosing `this`.

Exceptions

std::length_error Thrown if `dim` exceeds `max_space_dimension()`.

Definition at line 74 of file MIP_Problem.java.

References build_cpp_object().

11.31.2.2 parma_polyhedra_library::MIP_Problem::MIP_Problem (long *dim*, Constraint_System *cs*, Linear_Expression *obj*, Optimization_Mode *mode*) [inline]

Builds an MIP problem having space dimension `dim` from the constraint system `cs`, the objective function `obj` and optimization mode `mode`.

Parameters

dim The dimension of the vector space enclosing `this`.

cs The constraint system defining the feasible region.

obj The objective function.

mode The optimization mode.

Exceptions

std::length_error Thrown if `dim` exceeds `max_space_dimension()`.

std::invalid_argument Thrown if the constraint system contains any strict inequality or if the space dimension of the constraint system (resp., the objective function) is strictly greater than `dim`.

Definition at line 103 of file MIP_Problem.java.

References build_cpp_object().

11.31.2.3 parma_polyhedra_library::MIP_Problem::MIP_Problem (MIP_Problem *y*) [inline]

Builds a copy of `y`.

Definition at line 109 of file MIP_Problem.java.

References build_cpp_object().

11.31.3 Member Function Documentation

11.31.3.1 native void parma_polyhedra_library::MIP_Problem::add_constraint (Constraint *c*)

Adds a copy of constraint `c` to the MIP problem.

Exceptions

std::invalid_argument Thrown if the constraint `c` is a strict inequality or if its space dimension is strictly greater than the space dimension of `this`.

11.31.3.2 native void parma_polyhedra_library::MIP_Problem::add_constraints (Constraint_System *cs*)

Adds a copy of the constraints in `cs` to the MIP problem.

Exceptions

std::invalid_argument Thrown if the constraint system `cs` contains any strict inequality or if its space dimension is strictly greater than the space dimension of `*this`.

11.31.3.3 native void parma_polyhedra_library::MIP_Problem::add_space_dimensions_and_embed (long *m*)

Adds `m` new space dimensions and embeds the old MIP problem in the new vector space.

Parameters

m The number of dimensions to add.

Exceptions

std::length_error Thrown if adding `m` new space dimensions would cause the vector space to exceed dimension `max_space_dimension()`.

The new space dimensions will be those having the highest indexes in the new MIP problem; they are initially unconstrained.

11.31.3.4 native void parma_polyhedra_library::MIP_Problem::add_to_integer_space_dimensions (Variables_Set *i_vars*)

Sets the variables whose indexes are in set `i_vars` to be integer space dimensions.

Exceptions

std::invalid_argument Thrown if some index in `i_vars` does not correspond to a space dimension in `this`.

11.31.3.5 native String parma_polyhedra_library::MIP_Problem::ascii_dump ()

Returns an ascii formatted internal representation of `this`.

11.31.3.6 native void parma_polyhedra_library::MIP_Problem::build_cpp_object (MIP_Problem *y*) [private]

Builds the underlying C++ object.

11.31.3.7 native void parma_polyhedra_library::MIP_Problem::build_cpp_object (long *dim*, Constraint_System *cs*, Linear_Expression *obj*, Optimization_Mode *mode*) [private]

Builds the underlying C++ object.

11.31.3.8 native void parma_polyhedra_library::MIP_Problem::build_cpp_object (long *dim*) [private]

Builds the underlying C++ object.

Referenced by MIP_Problem().

11.31.3.9 native void parma_polyhedra_library::MIP_Problem::clear ()

Resets `this` to be equal to the trivial MIP problem.

The space dimension is reset to 0.

11.31.3.10 native Constraint_System parma_polyhedra_library::MIP_Problem::constraints ()

Returns the constraints .

11.31.3.11 native void parma_polyhedra_library::MIP_Problem::evaluate_objective_function (Generator *evaluating_point*, Coefficient *num*, Coefficient *den*)

Sets `num` and `den` so that $\frac{num}{den}$ is the result of evaluating the objective function on `evaluating_point`.

Parameters

evaluating_point The point on which the objective function will be evaluated.

num On exit will contain the numerator of the evaluated value.

den On exit will contain the denominator of the evaluated value.

Exceptions

std::invalid_argument Thrown if `this` and `evaluating_point` are dimension-incompatible or if the generator `evaluating_point` is not a point.

11.31.3.12 native Generator parma_polyhedra_library::MIP_Problem::feasible_point ()

Returns a feasible point for `*this`, if it exists.

Exceptions

std::domain_error Thrown if the MIP problem is not satisfiable.

11.31.3.13 native void parma_polyhedra_library::MIP_Problem::finalize () [protected]

Releases all resources managed by `this`.

11.31.3.14 native void parma_polyhedra_library::MIP_Problem::free ()

Releases all resources managed by `this`, also resetting it to a null reference.

11.31.3.15 native Control_Parameter_Value parma_polyhedra_library::MIP_Problem::get_control_parameter (Control_Parameter_Name *name*)

Returns the value of control parameter `name`.

11.31.3.16 native Variables_Set parma_polyhedra_library::MIP_Problem::integer_space_dimensions ()

Returns a set containing all the variables' indexes constrained to be integral.

11.31.3.17 native boolean parma_polyhedra_library::MIP_Problem::is_satisfiable ()

Checks satisfiability of `*this`.

Returns

`true` if and only if the MIP problem is satisfiable.

11.31.3.18 native long parma_polyhedra_library::MIP_Problem::max_space_dimension ()

Returns the maximum space dimension an MIP_Problem can handle.

11.31.3.19 native Linear_Expression parma_polyhedra_library::MIP_Problem::objective_function ()

Returns the objective function.

11.31.3.20 native boolean parma_polyhedra_library::MIP_Problem::OK ()

Checks if all the invariants are satisfied.

11.31.3.21 native void parma_polyhedra_library::MIP_Problem::optimal_value (Coefficient *num*, Coefficient *den*)

Sets `num` and `den` so that $\frac{num}{den}$ is the solution of the optimization problem.

Exceptions

std::domain_error Thrown if `*this` doesn't not have an optimizing point, i.e., if the MIP problem is unbounded or not satisfiable.

11.31.3.22 native Optimization_Mode parma_polyhedra_library::MIP_Problem::optimization_mode ()

Returns the optimization mode.

11.31.3.23 native Generator parma_polyhedra_library::MIP_Problem::optimizing_point ()

Returns an optimal point for `this`, if it exists.

Exceptions

std::domain_error Thrown if `this` doesn't not have an optimizing point, i.e., if the MIP problem is unbounded or not satisfiable.

11.31.3.24 native void parma_polyhedra_library::MIP_Problem::set_control_parameter (Control_Parameter_Value *value*)

Sets control parameter `value`.

11.31.3.25 native void parma_polyhedra_library::MIP_Problem::set_objective_function (Linear_Expression *obj*)

Sets the objective function to `obj`.

Exceptions

std::invalid_argument Thrown if the space dimension of `obj` is strictly greater than the space dimension of `this`.

11.31.3.26 native void parma_polyhedra_library::MIP_Problem::set_optimization_mode (Optimization_Mode *mode*)

Sets the optimization mode to `mode`.

11.31.3.27 native MIP_Problem_Status parma_polyhedra_library::MIP_Problem::solve ()

Optimizes the MIP problem.

Returns

An MIP_Problem_Status flag indicating the outcome of the optimization attempt (unfeasible, unbounded or optimized problem).

11.31.3.28 native long parma_polyhedra_library::MIP_Problem::space_dimension ()

Returns the space dimension of the MIP problem.

11.31.3.29 native String parma_polyhedra_library::MIP_Problem::toString ()

Returns a string representation of `this`.

11.31.3.30 native long parma_polyhedra_library::MIP_Problem::total_memory_in_bytes ()

Returns the total size in bytes of the memory occupied by the underlying C++ object.

The documentation for this class was generated from the following file:

- MIP_Problem.java

11.32 parma_polyhedra_library::Overflow_Error_Exception Class Reference

Exceptions due to overflow errors.

Public Member Functions

- Overflow_Error_Exception (String s)

  Constructor.

11.32.1 Detailed Description

Exceptions due to overflow errors. These exceptions can be obtained when the library has been configured to use integer coefficients having bounded size.

Definition at line 31 of file Overflow_Error_Exception.java.

11.32.2 Constructor & Destructor Documentation

11.32.2.1 parma_polyhedra_library::Overflow_Error_Exception::Overflow_Error_Exception (String *s*) [inline]

Constructor.

Definition at line 33 of file Overflow_Error_Exception.java.

The documentation for this class was generated from the following file:

- Overflow_Error_Exception.java

11.33 parma_polyhedra_library::Pair< K, V > Class Reference

A pair of values of type K and V.

Public Member Functions

- K getFirst ()

  Returns the object of type K.

- V getSecond ()

  Returns the object of type V.

Static Package Functions

- [static initializer]

Static Private Member Functions

- static native void initIDs ()

Private Attributes

- K first

  Stores an object of type K.

- V second

  Stores an object of type V.

11.33.1 Detailed Description

A pair of values of type K and V. An object of this class holds an ordered pair of values of type K and V.

Definition at line 30 of file Pair.java.

11.33.2 Member Function Documentation

11.33.2.1 parma_polyhedra_library::Pair< K, V >::[static initializer] () [inline, static, package]

11.33.2.2 K parma_polyhedra_library::Pair< K, V >::getFirst () [inline]

Returns the object of type K.

Definition at line 39 of file Pair.java.

11.33.2.3 V parma_polyhedra_library::Pair< K, V >::getSecond () [inline]

Returns the object of type V.

Definition at line 44 of file Pair.java.

11.33.2.4 static native void parma_polyhedra_library::Pair< K, V >::initIDs () [static, private]

11.33.3 Member Data Documentation

11.33.3.1 K parma_polyhedra_library::Pair< K, V >::first [private]

Stores an object of type K.

Definition at line 33 of file Pair.java.

11.33.3.2 V parma_polyhedra_library::Pair< K, V >::second [private]

Stores an object of type V.

Definition at line 36 of file Pair.java.

The documentation for this class was generated from the following file:

- Pair.java

11.34 parma_polyhedra_library::Parma_Polyhedra_Library Class Reference

A class collecting library-level functions.

Static Public Member Functions

Library initialization and finalization

- static native void initialize_library ()
  Initializes the Parma Polyhedra Library.
- static native void finalize_library ()
  Finalizes the Parma Polyhedra Library.

Version Checking

- static native int version_major ()
  Returns the major number of the PPL version.

- static native int version_minor ()
  Returns the minor number of the PPL version.
- static native int version_revision ()
  Returns the revision number of the PPL version.
- static native int version_beta ()
  Returns the beta number of the PPL version.
- static native String version ()
  Returns a string containing the PPL version.
- static native String banner ()
  Returns a string containing the PPL banner.

Floating-point rounding and precision settings.

- static native void set_rounding_for_PPL ()
  Sets the FPU rounding mode so that the PPL abstractions based on floating point numbers work correctly.
- static native void restore_pre_PPL_rounding ()
  Sets the FPU rounding mode as it was before initialization of the PPL.
- static native int irrational_precision ()
  Returns the precision parameter for irrational calculations.
- static native void set_irrational_precision (int p)
  Sets the precision parameter used for irrational calculations.

Timeout handling

- static native void set_timeout (int hsecs)
  Sets the timeout for computations whose completion could require an exponential amount of time.
- static native void reset_timeout ()
  Resets the timeout time so that the computation is not interrupted.
- static native void set_deterministic_timeout (int weight)
  Sets a threshold for computations whose completion could require an exponential amount of time.
- static native void reset_deterministic_timeout ()
  Resets the deterministic timeout so that the computation is not interrupted.

11.34.1 Detailed Description

A class collecting library-level functions.

Definition at line 144 of file Parma_Polyhedra_Library.java.

11.34.2 Member Function Documentation

11.34.2.1 static native String parma_polyhedra_library::Parma_Polyhedra_Library::banner () [static]

Returns a string containing the PPL banner.

The banner provides 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.

11.34.2.2 static native void parma_polyhedra_library::Parma_Polyhedra_Library::finalize_library () [static]

Finalizes the Parma Polyhedra Library.

This method must be called when work with the library is done. After finalization, no other library method can be called (except those in class Parma_Polyhedra_Library), unless the library is re-initialized by calling `initialize_library()`.

11.34.2.3 static native void parma_polyhedra_library::Parma_Polyhedra_Library::initialize_library () [static]

Initializes the Parma Polyhedra Library.

This method must be called after loading the library and before calling any other method from any other PPL package class.

11.34.2.4 static native int parma_polyhedra_library::Parma_Polyhedra_Library::irrational_precision () [static]

Returns the precision parameter for irrational calculations.

11.34.2.5 static native void parma_polyhedra_library::Parma_Polyhedra_Library::reset_deterministic_timeout () [static]

Resets the deterministic timeout so that the computation is not interrupted.

11.34.2.6 static native void parma_polyhedra_library::Parma_Polyhedra_Library::reset_timeout () [static]

Resets the timeout time so that the computation is not interrupted.

11.34.2.7 static native void parma_polyhedra_library::Parma_Polyhedra_Library::restore_pre_PPL_rounding () [static]

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.

11.34.2.8 static native void parma_polyhedra_library::Parma_Polyhedra_Library::set_deterministic_timeout (int *weight*) [static]

Sets a threshold for computations whose completion could require an exponential amount of time.

Parameters

weight The maximum computational weight allowed. It must be strictly greater than zero.

Computations taking exponential time will be interrupted some time after reaching the `weight` complexity threshold, by throwing a `Timeout_Exception` object. Otherwise, if the computation completes without being interrupted, then the deterministic timeout should be reset by calling `reset_deterministic_timeout()`.

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.).

Warning

The weight mechanism is under alpha testing. In particular, there is still no clear relation between the weight threshold and the actual computational complexity. As a consequence, client applications should be ready to reconsider the tuning of these weight thresholds when upgrading to newer version of the PPL.

11.34.2.9 static native void parma_polyhedra_library::Parma_Polyhedra_Library::set_irrational_precision (int *p*) [static]

Sets the precision parameter used for irrational calculations.

If `p` is less than or equal to `INT_MAX`, sets the precision parameter used for irrational calculations to `p`. Then, in the 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`.

11.34.2.10 static native void parma_polyhedra_library::Parma_Polyhedra_Library::set_rounding_for_PPL () [static]

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 been previously called.

11.34.2.11 static native void parma_polyhedra_library::Parma_Polyhedra_Library::set_timeout (int *hsecs*) [static]

Sets the timeout for computations whose completion could require an exponential amount of time.

Parameters

hsecs The number of hundreths of seconds. It must be strictly greater than zero.

Computations taking exponential time will be interrupted some time after `hsecs` hundreths of seconds have elapsed since the call to the timeout setting function, by throwing a `Timeout_Exception` object. Otherwise, if the computation completes without being interrupted, then the timeout should be reset by calling `reset_timeout()`.

11.34.2.12 static native String parma_polyhedra_library::Parma_Polyhedra_Library::version () [static]

Returns a string containing the PPL version.

11.34.2.13 static native int parma_polyhedra_library::Parma_Polyhedra_Library::version_beta () [static]

Returns the beta number of the PPL version.

11.34.2.14 static native int parma_polyhedra_library::Parma_Polyhedra_Library::version_major () [static]

Returns the major number of the PPL version.

11.34.2.15 static native int parma_polyhedra_library::Parma_Polyhedra_Library::version_minor () [static]

Returns the minor number of the PPL version.

11.34.2.16 static native int parma_polyhedra_library::Parma_Polyhedra_Library::version_revision () [static]

Returns the revision number of the PPL version.

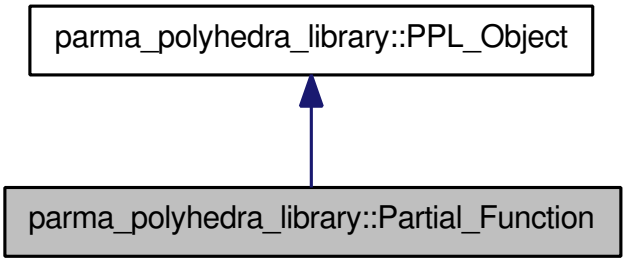
The documentation for this class was generated from the following file:

- Parma_Polyhedra_Library.java

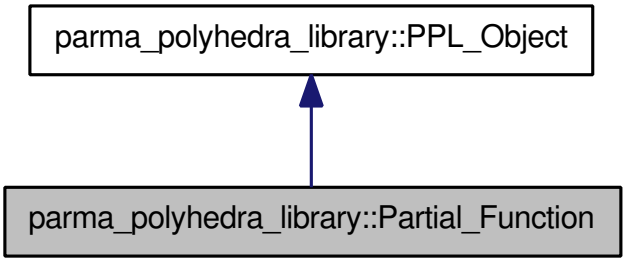
11.35 parma_polyhedra_library::Partial_Function Class Reference

A partial function on space dimension indices.

Inheritance diagram for parma_polyhedra_library::Partial_Function:



Collaboration diagram for parma_polyhedra_library::Partial_Function:



Public Member Functions

- Partial_Function ()

  Builds the empty map.

- native void insert (long i, long j)

  Inserts mapping from `i` to `j`.

- native boolean has_empty_codomain ()

  Returns `true` if and only if the partial function has an empty codomain (i.e., it is always undefined).

- native long max_in_codomain ()

  Returns the maximum value that belongs to the codomain of the partial function.

- native long maps (long i)

  If the partial function is defined on index `i`, returns its value.

- native void free ()

  Releases all resources managed by `this`, also resetting it to a null reference.

Protected Member Functions

- native void finalize ()

  Releases all resources managed by `this`.

Private Member Functions

- native void build_cpp_object ()

  Builds the underlying C++ object.

11.35.1 Detailed Description

A partial function on space dimension indices. This class is used in order to specify how space dimensions should be mapped by methods named `map_space_dimensions`.

Definition at line 31 of file Partial_Function.java.

11.35.2 Constructor & Destructor Documentation

11.35.2.1 parma_polyhedra_library::Partial_Function::Partial_Function () `[inline]`

Builds the empty map.

Definition at line 36 of file Partial_Function.java.

References build_cpp_object().

11.35.3 Member Function Documentation

11.35.3.1 native void parma_polyhedra_library::Partial_Function::build_cpp_object () `[private]`

Builds the underlying C++ object.

Referenced by Partial_Function().

11.35.3.2 native void parma_polyhedra_library::Partial_Function::finalize () `[protected]`

Releases all resources managed by `this`.

11.35.3.3 native void parma_polyhedra_library::Partial_Function::free ()

Releases all resources managed by `this`, also resetting it to a null reference.

11.35.3.4 native boolean parma_polyhedra_library::Partial_Function::has_empty_codomain ()

Returns `true` if and only if the partial function has an empty codomain (i.e., it is always undefined).

This method will always be called before the other methods of the interface. Moreover, if `true` is returned, then none of the other interface methods will be called.

11.35.3.5 native void parma_polyhedra_library::Partial_Function::insert (long *i*, long *j*)

Inserts mapping from `i` to `j`.

11.35.3.6 native long parma_polyhedra_library::Partial_Function::maps (long *i*)

If the partial function is defined on index `i`, returns its value.

The function returns a negative value if the partial function is not defined on domain value `i`.

11.35.3.7 native long parma_polyhedra_library::Partial_Function::max_in_codomain ()

Returns the maximum value that belongs to the codomain of the partial function.

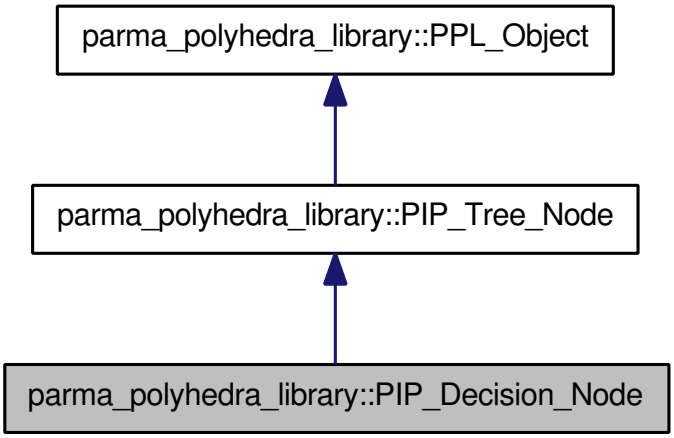
The documentation for this class was generated from the following file:

- Partial_Function.java

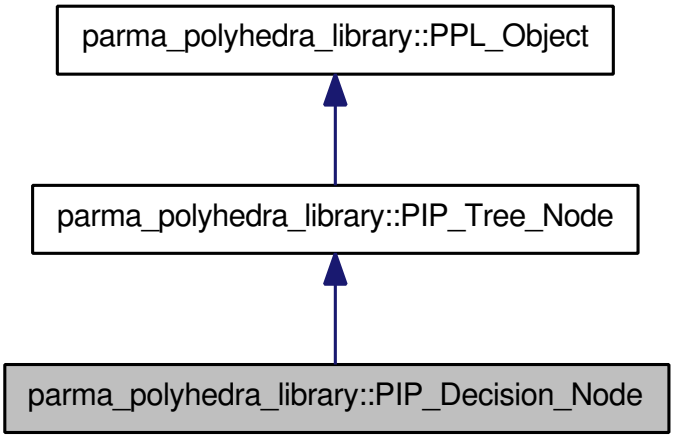
11.36 parma_polyhedra_library::PIP_Decision_Node Class Reference

An internal node of the PIP solution tree.

Inheritance diagram for parma_polyhedra_library::PIP_Decision_Node:



Collaboration diagram for parma_polyhedra_library::PIP_Decision_Node:



Public Member Functions

- native PIP_Tree_Node child_node (boolean branch)

  Returns the true branch (if `branch` *is true) or the false branch (if* `branch` *is false) of* `this`*.*

11.36.1 Detailed Description

An internal node of the PIP solution tree.

Definition at line 31 of file PIP_Decision_Node.java.

11.36.2 Member Function Documentation

11.36.2.1 native PIP_Tree_Node parma_polyhedra_library::PIP_Decision_Node::child_node (boolean *branch*)

Returns the true branch (if `branch` is true) or the false branch (if `branch` is false) of `this`.

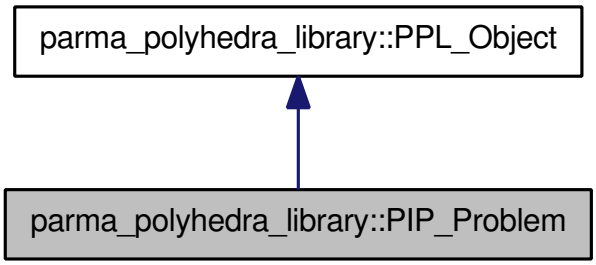
The documentation for this class was generated from the following file:

- PIP_Decision_Node.java

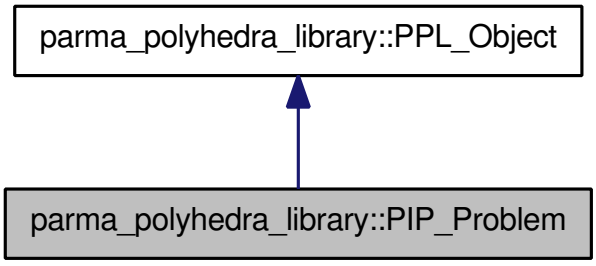
11.37 parma_polyhedra_library::PIP_Problem Class Reference

A Parametric Integer Programming problem.

Inheritance diagram for parma_polyhedra_library::PIP_Problem:



Collaboration diagram for parma_polyhedra_library::PIP_Problem:



Public Member Functions

- PIP_Problem (long dim)

  Builds a trivial PIP problem.

- PIP_Problem (long dim, Constraint_System cs, Variables_Set params)

  Builds a PIP problem from a sequence of constraints.

- PIP_Problem (PIP_Problem y)

  Builds a copy of y.

- native void free ()

  Releases all resources managed by `this`, also resetting it to a null reference.

Functions that Do Not Modify the PIP_Problem

- native long max_space_dimension ()

  Returns the maximum space dimension an PIP_Problem can handle.

- native long space_dimension ()

  Returns the space dimension of the PIP problem.

- native long number_of_parameter_space_dimensions ()

  Returns the number of parameter space dimensions of the PIP problem.

- native Variables_Set parameter_space_dimensions ()

  Returns all the parameter space dimensions of problem `pip`.

- native long get_big_parameter_dimension ()

  Returns the big parameter dimension of PIP problem `pip`.

- native long number_of_constraints ()

  Returns the number of constraints defining the feasible region of `pip`.

- native Constraint constraint_at_index (long dim)

  Returns the `i-th` constraint defining the feasible region of the PIP problem `pip`.

- native Constraint_System constraints ()

  Returns the constraints .

- native String ascii_dump ()

  Returns an ascii formatted internal representation of `this`.

- native String toString ()

  Returns a string representation of `this`.

- native long total_memory_in_bytes ()

  Returns the size in bytes of the memory occupied by the underlying C++ object.

- native long external_memory_in_bytes ()

  Returns the size in bytes of the memory managed by the underlying C++ object.

- native boolean OK ()

  Returns true if the pip problem is well formed, i.e., if it satisfies all its implementation invariants; returns 0 and perhaps makes some noise if broken. Useful for debugging purposes.

Functions that May Modify the PIP_Problem

- native void clear ()

  Resets `this` to be equal to the trivial PIP problem.

- native void add_space_dimensions_and_embed (long pip_vars, long pip_params)

Adds `pip_vars + pip_params` new space dimensions and embeds the PIP problem in the new vector space.

- native void add_to_parameter_space_dimensions (Variables_Set vars)
  Sets the space dimensions in `vars` to be parameter dimensions of the PIP problem.
- native void set_big_parameter_dimension (long d)
  Sets the big parameter dimension of PIP problem to `d`.
- native void add_constraint (Constraint c)
  Adds a copy of constraint `c` to the PIP problem.
- native void add_constraints (Constraint_System cs)
  Adds a copy of the constraints in `cs` to the PIP problem.

Computing the Solution of the PIP_Problem

- native boolean is_satisfiable ()
  Checks satisfiability of `*this`.
- native PIP_Problem_Status solve ()
  Optimizes the PIP problem.
- native PIP_Tree_Node solution ()
  Returns a solution for the PIP problem, if it exists.
- native PIP_Tree_Node optimizing_solution ()
  Returns an optimizing solution for the PIP problem, if it exists.

Querying/Setting Control Parameters

- native PIP_Problem_Control_Parameter_Value get_pip_problem_control_parameter (PIP_Problem_Control_Parameter_Name name)
  Returns the value of control parameter `name`.
- native void set_pip_problem_control_parameter (PIP_Problem_Control_Parameter_Value value)
  Sets control parameter `value`.

Protected Member Functions

- native void finalize ()
  Releases all resources managed by `this`.

Private Member Functions

- native void build_cpp_object (long dim)
  Builds the underlying C++ object.

- native void build_cpp_object (long dim, Constraint_System cs, Variables_Set vars)

  Builds the underlying C++ object.

- native void build_cpp_object (PIP_Problem y)

  Builds the underlying C++ object.

11.37.1 Detailed Description

A Parametric Integer Programming problem. An object of this class encodes a parametric integer (linear) programming problem. The PIP problem is specified by providing:

- the dimension of the vector space;
- the subset of those dimensions of the vector space that are interpreted as integer parameters (the other space dimensions are interpreted as non-parameter integer variables);
- a finite set of linear equality and (strict or non-strict) inequality constraints involving variables and/or parameters; these constraints are used to define:
  - the *feasible region*, if they involve one or more problem variable (and maybe some parameters);
  - the *initial context*, if they only involve the parameters;
- optionally, the so-called *big parameter*, i.e., a problem parameter to be considered arbitrarily big.

Note that all problem variables and problem parameters are assumed to take non-negative integer values, so that there is no need to specify non-negativity constraints.

The class provides support for the (incremental) solution of the PIP problem based on variations of the revised simplex method and on Gomory cut generation techniques.

The solution for a PIP problem is the lexicographic minimum of the integer points of the feasible region, expressed in terms of the parameters. As the problem to be solved only involves non-negative variables and parameters, the problem will always be either unfeasible or optimizable.

As the feasibility and the solution value of a PIP problem depend on the values of the parameters, the solution is a binary decision tree, dividing the context parameter set into subsets. The tree nodes are of two kinds:

- *Decision* nodes. These are internal tree nodes encoding one or more linear tests on the parameters; if all the tests are satisfied, then the solution is the node's *true* child; otherwise, the solution is the node's *false* child;
- *Solution* nodes. These are leaf nodes in the tree, encoding the solution of the problem in the current context subset, where each variable is defined in terms of a linear expression of the parameters. Solution nodes also optionally embed a set of parameter constraints: if all these constraints are satisfied, the solution is described by the node, otherwise the problem has no solution.

It may happen that a decision node has no *false* child. This means that there is no solution if at least one of the corresponding constraints is not satisfied. Decision nodes having two or more linear tests on the parameters cannot have a *false* child. Decision nodes always have a *true* child.

Both kinds of tree nodes may also contain the definition of extra parameters which are artificially introduced by the solver to enforce an integral solution. Such artificial parameters are defined by the integer division of a linear expression on the parameters by an integer coefficient.

By exploiting the incremental nature of the solver, it is possible to reuse part of the computational work already done when solving variants of a given PIP_Problem: currently, incremental resolution supports the addition of space dimensions, the addition of parameters and the addition of constraints.

Definition at line 97 of file PIP_Problem.java.

11.37.2 Constructor & Destructor Documentation

11.37.2.1 parma_polyhedra_library::PIP_Problem::PIP_Problem (long *dim*) [inline]

Builds a trivial PIP problem.

A trivial PIP problem requires to compute the lexicographic minimum on a vector space under no constraints and with no parameters: due to the implicit non-negativity constraints, the origin of the vector space is an optimal solution.

Parameters

dim The dimension of the vector space enclosing `*this` (optional argument with default value 0).

Exceptions

std::length_error Thrown if `dim` exceeds `max_space_dimension()`.

Definition at line 113 of file PIP_Problem.java.

References build_cpp_object().

11.37.2.2 parma_polyhedra_library::PIP_Problem::PIP_Problem (long *dim*, Constraint_System *cs*, Variables_Set *params*) [inline]

Builds a PIP problem from a sequence of constraints.

Builds a PIP problem having space dimension `dim` from the constraint system cs; the dimensions `vars` are interpreted as parameters.

Definition at line 123 of file PIP_Problem.java.

References build_cpp_object().

11.37.2.3 parma_polyhedra_library::PIP_Problem::PIP_Problem (PIP_Problem *y*) [inline]

Builds a copy of `y`.

Definition at line 128 of file PIP_Problem.java.

References build_cpp_object().

11.37.3 Member Function Documentation

11.37.3.1 native void parma_polyhedra_library::PIP_Problem::add_constraint (Constraint *c*)

Adds a copy of constraint `c` to the PIP problem.

Exceptions

std::invalid_argument Thrown if the constraint `c` is a strict inequality or if its space dimension is strictly greater than the space dimension of `this`.

11.37.3.2 native void parma_polyhedra_library::PIP_Problem::add_constraints (Constraint_System *cs*)

Adds a copy of the constraints in `cs` to the PIP problem.

Exceptions

std::invalid_argument Thrown if the constraint system `cs` contains any strict inequality or if its space dimension is strictly greater than the space dimension of `*this`.

11.37.3.3 native void parma_polyhedra_library::PIP_Problem::add_space_dimensions_and_embed (long *pip_vars*, long *pip_params*)

Adds `pip_vars + pip_params` new space dimensions and embeds the PIP problem in the new vector space.

Parameters

pip_vars The number of space dimensions to add that are interpreted as PIP problem variables (i.e., non parameters). These are added before adding the `pip_params` parameters.

pip_params The number of space dimensions to add that are interpreted as PIP problem parameters. These are added after having added the `pip_vars` problem variables.

The new space dimensions will be those having the highest indexes in the new PIP problem; they are initially unconstrained.

11.37.3.4 native void parma_polyhedra_library::PIP_Problem::add_to_parameter_space_dimensions (Variables_Set *vars*)

Sets the space dimensions in `vars` to be parameter dimensions of the PIP problem.

11.37.3.5 native String parma_polyhedra_library::PIP_Problem::ascii_dump ()

Returns an ascii formatted internal representation of `this`.

11.37.3.6 native void parma_polyhedra_library::PIP_Problem::build_cpp_object (PIP_Problem *y*) [private]

Builds the underlying C++ object.

11.37.3.7 native void parma_polyhedra_library::PIP_Problem::build_cpp_object (long *dim*, Constraint_System *cs*, Variables_Set *vars*) [private]

Builds the underlying C++ object.

11.37.3.8 native void parma_polyhedra_library::PIP_Problem::build_cpp_object (long *dim*) [private]

Builds the underlying C++ object.

Referenced by PIP_Problem().

11.37.3.9 native void parma_polyhedra_library::PIP_Problem::clear ()

Resets `this` to be equal to the trivial PIP problem.

The space dimension is reset to 0.

11.37.3.10 native Constraint parma_polyhedra_library::PIP_Problem::constraint_at_index (long *dim*)

Returns the `i-th` constraint defining the feasible region of the PIP problem `pip`.

11.37.3.11 native Constraint_System parma_polyhedra_library::PIP_Problem::constraints ()

Returns the constraints .

11.37.3.12 native long parma_polyhedra_library::PIP_Problem::external_memory_in_bytes ()

Returns the size in bytes of the memory managed by the underlying C++ object.

11.37.3.13 native void parma_polyhedra_library::PIP_Problem::finalize () [protected]

Releases all resources managed by `this`.

11.37.3.14 native void parma_polyhedra_library::PIP_Problem::free ()

Releases all resources managed by `this`, also resetting it to a null reference.

11.37.3.15 native long parma_polyhedra_library::PIP_Problem::get_big_parameter_dimension ()

Returns the big parameter dimension of PIP problem `pip`.

11.37.3.16 native PIP_Problem_Control_Parameter_Value parma_polyhedra_library::PIP_Problem::get_pip_problem_control_parameter (PIP_Problem_Control_Parameter_Name *name*)

Returns the value of control parameter `name`.

11.37.3.17 native boolean parma_polyhedra_library::PIP_Problem::is_satisfiable ()

Checks satisfiability of `*this`.

Returns

`true` if and only if the PIP problem is satisfiable.

11.37.3.18 native long parma_polyhedra_library::PIP_Problem::max_space_dimension ()

Returns the maximum space dimension an PIP_Problem can handle.

11.37.3.19 native long parma_polyhedra_library::PIP_Problem::number_of_constraints ()

Returns the number of constraints defining the feasible region of `pip`.

11.37.3.20 native long parma_polyhedra_library::PIP_Problem::number_of_parameter_space_dimensions ()

Returns the number of parameter space dimensions of the PIP problem.

11.37.3.21 native boolean parma_polyhedra_library::PIP_Problem::OK ()

Returns true if the pip problem is well formed, i.e., if it satisfies all its implementation invariants; returns 0 and perhaps makes some noise if broken. Useful for debugging purposes.

11.37.3.22 native PIP_Tree_Node parma_polyhedra_library::PIP_Problem::optimizing_solution ()

Returns an optimizing solution for the PIP problem, if it exists.

11.37.3.23 native Variables_Set parma_polyhedra_library::PIP_Problem::parameter_space_dimensions ()

Returns all the parameter space dimensions of problem `pip`.

11.37.3.24 native void parma_polyhedra_library::PIP_Problem::set_big_parameter_dimension (long *d*)

Sets the big parameter dimension of PIP problem to `d`.

11.37.3.25 native void parma_polyhedra_library::PIP_Problem::set_pip_problem_control_parameter (PIP_Problem_Control_Parameter_Value *value*)

Sets control parameter `value`.

11.37.3.26 native PIP_Tree_Node parma_polyhedra_library::PIP_Problem::solution ()

Returns a solution for the PIP problem, if it exists.

11.37.3.27 native PIP_Problem_Status parma_polyhedra_library::PIP_Problem::solve ()

Optimizes the PIP problem.

Solves the PIP problem, returning an exit status.

Returns

`UNFEASIBLE_PIP_PROBLEM` if the PIP problem is not satisfiable; `OPTIMIZED_PIP_PROBLEM` if the PIP problem admits an optimal solution.

11.37.3.28 native long parma_polyhedra_library::PIP_Problem::space_dimension ()

Returns the space dimension of the PIP problem.

11.37.3.29 native String parma_polyhedra_library::PIP_Problem::toString ()

Returns a string representation of `this`.

11.37.3.30 native long parma_polyhedra_library::PIP_Problem::total_memory_in_bytes ()

Returns the size in bytes of the memory occupied by the underlying C++ object.

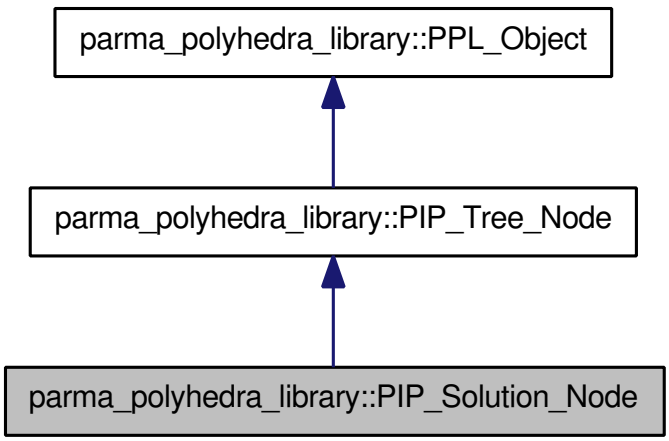
The documentation for this class was generated from the following file:

- PIP_Problem.java

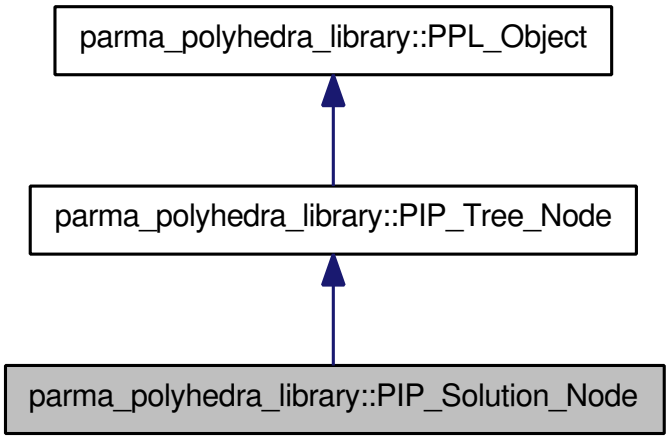
11.38 parma_polyhedra_library::PIP_Solution_Node Class Reference

A leaf node of the PIP solution tree.

Inheritance diagram for parma_polyhedra_library::PIP_Solution_Node:



Collaboration diagram for parma_polyhedra_library::PIP_Solution_Node:



Public Member Functions

- native Linear_Expression parametric_values (Variable var)

  Returns the parametric expression of the values of variable `var` *in solution node* `this`.

11.38.1 Detailed Description

A leaf node of the PIP solution tree.

Definition at line 31 of file PIP_Solution_Node.java.

11.38.2 Member Function Documentation

11.38.2.1 native Linear_Expression parma_polyhedra_library::PIP_Solution_Node::parametric_values (Variable *var*)

Returns the parametric expression of the values of variable `var` in solution node `this`.

The returned parametric expression will only refer to (problem or artificial) parameters.

Parameters

var The variable being queried.

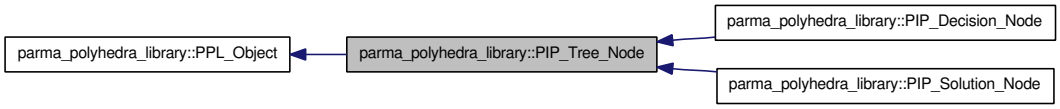
The documentation for this class was generated from the following file:

- PIP_Solution_Node.java

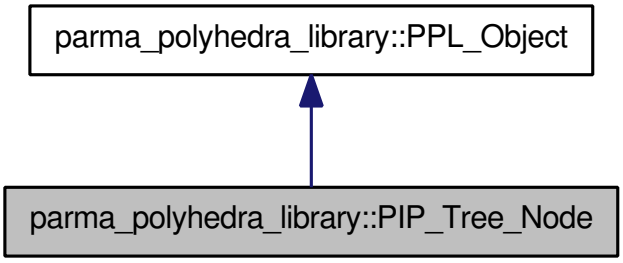
11.39 parma_polyhedra_library::PIP_Tree_Node Class Reference

A node of the PIP solution tree.

Inheritance diagram for parma_polyhedra_library::PIP_Tree_Node:



Collaboration diagram for parma_polyhedra_library::PIP_Tree_Node:



Public Member Functions

- native PIP_Solution_Node as_solution ()

  Returns the solution node if `this` *is a solution node, and 0 otherwise.*

- native PIP_Decision_Node as_decision ()

  Returns the decision node if `this` *is a decision node, and 0 otherwise.*

- native boolean OK ()

  Returns true if the pip tree is well formed, i.e., if it satisfies all its implementation invariants; returns 0 and perhaps makes some noise if broken. Useful for debugging purposes.

- native long number_of_artificials ()

  Returns the number of artificial parameters in the PIP_Tree_Node.

- native Artificial_Parameter_Sequence artificials ()

  Returns the sequence of (Java) artificial parameters in the PIP_Tree_Node.

- native Constraint_System constraints ()

  Returns the system of parameter constraints controlling the PIP_Tree_Node.

- native String toString ()

  Returns a string representation of `this`.

11.39.1 Detailed Description

A node of the PIP solution tree. This is the base class for the nodes of the binary trees representing the solutions of PIP problems. From this one, two classes are derived:

- PIP_Decision_Node, for the internal nodes of the tree;
- PIP_Solution_Node, for the leaves of the tree.

Definition at line 37 of file PIP_Tree_Node.java.

11.39.2 Member Function Documentation

11.39.2.1 native Artificial_Parameter_Sequence parma_polyhedra_library::PIP_Tree_Node::artificials ()

Returns the sequence of (Java) artificial parameters in the PIP_Tree_Node.

11.39.2.2 native PIP_Decision_Node parma_polyhedra_library::PIP_Tree_Node::as_decision ()

Returns the decision node if `this` is a decision node, and 0 otherwise.

11.39.2.3 native PIP_Solution_Node parma_polyhedra_library::PIP_Tree_Node::as_solution ()

Returns the solution node if `this` is a solution node, and 0 otherwise.

11.39.2.4 native Constraint_System parma_polyhedra_library::PIP_Tree_Node::constraints ()

Returns the system of parameter constraints controlling the PIP_Tree_Node.

The indices in the constraints are the same as the original variables and parameters. Coefficients in indices corresponding to variables always are zero.

11.39.2.5 native long parma_polyhedra_library::PIP_Tree_Node::number_of_artificials ()

Returns the number of artificial parameters in the PIP_Tree_Node.

11.39.2.6 native boolean parma_polyhedra_library::PIP_Tree_Node::OK ()

Returns true if the pip tree is well formed, i.e., if it satisfies all its implementation invariants; returns 0 and perhaps makes some noise if broken. Useful for debugging purposes.

11.39.2.7 native String parma_polyhedra_library::PIP_Tree_Node::toString ()

Returns a string representation of `this`.

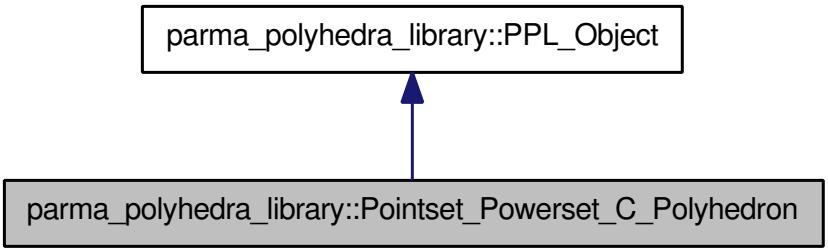
The documentation for this class was generated from the following file:

- PIP_Tree_Node.java

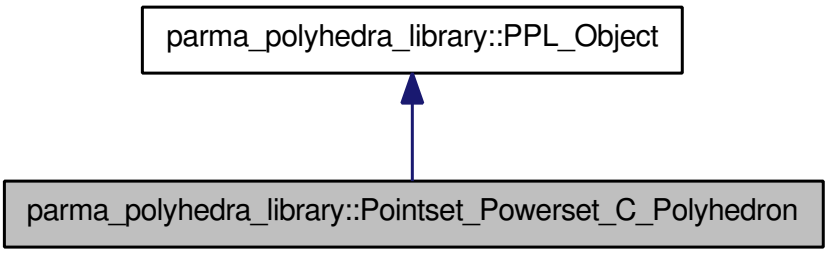
11.40 parma_polyhedra_library::Pointset_Powerset_C_Polyhedron Class Reference

A powerset of C_Polyhedron objects.

Inheritance diagram for parma_polyhedra_library::Pointset_Powerset_C_Polyhedron:



Collaboration diagram for parma_polyhedra_library::Pointset_Powerset_C_Polyhedron:



Public Member Functions

Ad Hoc Functions for Pointset_Powerset domains

- native void omega_reduce ()
  Drops from the sequence of disjuncts in `this` all the non-maximal elements, so that a non-redundant powerset if obtained.
- native long size ()
  Returns the number of disjuncts.
- native boolean geometrically_covers (Pointset_Powerset_C_Polyhedron y)
  Returns `true` if and only if `this` geometrically covers `y`.
- native boolean geometrically_equals (Pointset_Powerset_C_Polyhedron y)
  Returns `true` if and only if `this` is geometrically equal to `y`.
- native Pointset_Powerset_C_Polyhedron_Iterator begin_iterator ()

Returns an iterator referring to the beginning of the sequence of disjuncts of `this`.

- native Pointset_Powerset_C_Polyhedron_Iterator end_iterator ()
  Returns an iterator referring to past the end of the sequence of disjuncts of `this`.
- native void add_disjunct (C_Polyhedron d)
  Adds to `this` *a copy of disjunct* `d`.
- native void drop_disjunct (Pointset_Powerset_C_Polyhedron_Iterator iter)
  Drops from `this` *the disjunct referred by* `iter`*; returns an iterator referring to the disjunct following the dropped one.*
- native void drop_disjuncts (Pointset_Powerset_C_Polyhedron_Iterator first, Pointset_Powerset_C_Polyhedron_Iterator last)
  Drops from `this` *all the disjuncts from* `first` *to* `last` *(excluded).*
- native void pairwise_reduce ()
  Modifies `this` *by (recursively) merging together the pairs of disjuncts whose upper-bound is the same as their set-theoretical union.*

11.40.1 Detailed Description

A powerset of C_Polyhedron objects. The powerset domains can be instantiated by taking as a base domain any fixed semantic geometric description (C and NNC polyhedra, BD and octagonal shapes, boxes and grids). An element of the powerset domain represents a disjunctive collection of base objects (its disjuncts), all having the same space dimension.

Besides the methods that are available in all semantic geometric descriptions (whose documentation is not repeated here), the powerset domain also provides several ad hoc methods. In particular, the iterator types allow for the examination and manipulation of the collection of disjuncts.

Definition at line 1186 of file Fake_Class_for_Doxygen.java.

11.40.2 Member Function Documentation

11.40.2.1 native void parma_polyhedra_library::Pointset_Powerset_C_Polyhedron::add_disjunct (C_Polyhedron *d*)

Adds to `this` a copy of disjunct `d`.

11.40.2.2 native Pointset_Powerset_C_Polyhedron_Iterator parma_polyhedra_library::Pointset_Powerset_C_Polyhedron::begin_iterator ()

Returns an iterator referring to the beginning of the sequence of disjuncts of `this`.

11.40.2.3 native void parma_polyhedra_library::Pointset_Powerset_C_Polyhedron::drop_disjunct (Pointset_Powerset_C_Polyhedron_Iterator *iter*)

Drops from `this` the disjunct referred by `iter`; returns an iterator referring to the disjunct following the dropped one.

11.40.2.4 native void parma_polyhedra_library::Pointset_Powerset_C_-Polyhedron::drop_disjuncts (Pointset_Powerset_C_Polyhedron_Iterator *first*, Pointset_Powerset_C_Polyhedron_Iterator *last*)

Drops from `this` all the disjuncts from `first` to `last` (excluded).

11.40.2.5 native Pointset_Powerset_C_Polyhedron_Iterator parma_polyhedra_library::Pointset_-Powerset_C_Polyhedron::end_iterator ()

Returns an iterator referring to past the end of the sequence of disjuncts of `this`.

11.40.2.6 native boolean parma_polyhedra_library::Pointset_Powerset_C_-Polyhedron::geometrically_covers (Pointset_Powerset_C_Polyhedron *y*)

Returns `true` if and only if `this` geometrically covers `y`.

11.40.2.7 native boolean parma_polyhedra_library::Pointset_Powerset_C_-Polyhedron::geometrically_equals (Pointset_Powerset_C_Polyhedron *y*)

Returns `true` if and only if `this` is geometrically equal to `y`.

11.40.2.8 native void parma_polyhedra_library::Pointset_Powerset_C_Polyhedron::omega_-reduce ()

Drops from the sequence of disjuncts in `this` all the non-maximal elements, so that a non-redundant powerset if obtained.

11.40.2.9 native void parma_polyhedra_library::Pointset_Powerset_C_Polyhedron::pairwise_-reduce ()

Modifies `this` by (recursively) merging together the pairs of disjuncts whose upper-bound is the same as their set-theoretical union.

11.40.2.10 native long parma_polyhedra_library::Pointset_Powerset_C_Polyhedron::size ()

Returns the number of disjuncts.

If present, Omega-redundant elements will be counted too.

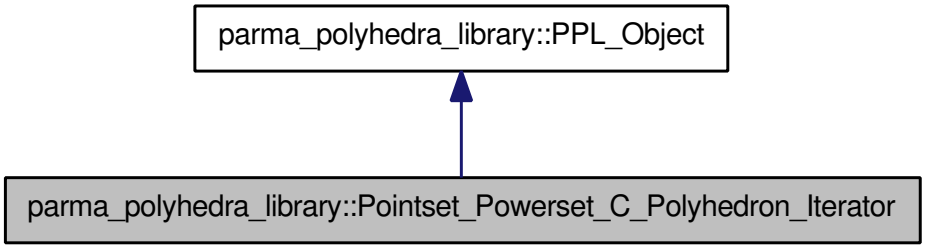
The documentation for this class was generated from the following file:

- Fake_Class_for_Doxygen.java

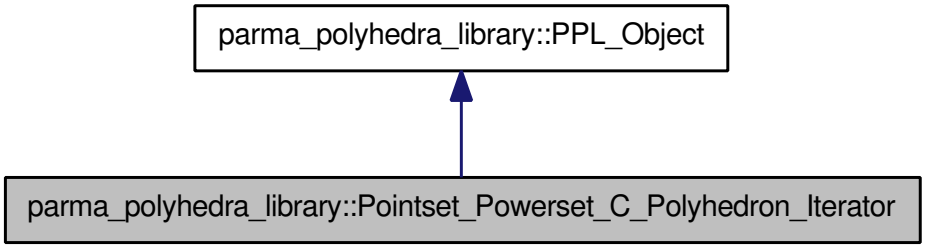
11.41 parma_polyhedra_library::Pointset_Powerset_C_Polyhedron_Iterator Class Reference

An iterator class for the disjuncts of a Pointset_Powerset_C_Polyhedron.

Inheritance diagram for parma_polyhedra_library::Pointset_Powerset_C_Polyhedron_Iterator:



Collaboration diagram for parma_polyhedra_library::Pointset_Powerset_C_Polyhedron_Iterator:



Public Member Functions

- Pointset_Powerset_C_Polyhedron_Iterator (Pointset_Powerset_C_Polyhedron_Iterator y)

  Builds a copy of iterator `y`.

- native boolean equals (Pointset_Powerset_C_Polyhedron_Iterator itr)

  Returns `true` *if and only if* `this` *and* `itr` *are equal.*

- native void next ()

  Modifies `this` *so that it refers to the next disjunct.*

- native void prev ()

  Modifies `this` *so that it refers to the previous disjunct.*

- native C_Polyhedron get_disjunct ()

  Returns the disjunct referenced by `this`*.*

- native void free ()

  Releases resources and resets `this` *to a null reference.*

Protected Member Functions

- native void finalize ()

  Releases the resources managed by `this`*.*

11.41.1 Detailed Description

An iterator class for the disjuncts of a Pointset_Powerset_C_Polyhedron.

Definition at line 1257 of file Fake_Class_for_Doxygen.java.

11.41.2 Constructor & Destructor Documentation

11.41.2.1 parma_polyhedra_library::Pointset_Powerset_C_Polyhedron_Iterator::Pointset_Powerset_C_Polyhedron_Iterator (Pointset_Powerset_C_Polyhedron_Iterator *y*)

Builds a copy of iterator `y`.

11.41.3 Member Function Documentation

11.41.3.1 native boolean parma_polyhedra_library::Pointset_Powerset_C_Polyhedron_Iterator::equals (Pointset_Powerset_C_Polyhedron_Iterator *itr*)

Returns `true` if and only if `this` and `itr` are equal.

11.41.3.2 native void parma_polyhedra_library::Pointset_Powerset_C_Polyhedron_Iterator::finalize () [protected]

Releases the resources managed by `this`.

11.41.3.3 native void parma_polyhedra_library::Pointset_Powerset_C_Polyhedron_Iterator::free ()

Releases resources and resets `this` to a null reference.

11.41.3.4 native C_Polyhedron parma_polyhedra_library::Pointset_Powerset_C_Polyhedron_Iterator::get_disjunct ()

Returns the disjunct referenced by `this`.

Warning

On exit, the C_Polyhedron disjunct is still owned by the powerset object: any function call on the owning powerset object may invalidate it. Moreover, the disjunct is meant to be immutable and should not be modified in any way (its resources will be released when deleting the owning powerset). If really needed, the disjunct may be copied into a new object, which will be under control of the user.

11.41.3.5 native void parma_polyhedra_library::Pointset_Powerset_C_Polyhedron_Iterator::next ()

Modifies `this` so that it refers to the next disjunct.

11.41.3.6 native void parma_polyhedra_library::Pointset_Powerset_C_Polyhedron_Iterator::prev ()

Modifies `this` so that it refers to the previous disjunct.

The documentation for this class was generated from the following file:

- Fake_Class_for_Doxygen.java

11.42 parma_polyhedra_library::Poly_Con_Relation Class Reference

The relation between a polyhedron and a constraint.

Public Member Functions

- Poly_Con_Relation (int val)

  Constructs from a integer value.

- boolean implies (Poly_Con_Relation y)

  True if and only if `*this` implies `y`.

Static Public Member Functions

- static Poly_Con_Relation nothing ()

  The assertion that says nothing.

- static Poly_Con_Relation is_disjoint ()

  The polyhedron and the set of points satisfying the constraint are disjoint.

- static Poly_Con_Relation strictly_intersects ()

  The polyhedron intersects the set of points satisfying the constraint, but it is not included in it.

- static Poly_Con_Relation is_included ()

  The polyhedron is included in the set of points satisfying the constraint.

- static Poly_Con_Relation saturates ()

  The polyhedron is included in the set of points saturating the constraint.

Static Public Attributes

- static final int NOTHING = 0
- static final int IS_DISJOINT = 1
- static final int STRICTLY_INTERSECTS = 2
- static final int IS_INCLUDED = 4
- static final int SATURATES = 8

Static Package Functions

- [static initializer]

Static Private Member Functions

- static native void initIDs ()

Private Attributes

- int mask_value

  Holds the value of the possible relations.

11.42.1 Detailed Description

The relation between a polyhedron and a constraint. This class implements conjunctions of assertions on the relation between a polyhedron and a constraint.

Definition at line 31 of file Poly_Con_Relation.java.

11.42.2 Constructor & Destructor Documentation

11.42.2.1 parma_polyhedra_library::Poly_Con_Relation::Poly_Con_Relation (int *val*) [inline]

Constructs from a integer value.

Definition at line 43 of file Poly_Con_Relation.java.

References mask_value.

Referenced by is_disjoint(), is_included(), nothing(), saturates(), and strictly_intersects().

11.42.3 Member Function Documentation

11.42.3.1 parma_polyhedra_library::Poly_Con_Relation::[static initializer] () [inline, static, package]

11.42.3.2 boolean parma_polyhedra_library::Poly_Con_Relation::implies (Poly_Con_Relation *y*) [inline]

True if and only if `*this` implies `y`.

Definition at line 85 of file Poly_Con_Relation.java.

References mask_value.

11.42.3.3 static native void parma_polyhedra_library::Poly_Con_Relation::initIDs () [static, private]

11.42.3.4 static Poly_Con_Relation parma_polyhedra_library::Poly_Con_Relation::is_disjoint () [inline, static]

The polyhedron and the set of points satisfying the constraint are disjoint.

Definition at line 56 of file Poly_Con_Relation.java.

References Poly_Con_Relation().

11.42.3.5 static Poly_Con_Relation parma_polyhedra_library::Poly_Con_Relation::is_included () [inline, static]

The polyhedron is included in the set of points satisfying the constraint.

Definition at line 72 of file Poly_Con_Relation.java.

References Poly_Con_Relation().

11.42.3.6 static Poly_Con_Relation parma_polyhedra_library::Poly_Con_Relation::nothing () [inline, static]

The assertion that says nothing.

Definition at line 48 of file Poly_Con_Relation.java.

References Poly_Con_Relation().

11.42.3.7 static Poly_Con_Relation parma_polyhedra_library::Poly_Con_Relation::saturates () [inline, static]

The polyhedron is included in the set of points saturating the constraint.

Definition at line 80 of file Poly_Con_Relation.java.

References Poly_Con_Relation().

11.42.3.8 static Poly_Con_Relation parma_polyhedra_library::Poly_Con_Relation::strictly_intersects () [inline, static]

The polyhedron intersects the set of points satisfying the constraint, but it is not included in it.

Definition at line 64 of file Poly_Con_Relation.java.

References Poly_Con_Relation().

11.42.4 Member Data Documentation

11.42.4.1 final int parma_polyhedra_library::Poly_Con_Relation::IS_DISJOINT = 1 [static]

Definition at line 34 of file Poly_Con_Relation.java.

11.42.4.2 final int parma_polyhedra_library::Poly_Con_Relation::IS_INCLUDED = 4 [static]

Definition at line 36 of file Poly_Con_Relation.java.

11.42.4.3 int parma_polyhedra_library::Poly_Con_Relation::mask_value [private]

Holds the value of the possible relations.

Definition at line 40 of file Poly_Con_Relation.java.

Referenced by implies(), and Poly_Con_Relation().

11.42.4.4 final int parma_polyhedra_library::Poly_Con_Relation::NOTHING = 0 [static]

Definition at line 33 of file Poly_Con_Relation.java.

11.42.4.5 final int parma_polyhedra_library::Poly_Con_Relation::SATURATES = 8 [static]

Definition at line 37 of file Poly_Con_Relation.java.

11.42.4.6 final int parma_polyhedra_library::Poly_Con_Relation::STRICTLY_INTERSECTS = 2 [static]

Definition at line 35 of file Poly_Con_Relation.java.

The documentation for this class was generated from the following file:

- Poly_Con_Relation.java

11.43 parma_polyhedra_library::Poly_Gen_Relation Class Reference

The relation between a polyhedron and a generator.

Public Member Functions

- Poly_Gen_Relation (int val)

  Constructs from a integer value.

- boolean implies (Poly_Gen_Relation y)

  True if and only if `*this` implies `y`.

Static Public Member Functions

- static Poly_Gen_Relation nothing ()

  The assertion that says nothing.

- static Poly_Gen_Relation subsumes ()

  Adding the generator would not change the polyhedron.

Static Public Attributes

- static final int NOTHING = 0
- static final int SUBSUMES = 1

Static Package Functions

- [static initializer]

Static Private Member Functions

- static native void initIDs ()

Private Attributes

- int mask_value

  Holds the value of the possible relations.

11.43.1 Detailed Description

The relation between a polyhedron and a generator. This class implements conjunctions of assertions on the relation between a polyhedron and a generator.

Definition at line 31 of file Poly_Gen_Relation.java.

11.43.2 Constructor & Destructor Documentation

11.43.2.1 parma_polyhedra_library::Poly_Gen_Relation::Poly_Gen_Relation (int *val*) `[inline]`

Constructs from a integer value.

Definition at line 40 of file Poly_Gen_Relation.java.

References mask_value.

Referenced by nothing(), and subsumes().

11.43.3 Member Function Documentation

11.43.3.1 parma_polyhedra_library::Poly_Gen_Relation::[static initializer] () `[inline, static, package]`

11.43.3.2 boolean parma_polyhedra_library::Poly_Gen_Relation::implies (Poly_Gen_Relation *y*) [inline]

True if and only if `*this` implies `y`.

Definition at line 55 of file Poly_Gen_Relation.java.

References mask_value.

11.43.3.3 static native void parma_polyhedra_library::Poly_Gen_Relation::initIDs () [static, private]

11.43.3.4 static Poly_Gen_Relation parma_polyhedra_library::Poly_Gen_Relation::nothing () [inline, static]

The assertion that says nothing.

Definition at line 45 of file Poly_Gen_Relation.java.

References Poly_Gen_Relation().

11.43.3.5 static Poly_Gen_Relation parma_polyhedra_library::Poly_Gen_Relation::subsumes () [inline, static]

Adding the generator would not change the polyhedron.

Definition at line 50 of file Poly_Gen_Relation.java.

References Poly_Gen_Relation().

11.43.4 Member Data Documentation

11.43.4.1 int parma_polyhedra_library::Poly_Gen_Relation::mask_value [private]

Holds the value of the possible relations.

Definition at line 37 of file Poly_Gen_Relation.java.

Referenced by implies(), and Poly_Gen_Relation().

11.43.4.2 final int parma_polyhedra_library::Poly_Gen_Relation::NOTHING = 0 [static]

Definition at line 33 of file Poly_Gen_Relation.java.

11.43.4.3 final int parma_polyhedra_library::Poly_Gen_Relation::SUBSUMES = 1 [static]

Definition at line 34 of file Poly_Gen_Relation.java.

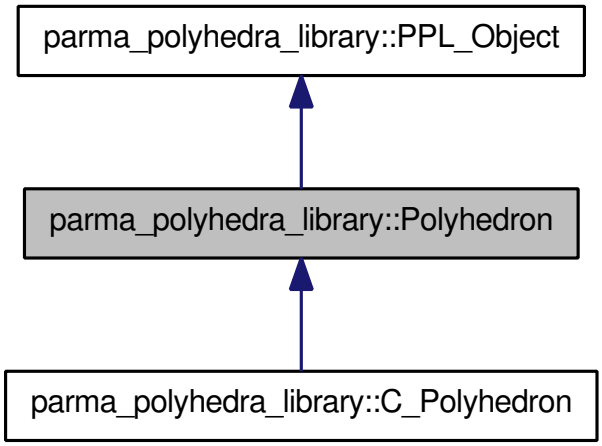
The documentation for this class was generated from the following file:

- Poly_Gen_Relation.java

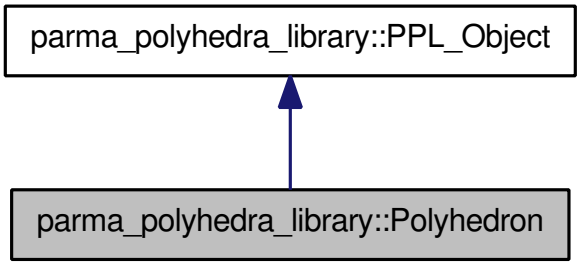
11.44 parma_polyhedra_library::Polyhedron Class Reference

The Java base class for (C and NNC) convex polyhedra.

Inheritance diagram for parma_polyhedra_library::Polyhedron:



Collaboration diagram for parma_polyhedra_library::Polyhedron:



Public Member Functions

Member Functions that Do Not Modify the Polyhedron

- native long space_dimension ()
  Returns the dimension of the vector space enclosing `this`.
- native long affine_dimension ()

Returns 0, *if* `this` *is empty; otherwise, returns the affine dimension of* `this`.

- native Constraint_System constraints ()
  Returns the system of constraints.
- native Congruence_System congruences ()
  Returns a system of (equality) congruences satisfied by `this`.
- native Constraint_System minimized_constraints ()
  Returns the system of constraints, with no redundant constraint.
- native Congruence_System minimized_congruences ()
  Returns a system of (equality) congruences satisfied by `this`, *with no redundant congruences and having the same affine dimension as* `this`.
- native boolean is_empty ()
  Returns `true` *if and only if* `this` *is an empty polyhedron.*
- native boolean is_universe ()
  Returns `true` *if and only if* `this` *is a universe polyhedron.*
- native boolean is_bounded ()
  Returns `true` *if and only if* `this` *is a bounded polyhedron.*
- native boolean is_discrete ()
  Returns `true` *if and only if* `this` *is discrete.*
- native boolean is_topologically_closed ()
  Returns `true` *if and only if* `this` *is a topologically closed subset of the vector space.*
- native boolean contains_integer_point ()
  Returns `true` *if and only if* `this` *contains at least one integer point.*
- native boolean constrains (Variable var)
  Returns `true` *if and only if* `var` *is constrained in* `this`.
- native boolean bounds_from_above (Linear_Expression expr)
  Returns `true` *if and only if* `expr` *is bounded from above in* `this`.
- native boolean bounds_from_below (Linear_Expression expr)
  Returns `true` *if and only if* `expr` *is bounded from below in* `this`.
- native boolean maximize (Linear_Expression expr, Coefficient sup_n, Coefficient sup_d, By_Reference< Boolean > maximum)
  Returns `true` *if and only if* `this` *is not empty and* `expr` *is bounded from above in* `this`, *in which case the supremum value is computed.*
- native boolean minimize (Linear_Expression expr, Coefficient inf_n, Coefficient inf_d, By_Reference< Boolean > minimum)
  Returns `true` *if and only if* `this` *is not empty and* `expr` *is bounded from below in* `this`, *in which case the infimum value is computed.*
- native boolean maximize (Linear_Expression expr, Coefficient sup_n, Coefficient sup_d, By_Reference< Boolean > maximum, Generator g)
  Returns `true` *if and only if* `this` *is not empty and* `expr` *is bounded from above in* `this`, *in which case the supremum value and a point where* `expr` *reaches it are computed.*

- native boolean minimize (Linear_Expression expr, Coefficient inf_n, Coefficient inf_d, By_-Reference< Boolean > minimum, Generator g)

  Returns `true` *if and only if* `this` *is not empty and* `expr` *is bounded from below in* `this`*, in which case the infimum value and a point where* `expr` *reaches it are computed.*

- native Poly_Con_Relation relation_with (Constraint c)

  Returns the relations holding between the polyhedron `this` *and the constraint* `c`*.*

- native Poly_Gen_Relation relation_with (Generator c)

  Returns the relations holding between the polyhedron `this` *and the generator* `g`*.*

- native Poly_Con_Relation relation_with (Congruence c)

  Returns the relations holding between the polyhedron `this` *and the congruence* `c`*.*

- native boolean contains (Polyhedron y)

  Returns `true` *if and only if* `this` *contains* `y`*.*

- native boolean strictly_contains (Polyhedron y)

  Returns `true` *if and only if* `this` *strictly contains* `y`*.*

- native boolean is_disjoint_from (Polyhedron y)

  Returns `true` *if and only if* `this` *and* `y` *are disjoint.*

- native boolean equals (Polyhedron y)

  Returns `true` *if and only if* `this` *and* `y` *are equal.*

- boolean equals (Object y)

  Returns `true` *if and only if* `this` *and* `y` *are equal.*

- native int hashCode ()

  Returns a hash code for `this`*.*

- native long external_memory_in_bytes ()

  Returns the size in bytes of the memory managed by `this`*.*

- native long total_memory_in_bytes ()

  Returns the total size in bytes of the memory occupied by `this`*.*

- native String toString ()

  Returns a string representing `this`*.*

- native String ascii_dump ()

  Returns a string containing a low-level representation of `this`*.*

- native boolean OK ()

  Checks if all the invariants are satisfied.

Space Dimension Preserving Member Functions that May Modify the Polyhedron

- native void add_constraint (Constraint c)

  Adds a copy of constraint `c` *to the system of constraints of* `this` *(without minimizing the result).*

- native void add_congruence (Congruence cg)

Adds a copy of congruence `cg` to `this`, if `cg` can be exactly represented by a polyhedron.

- native void add_constraints (Constraint_System cs)
  Adds a copy of the constraints in `cs` to the system of constraints of `this` (without minimizing the result).
- native void add_congruences (Congruence_System cgs)
  Adds a copy of the congruences in `cgs` to `this`, if all the congruences can be exactly represented by a polyhedron.
- native void refine_with_constraint (Constraint c)
  Uses a copy of constraint `c` to refine `this`.
- native void refine_with_congruence (Congruence cg)
  Uses a copy of congruence `cg` to refine `this`.
- native void refine_with_constraints (Constraint_System cs)
  Uses a copy of the constraints in `cs` to refine `this`.
- native void refine_with_congruences (Congruence_System cgs)
  Uses a copy of the congruences in `cgs` to refine `this`.
- native void intersection_assign (Polyhedron y)
  Assigns to `this` the intersection of `this` and `y`. The result is not guaranteed to be minimized.
- native void upper_bound_assign (Polyhedron y)
  Assigns to `this` the upper bound of `this` and `y`.
- native void difference_assign (Polyhedron y)
  Assigns to `this` the poly-difference of `this` and `y`. The result is not guaranteed to be minimized.
- native void time_elapse_assign (Polyhedron y)
  Assigns to `this` the result of computing the time-elapse between `this` and `y`.
- native void topological_closure_assign ()
  Assigns to `this` its topological closure.
- native boolean simplify_using_context_assign (Polyhedron y)
  Assigns to `this` a meet-preserving simplification of `this` with respect to `y`. If `false` is returned, then the intersection is empty.
- native void affine_image (Variable var, Linear_Expression expr, Coefficient denominator)
  Assigns to `this` the affine image of `this` under the function mapping variable `var` to the affine expression specified by `expr` and `denominator`.
- native void affine_preimage (Variable var, Linear_Expression expr, Coefficient denominator)
  Assigns to `this` the affine preimage of `this` under the function mapping variable `var` to the affine expression specified by `expr` and `denominator`.
- native void bounded_affine_image (Variable var, Linear_Expression lb_expr, Linear_Expression ub_expr, Coefficient denominator)
  Assigns to `this` the image of `this` with respect to the bounded affine relation $\frac{\text{lb\_expr}}{\text{denominator}} \leq \text{var}' \leq \frac{\text{ub\_expr}}{\text{denominator}}$.
- native void bounded_affine_preimage (Variable var, Linear_Expression lb_expr, Linear_Expression ub_expr, Coefficient denominator)

Assigns to `this` *the preimage of* `this` *with respect to the bounded affine relation* $\frac{\text{lb\_expr}}{\text{denominator}} \leq \text{var}' \leq \frac{\text{ub\_expr}}{\text{denominator}}$.

- native void generalized_affine_image (Variable var, Relation_Symbol relsym, Linear_Expression expr, Coefficient denominator)

  Assigns to `this` *the image of* `this` *with respect to the generalized affine relation* $\text{var}' \bowtie \frac{\text{expr}}{\text{denominator}}$, *where* $\bowtie$ *is the relation symbol encoded by* `relsym`.

- native void generalized_affine_preimage (Variable var, Relation_Symbol relsym, Linear_Expression expr, Coefficient denominator)

  Assigns to `this` *the preimage of* `this` *with respect to the generalized affine relation* $\text{var}' \bowtie \frac{\text{expr}}{\text{denominator}}$, *where* $\bowtie$ *is the relation symbol encoded by* `relsym`.

- native void generalized_affine_image (Linear_Expression lhs, Relation_Symbol relsym, Linear_Expression rhs)

  Assigns to `this` *the image of* `this` *with respect to the generalized affine relation* $\text{lhs}' \bowtie \text{rhs}$, *where* $\bowtie$ *is the relation symbol encoded by* `relsym`.

- native void generalized_affine_preimage (Linear_Expression lhs, Relation_Symbol relsym, Linear_Expression rhs)

  Assigns to `this` *the preimage of* `this` *with respect to the generalized affine relation* $\text{lhs}' \bowtie \text{rhs}$, *where* $\bowtie$ *is the relation symbol encoded by* `relsym`.

- native void unconstrain_space_dimension (Variable var)

  Computes the cylindrification of `this` *with respect to space dimension* `var`, *assigning the result to* `this`.

- native void unconstrain_space_dimensions (Variables_Set vars)

  Computes the cylindrification of `this` *with respect to the set of space dimensions* `vars`, *assigning the result to* `this`.

- native void widening_assign (Polyhedron y, By_Reference< Integer > tp)

  Assigns to `this` *the result of computing the H79-widening between* `this` *and* `y`.

Member Functions that May Modify the Dimension of the Vector Space

- native void swap (Polyhedron y)

  Swaps `this` *with polyhedron* `y`. *(*`this` *and* `y` *can be dimension-incompatible.).*

- native void add_space_dimensions_and_embed (long m)

  Adds `m` *new space dimensions and embeds the old polyhedron in the new vector space.*

- native void add_space_dimensions_and_project (long m)

  Adds `m` *new space dimensions to the polyhedron and does not embed it in the new vector space.*

- native void concatenate_assign (Polyhedron y)

  Assigns to `this` *the concatenation of* `this` *and* `y`, *taken in this order.*

- native void remove_space_dimensions (Variables_Set vars)

  Removes all the specified dimensions from the vector space.

- native void remove_higher_space_dimensions (long new_dimension)

  Removes the higher dimensions of the vector space so that the resulting space will have dimension `new_dimension`.

- native void expand_space_dimension (Variable var, long m)
  Creates `m` copies of the space dimension corresponding to `var`.

- native void fold_space_dimensions (Variables_Set vars, Variable dest)
  Folds the space dimensions in `vars` into `dest`.

- native void map_space_dimensions (Partial_Function pfunc)
  Remaps the dimensions of the vector space according to a partial function.

Ad Hoc Functions for (C or NNC) Polyhedra

The functions listed here below, being specific of the polyhedron domains, do not have a correspondence in other semantic geometric descriptions.

- native Generator_System generators ()
  Returns the system of generators.

- native Generator_System minimized_generators ()
  Returns the system of generators, with no redundant generator.

- native void add_generator (Generator g)
  Adds a copy of generator `g` to the system of generators of `this` (without minimizing the result).

- native void add_generators (Generator_System gs)
  Adds a copy of the generators in `gs` to the system of generators of `this` (without minimizing the result).

- native void poly_hull_assign (Polyhedron y)
  Same as upper_bound_assign.

- native void poly_difference_assign (Polyhedron y)
  Same as difference_assign.

- native void BHRZ03_widening_assign (Polyhedron y, By_Reference< Integer > tp)
  Assigns to `this` the result of computing the BHRZ03-widening between `this` and `y`.

- native void H79_widening_assign (Polyhedron y, By_Reference< Integer > tp)
  Assigns to `this` the result of computing the H79-widening between `this` and `y`.

- native void limited_BHRZ03_extrapolation_assign (Polyhedron y, Constraint_System cs, By_Reference< Integer > tp)
  Improves the result of the BHRZ03-widening computation by also enforcing those constraints in `cs` that are satisfied by all the points of `this`.

- native void limited_H79_extrapolation_assign (Polyhedron y, Constraint_System cs, By_Reference< Integer > tp)
  Improves the result of the H79-widening computation by also enforcing those constraints in `cs` that are satisfied by all the points of `this`.

- native void bounded_BHRZ03_extrapolation_assign (Polyhedron y, Constraint_System cs, By_Reference< Integer > tp)
  Improves the result of the BHRZ03-widening computation by also enforcing those constraints in `cs` that are satisfied by all the points of `this`, plus all the constraints of the form $\pm x \leq r$ and $\pm x < r$, with $r \in \mathbb{Q}$, that are satisfied by all the points of `this`.

- native void bounded_H79_extrapolation_assign (Polyhedron y, Constraint_System cs, By_-Reference< Integer > tp)

  Improves the result of the H79-widening computation by also enforcing those constraints in `cs` *that are satisfied by all the points of* `this`*, plus all the constraints of the form* $\pm x \leq r$ *and* $\pm x < r$*, with* $r \in \mathbb{Q}$*, that are satisfied by all the points of* `this`*.*

11.44.1 Detailed Description

The Java base class for (C and NNC) convex polyhedra. The base class Polyhedron provides declarations for most of the methods common to classes C_Polyhedron and NNC_Polyhedron. Note that the user should always use the derived classes. Moreover, C and NNC polyhedra can not be freely interchanged: as specified in the main manual, most library functions require their arguments to be topologically compatible.

Definition at line 38 of file Fake_Class_for_Doxygen.java.

11.44.2 Member Function Documentation

11.44.2.1 native void parma_polyhedra_library::Polyhedron::add_congruence (Congruence *cg*)

Adds a copy of congruence `cg` to `this`, if `cg` can be exactly represented by a polyhedron.

Exceptions

Invalid_Argument_Exception Thrown if `this` and congruence `cg` are dimension-incompatible, of if `cg` is a proper congruence which is neither a tautology, nor a contradiction.

11.44.2.2 native void parma_polyhedra_library::Polyhedron::add_congruences (Congruence_System *cgs*)

Adds a copy of the congruences in `cgs` to `this`, if all the congruences can be exactly represented by a polyhedron.

Parameters

cgs The congruences to be added.

Exceptions

Invalid_Argument_Exception Thrown if `this` and `cgs` are dimension-incompatible, of if there exists in `cgs` a proper congruence which is neither a tautology, nor a contradiction.

11.44.2.3 native void parma_polyhedra_library::Polyhedron::add_constraint (Constraint *c*)

Adds a copy of constraint `c` to the system of constraints of `this` (without minimizing the result).

Parameters

c The constraint that will be added to the system of constraints of `this`.

Exceptions

Invalid_Argument_Exception Thrown if `this` and constraint `c` are topology-incompatible or dimension-incompatible.

11.44.2.4 native void parma_polyhedra_library::Polyhedron::add_constraints (Constraint_System *cs*)

Adds a copy of the constraints in `cs` to the system of constraints of `this` (without minimizing the result).

Parameters

cs Contains the constraints that will be added to the system of constraints of `this`.

Exceptions

Invalid_Argument_Exception Thrown if `this` and `cs` are topology-incompatible or dimension-incompatible.

11.44.2.5 native void parma_polyhedra_library::Polyhedron::add_generator (Generator *g*)

Adds a copy of generator `g` to the system of generators of `this` (without minimizing the result).

Exceptions

Invalid_Argument_Exception Thrown if `this` and generator `g` are topology-incompatible or dimension-incompatible, or if `this` is an empty polyhedron and `g` is not a point.

11.44.2.6 native void parma_polyhedra_library::Polyhedron::add_generators (Generator_System *gs*)

Adds a copy of the generators in `gs` to the system of generators of `this` (without minimizing the result).

Parameters

gs Contains the generators that will be added to the system of generators of `this`.

Exceptions

Invalid_Argument_Exception Thrown if `this` and `gs` are topology-incompatible or dimension-incompatible, or if `this` is empty and the system of generators `gs` is not empty, but has no points.

11.44.2.7 native void parma_polyhedra_library::Polyhedron::add_space_dimensions_and_embed (long *m*)

Adds `m` new space dimensions and embeds the old polyhedron in the new vector space.

Parameters

m The number of dimensions to add.

Exceptions

Length_Error_Exception Thrown if adding `m` new space dimensions would cause the vector space to exceed dimension `max_space_dimension()`.

11.44.2.8 native void parma_polyhedra_library::Polyhedron::add_space_dimensions_and_project (long *m*)

Adds `m` new space dimensions to the polyhedron and does not embed it in the new vector space.

Parameters

m The number of space dimensions to add.

Exceptions

Length_Error_Exception Thrown if adding `m` new space dimensions would cause the vector space to exceed dimension `max_space_dimension()`.

11.44.2.9 native long parma_polyhedra_library::Polyhedron::affine_dimension ()

Returns 0, if `this` is empty; otherwise, returns the *affine dimension* of `this`.

11.44.2.10 native void parma_polyhedra_library::Polyhedron::affine_image (Variable *var*, Linear_Expression *expr*, Coefficient *denominator*)

Assigns to `this` the *affine image* of `this` under the function mapping variable `var` to the affine expression specified by `expr` and `denominator`.

Parameters

var The variable to which the affine expression is assigned;

expr The numerator of the affine expression;

denominator The denominator of the affine expression (optional argument with default value 1).

Exceptions

Invalid_Argument_Exception Thrown if `denominator` is zero or if `expr` and `this` are dimension-incompatible or if `var` is not a space dimension of `this`.

11.44.2.11 native void parma_polyhedra_library::Polyhedron::affine_preimage (Variable *var*, Linear_Expression *expr*, Coefficient *denominator*)

Assigns to `this` the *affine preimage* of `this` under the function mapping variable `var` to the affine expression specified by `expr` and `denominator`.

Parameters

var The variable to which the affine expression is substituted;

expr The numerator of the affine expression;

denominator The denominator of the affine expression (optional argument with default value 1).

Exceptions

Invalid_Argument_Exception Thrown if `denominator` is zero or if `expr` and `this` are dimension-incompatible or if `var` is not a space dimension of `this`.

11.44.2.12 native String parma_polyhedra_library::Polyhedron::ascii_dump ()

Returns a string containing a low-level representation of `this`.

Useful for debugging purposes.

11.44.2.13 native void parma_polyhedra_library::Polyhedron::BHRZ03_widening_assign (Polyhedron *y*, By_Reference< Integer > *tp*)

Assigns to `this` the result of computing the *BHRZ03-widening* between `this` and `y`.

Parameters

y A polyhedron that *must* be contained in `this`;

tp A reference to an unsigned variable storing the number of available tokens (to be used when applying the *widening with tokens* delay technique).

Exceptions

Invalid_Argument_Exception Thrown if `this` and `y` are topology-incompatible or dimension-incompatible.

11.44.2.14 native void parma_polyhedra_library::Polyhedron::bounded_affine_image (Variable *var*, Linear_Expression *lb_expr*, Linear_Expression *ub_expr*, Coefficient *denominator*)

Assigns to `this` the image of `this` with respect to the *bounded affine relation* $\frac{\text{lb\_expr}}{\text{denominator}} \leq \text{var}' \leq \frac{\text{ub\_expr}}{\text{denominator}}$.

Parameters

var The variable updated by the affine relation;

lb_expr The numerator of the lower bounding affine expression;

ub_expr The numerator of the upper bounding affine expression;

denominator The (common) denominator for the lower and upper bounding affine expressions (optional argument with default value 1).

Exceptions

Invalid_Argument_Exception Thrown if `denominator` is zero or if `lb_expr` (resp., `ub_expr`) and `this` are dimension-incompatible or if `var` is not a space dimension of `this`.

11.44.2.15 native void parma_polyhedra_library::Polyhedron::bounded_affine_preimage (Variable *var*, Linear_Expression *lb_expr*, Linear_Expression *ub_expr*, Coefficient *denominator*)

Assigns to `this` the preimage of `this` with respect to the *bounded affine relation* $\frac{\text{lb\_expr}}{\text{denominator}} \leq \text{var}' \leq \frac{\text{ub\_expr}}{\text{denominator}}$.

Parameters

var The variable updated by the affine relation;

lb_expr The numerator of the lower bounding affine expression;

ub_expr The numerator of the upper bounding affine expression;

denominator The (common) denominator for the lower and upper bounding affine expressions (optional argument with default value 1).

Exceptions

Invalid_Argument_Exception Thrown if `denominator` is zero or if `lb_expr` (resp., `ub_expr`) and `this` are dimension-incompatible or if `var` is not a space dimension of `this`.

11.44.2.16 native void parma_polyhedra_library::Polyhedron::bounded_BHRZ03_extrapolation_assign (Polyhedron *y*, Constraint_System *cs*, By_Reference< Integer > *tp*)

Improves the result of the *BHRZ03-widening* computation by also enforcing those constraints in `cs` that are satisfied by all the points of `this`, plus all the constraints of the form $\pm x \leq r$ and $\pm x < r$, with $r \in \mathbb{Q}$, that are satisfied by all the points of `this`.

Parameters

y A polyhedron that *must* be contained in `this`;

cs The system of constraints used to improve the widened polyhedron;

tp A reference to an unsigned variable storing the number of available tokens (to be used when applying the *widening with tokens* delay technique).

Exceptions

Invalid_Argument_Exception Thrown if `this`, `y` and `cs` are topology-incompatible or dimension-incompatible.

11.44.2.17 native void parma_polyhedra_library::Polyhedron::bounded_H79_extrapolation_assign (Polyhedron *y*, Constraint_System *cs*, By_Reference< Integer > *tp*)

Improves the result of the *H79-widening* computation by also enforcing those constraints in `cs` that are satisfied by all the points of `this`, plus all the constraints of the form $\pm x \leq r$ and $\pm x < r$, with $r \in \mathbb{Q}$, that are satisfied by all the points of `this`.

Parameters

y A polyhedron that *must* be contained in `this`;

cs The system of constraints used to improve the widened polyhedron;

tp A reference to an unsigned variable storing the number of available tokens (to be used when applying the *widening with tokens* delay technique).

Exceptions

Invalid_Argument_Exception Thrown if `this`, `y` and `cs` are topology-incompatible or dimension-incompatible.

11.44.2.18 native boolean parma_polyhedra_library::Polyhedron::bounds_from_above (Linear_Expression *expr*)

Returns `true` if and only if `expr` is bounded from above in `this`.

Exceptions

Invalid_Argument_Exception Thrown if `expr` and `this` are dimension-incompatible.

11.44.2.19 native boolean parma_polyhedra_library::Polyhedron::bounds_from_below (Linear_Expression *expr*)

Returns `true` if and only if `expr` is bounded from below in `this`.

Exceptions

Invalid_Argument_Exception Thrown if `expr` and `this` are dimension-incompatible.

11.44.2.20 native void parma_polyhedra_library::Polyhedron::concatenate_assign (Polyhedron *y*)

Assigns to `this` the *concatenation* of `this` and `y`, taken in this order.

Exceptions

Invalid_Argument_Exception Thrown if `this` and `y` are topology-incompatible.

Length_Error_Exception Thrown if the concatenation would cause the vector space to exceed dimension `max_space_dimension()`.

11.44.2.21 native Congruence_System parma_polyhedra_library::Polyhedron::congruences ()

Returns a system of (equality) congruences satisfied by `this`.

11.44.2.22 native boolean parma_polyhedra_library::Polyhedron::constrains (Variable *var*)

Returns `true` if and only if `var` is constrained in `this`.

Exceptions

Invalid_Argument_Exception Thrown if `var` is not a space dimension of `this`.

11.44.2.23 native Constraint_System parma_polyhedra_library::Polyhedron::constraints ()

Returns the system of constraints.

11.44.2.24 native boolean parma_polyhedra_library::Polyhedron::contains (Polyhedron *y*)

Returns `true` if and only if `this` contains `y`.

Exceptions

Invalid_Argument_Exception Thrown if `this` and `y` are topology-incompatible or dimension-incompatible.

11.44.2.25 native boolean parma_polyhedra_library::Polyhedron::contains_integer_point ()

Returns `true` if and only if `this` contains at least one integer point.

11.44.2.26 native void parma_polyhedra_library::Polyhedron::difference_assign (Polyhedron *y*)

Assigns to `this` the *poly-difference* of `this` and `y`. The result is not guaranteed to be minimized.

Exceptions

Invalid_Argument_Exception Thrown if `this` and `y` are topology-incompatible or dimension-incompatible.

11.44.2.27 boolean parma_polyhedra_library::Polyhedron::equals (Object *y*)

Returns `true` if and only if `this` and `y` are equal.

11.44.2.28 native boolean parma_polyhedra_library::Polyhedron::equals (Polyhedron *y*)

Returns `true` if and only if `this` and `y` are equal.

11.44.2.29 native void parma_polyhedra_library::Polyhedron::expand_space_dimension (Variable *var*, long *m*)

Creates `m` copies of the space dimension corresponding to `var`.

Parameters

var The variable corresponding to the space dimension to be replicated;

m The number of replicas to be created.

Exceptions

Invalid_Argument_Exception Thrown if `var` does not correspond to a dimension of the vector space.

Length_Error_Exception Thrown if adding `m` new space dimensions would cause the vector space to exceed dimension `max_space_dimension()`.

11.44.2.30 native long parma_polyhedra_library::Polyhedron::external_memory_in_bytes ()

Returns the size in bytes of the memory managed by `this`.

11.44.2.31 native void parma_polyhedra_library::Polyhedron::fold_space_dimensions (Variables_Set *vars*, Variable *dest*)

Folds the space dimensions in `vars` into `dest`.

Parameters

vars The set of Variable objects corresponding to the space dimensions to be folded;

dest The variable corresponding to the space dimension that is the destination of the folding operation.

Exceptions

Invalid_Argument_Exception Thrown if `this` is dimension-incompatible with `dest` or with one of the Variable objects contained in `vars`. Also thrown if `dest` is contained in `vars`.

11.44.2.32 native void parma_polyhedra_library::Polyhedron::generalized_affine_image (Linear_Expression *lhs*, Relation_Symbol *relsym*, Linear_Expression *rhs*)

Assigns to `this` the image of `this` with respect to the *generalized affine relation* $\mathrm{lhs}' \bowtie \mathrm{rhs}$, where $\bowtie$ is the relation symbol encoded by `relsym`.

Parameters

lhs The left hand side affine expression;

relsym The relation symbol;

rhs The right hand side affine expression.

Exceptions

Invalid_Argument_Exception Thrown if `this` is dimension-incompatible with `lhs` or `rhs` or if `this` is a C_Polyhedron and `relsym` is a strict relation symbol.

11.44.2.33 native void parma_polyhedra_library::Polyhedron::generalized_affine_image (Variable *var*, Relation_Symbol *relsym*, Linear_Expression *expr*, Coefficient *denominator*)

Assigns to `this` the image of `this` with respect to the *generalized affine relation* $\mathrm{var}' \bowtie \frac{\mathrm{expr}}{\mathrm{denominator}}$, where $\bowtie$ is the relation symbol encoded by `relsym`.

Parameters

var The left hand side variable of the generalized affine relation;

relsym The relation symbol;

expr The numerator of the right hand side affine expression;

denominator The denominator of the right hand side affine expression (optional argument with default value 1).

Exceptions

Invalid_Argument_Exception Thrown if `denominator` is zero or if `expr` and `this` are dimension-incompatible or if `var` is not a space dimension of `this` or if `this` is a C_Polyhedron and `relsym` is a strict relation symbol.

11.44.2.34 native void parma_polyhedra_library::Polyhedron::generalized_affine_preimage (Linear_Expression *lhs*, Relation_Symbol *relsym*, Linear_Expression *rhs*)

Assigns to `this` the preimage of `this` with respect to the *generalized affine relation* $\mathrm{lhs}' \bowtie \mathrm{rhs}$, where $\bowtie$ is the relation symbol encoded by `relsym`.

Parameters

lhs The left hand side affine expression;

relsym The relation symbol;

rhs The right hand side affine expression.

Exceptions

Invalid_Argument_Exception Thrown if `this` is dimension-incompatible with `lhs` or `rhs` or if `this` is a C_Polyhedron and `relsym` is a strict relation symbol.

11.44.2.35 native void parma_polyhedra_library::Polyhedron::generalized_affine_preimage (Variable *var*, Relation_Symbol *relsym*, Linear_Expression *expr*, Coefficient *denominator*)

Assigns to `this` the preimage of `this` with respect to the *generalized affine relation* $\mathrm{var}' \bowtie \frac{\mathrm{expr}}{\mathrm{denominator}}$, where $\bowtie$ is the relation symbol encoded by `relsym`.

Parameters

var The left hand side variable of the generalized affine relation;

relsym The relation symbol;

expr The numerator of the right hand side affine expression;

denominator The denominator of the right hand side affine expression (optional argument with default value 1).

Exceptions

Invalid_Argument_Exception Thrown if `denominator` is zero or if `expr` and `this` are dimension-incompatible or if `var` is not a space dimension of `this` or if `this` is a C_Polyhedron and `relsym` is a strict relation symbol.

11.44.2.36 native Generator_System parma_polyhedra_library::Polyhedron::generators ()

Returns the system of generators.

11.44.2.37 native void parma_polyhedra_library::Polyhedron::H79_widening_assign (Polyhedron *y*, By_Reference< Integer > *tp*)

Assigns to `this` the result of computing the *H79-widening* between `this` and `y`.

Parameters

y A polyhedron that *must* be contained in `this`;

tp A reference to an unsigned variable storing the number of available tokens (to be used when applying the *widening with tokens* delay technique).

Exceptions

Invalid_Argument_Exception Thrown if `this` and `y` are topology-incompatible or dimension-incompatible.

11.44.2.38 native int parma_polyhedra_library::Polyhedron::hashCode ()

Returns a hash code for `this`.

If `x` and `y` are such that `x == y`, then `x.hash_code() == y.hash_code()`.

11.44.2.39 native void parma_polyhedra_library::Polyhedron::intersection_assign (Polyhedron *y*)

Assigns to `this` the intersection of `this` and `y`. The result is not guaranteed to be minimized.

Exceptions

Invalid_Argument_Exception Thrown if `this` and `y` are topology-incompatible or dimension-incompatible.

11.44.2.40 native boolean parma_polyhedra_library::Polyhedron::is_bounded ()

Returns `true` if and only if `this` is a bounded polyhedron.

11.44.2.41 native boolean parma_polyhedra_library::Polyhedron::is_discrete ()

Returns `true` if and only if `this` is discrete.

11.44.2.42 native boolean parma_polyhedra_library::Polyhedron::is_disjoint_from (Polyhedron *y*)

Returns `true` if and only if `this` and `y` are disjoint.

Exceptions

Invalid_Argument_Exception Thrown if `x` and `y` are topology-incompatible or dimension-incompatible.

11.44.2.43 native boolean parma_polyhedra_library::Polyhedron::is_empty ()

Returns `true` if and only if `this` is an empty polyhedron.

11.44.2.44 native boolean parma_polyhedra_library::Polyhedron::is_topologically_closed ()

Returns `true` if and only if `this` is a topologically closed subset of the vector space.

11.44.2.45 native boolean parma_polyhedra_library::Polyhedron::is_universe ()

Returns `true` if and only if `this` is a universe polyhedron.

11.44.2.46 native void parma_polyhedra_library::Polyhedron::limited_BHRZ03_extrapolation_assign (Polyhedron *y*, Constraint_System *cs*, By_Reference< Integer > *tp*)

Improves the result of the *BHRZ03-widening* computation by also enforcing those constraints in `cs` that are satisfied by all the points of `this`.

Parameters

y A polyhedron that *must* be contained in `this`;

cs The system of constraints used to improve the widened polyhedron;

tp A reference to an unsigned variable storing the number of available tokens (to be used when applying the *widening with tokens* delay technique).

Exceptions

Invalid_Argument_Exception Thrown if `this`, `y` and `cs` are topology-incompatible or dimension-incompatible.

11.44.2.47 native void parma_polyhedra_library::Polyhedron::limited_H79_extrapolation_assign (Polyhedron *y*, Constraint_System *cs*, By_Reference< Integer > *tp*)

Improves the result of the *H79-widening* computation by also enforcing those constraints in `cs` that are satisfied by all the points of `this`.

Parameters

y A polyhedron that *must* be contained in `this`;

cs The system of constraints used to improve the widened polyhedron;

tp A reference to an unsigned variable storing the number of available tokens (to be used when applying the *widening with tokens* delay technique).

Exceptions

Invalid_Argument_Exception Thrown if `this`, `y` and `cs` are topology-incompatible or dimension-incompatible.

11.44.2.48 native void parma_polyhedra_library::Polyhedron::map_space_dimensions (Partial_Function *pfunc*)

Remaps the dimensions of the vector space according to a *partial function*.

Parameters

pfunc The partial function specifying the destiny of each space dimension.

11.44.2.49 native boolean parma_polyhedra_library::Polyhedron::maximize (Linear_Expression *expr*, Coefficient *sup_n*, Coefficient *sup_d*, By_Reference< Boolean > *maximum*, Generator *g*)

Returns `true` if and only if `this` is not empty and `expr` is bounded from above in `this`, in which case the supremum value and a point where `expr` reaches it are computed.

Parameters

expr The linear expression to be maximized subject to `this`;

sup_n The numerator of the supremum value;

sup_d The denominator of the supremum value;

maximum `true` if and only if the supremum is also the maximum value;

g When maximization succeeds, will be assigned the point or closure point where `expr` reaches its supremum value.

Exceptions

Invalid_Argument_Exception Thrown if `expr` and `this` are dimension-incompatible.

If `this` is empty or `expr` is not bounded from above, `false` is returned and `sup_n`, `sup_d`, `maximum` and `g` are left untouched.

11.44.2.50 native boolean parma_polyhedra_library::Polyhedron::maximize (Linear_Expression *expr*, Coefficient *sup_n*, Coefficient *sup_d*, By_Reference< Boolean > *maximum*)

Returns `true` if and only if `this` is not empty and `expr` is bounded from above in `this`, in which case the supremum value is computed.

Parameters

expr The linear expression to be maximized subject to `this`;

sup_n The numerator of the supremum value;

sup_d The denominator of the supremum value;

maximum `true` if and only if the supremum is also the maximum value.

Exceptions

Invalid_Argument_Exception Thrown if `expr` and `this` are dimension-incompatible.

If `this` is empty or `expr` is not bounded from above, `false` is returned and `sup_n`, `sup_d` and `maximum` are left untouched.

11.44.2.51 native boolean parma_polyhedra_library::Polyhedron::minimize (Linear_Expression *expr*, Coefficient *inf_n*, Coefficient *inf_d*, By_Reference< Boolean > *minimum*, Generator *g*)

Returns `true` if and only if `this` is not empty and `expr` is bounded from below in `this`, in which case the infimum value and a point where `expr` reaches it are computed.

Parameters

expr The linear expression to be minimized subject to `this`;

inf_n The numerator of the infimum value;

inf_d The denominator of the infimum value;

minimum `true` if and only if the infimum is also the minimum value;

g When minimization succeeds, will be assigned a point or closure point where `expr` reaches its infimum value.

Exceptions

Invalid_Argument_Exception Thrown if `expr` and `this` are dimension-incompatible.

If `this` is empty or `expr` is not bounded from below, `false` is returned and `inf_n`, `inf_d`, `minimum` and `g` are left untouched.

11.44.2.52 native boolean parma_polyhedra_library::Polyhedron::minimize (Linear_Expression *expr*, Coefficient *inf_n*, Coefficient *inf_d*, By_Reference< Boolean > *minimum*)

Returns `true` if and only if `this` is not empty and `expr` is bounded from below in `this`, in which case the infimum value is computed.

Parameters

expr The linear expression to be minimized subject to `this`;

inf_n The numerator of the infimum value;

inf_d The denominator of the infimum value;

minimum `true` if and only if the infimum is also the minimum value.

Exceptions

Invalid_Argument_Exception Thrown if `expr` and `this` are dimension-incompatible.

If `this` is empty or `expr` is not bounded from below, `false` is returned and `inf_n`, `inf_d` and `minimum` are left untouched.

11.44.2.53 native Congruence_System parma_polyhedra_library::Polyhedron::minimized_congruences ()

Returns a system of (equality) congruences satisfied by `this`, with no redundant congruences and having the same affine dimension as `this`.

11.44.2.54 native Constraint_System parma_polyhedra_library::Polyhedron::minimized_constraints ()

Returns the system of constraints, with no redundant constraint.

11.44.2.55 native Generator_System parma_polyhedra_library::Polyhedron::minimized_generators ()

Returns the system of generators, with no redundant generator.

11.44.2.56 native boolean parma_polyhedra_library::Polyhedron::OK ()

Checks if all the invariants are satisfied.

11.44.2.57 native void parma_polyhedra_library::Polyhedron::poly_difference_assign (Polyhedron *y*)

Same as difference_assign.

11.44.2.58 native void parma_polyhedra_library::Polyhedron::poly_hull_assign (Polyhedron *y*)

Same as upper_bound_assign.

11.44.2.59 native void parma_polyhedra_library::Polyhedron::refine_with_congruence (Congruence *cg*)

Uses a copy of congruence `cg` to refine `this`.

Exceptions

Invalid_Argument_Exception Thrown if `this` and congruence `cg` are dimension-incompatible.

11.44.2.60 native void parma_polyhedra_library::Polyhedron::refine_with_congruences (Congruence_System *cgs*)

Uses a copy of the congruences in `cgs` to refine `this`.

Parameters

cgs Contains the congruences used to refine the system of constraints of `this`.

Exceptions

Invalid_Argument_Exception Thrown if `this` and `cgs` are dimension-incompatible.

11.44.2.61 native void parma_polyhedra_library::Polyhedron::refine_with_constraint (Constraint *c*)

Uses a copy of constraint `c` to refine `this`.

Exceptions

Invalid_Argument_Exception Thrown if `this` and constraint `c` are dimension-incompatible.

11.44.2.62 native void parma_polyhedra_library::Polyhedron::refine_with_constraints (Constraint_System *cs*)

Uses a copy of the constraints in `cs` to refine `this`.

Parameters

cs Contains the constraints used to refine the system of constraints of `this`.

Exceptions

Invalid_Argument_Exception Thrown if `this` and `cs` are dimension-incompatible.

11.44.2.63 native Poly_Con_Relation parma_polyhedra_library::Polyhedron::relation_with (Congruence *c*)

Returns the relations holding between the polyhedron `this` and the congruence `c`.

Exceptions

Invalid_Argument_Exception Thrown if `this` and congruence `c` are dimension-incompatible.

11.44.2.64 native Poly_Gen_Relation parma_polyhedra_library::Polyhedron::relation_with (Generator *c*)

Returns the relations holding between the polyhedron `this` and the generator `g`.

Exceptions

Invalid_Argument_Exception Thrown if `this` and generator `g` are dimension-incompatible.

11.44.2.65 native Poly_Con_Relation parma_polyhedra_library::Polyhedron::relation_with (Constraint *c*)

Returns the relations holding between the polyhedron `this` and the constraint `c`.

Exceptions

Invalid_Argument_Exception Thrown if `this` and constraint `c` are dimension-incompatible.

11.44.2.66 native void parma_polyhedra_library::Polyhedron::remove_higher_space_dimensions (long *new_dimension*)

Removes the higher dimensions of the vector space so that the resulting space will have dimension `new_dimension`.

Exceptions

Invalid_Argument_Exception Thrown if `new_dimensions` is greater than the space dimension of `this`.

11.44.2.67 native void parma_polyhedra_library::Polyhedron::remove_space_dimensions (Variables_Set *vars*)

Removes all the specified dimensions from the vector space.

Parameters

vars The set of Variable objects corresponding to the space dimensions to be removed.

Exceptions

Invalid_Argument_Exception Thrown if `this` is dimension-incompatible with one of the Variable objects contained in `vars`.

11.44.2.68 native boolean parma_polyhedra_library::Polyhedron::simplify_using_context_assign (Polyhedron *y*)

Assigns to `this` a *meet-preserving simplification* of `this` with respect to `y`. If `false` is returned, then the intersection is empty.

Exceptions

Invalid_Argument_Exception Thrown if `this` and `y` are topology-incompatible or dimension-incompatible.

11.44.2.69 native long parma_polyhedra_library::Polyhedron::space_dimension ()

Returns the dimension of the vector space enclosing `this`.

11.44.2.70 native boolean parma_polyhedra_library::Polyhedron::strictly_contains (Polyhedron *y*)

Returns `true` if and only if `this` strictly contains `y`.

Exceptions

Invalid_Argument_Exception Thrown if `this` and `y` are topology-incompatible or dimension-incompatible.

11.44.2.71 native void parma_polyhedra_library::Polyhedron::swap (Polyhedron *y*)

Swaps `this` with polyhedron `y`. (`this` and `y` can be dimension-incompatible.).

Exceptions

Invalid_Argument_Exception Thrown if `x` and `y` are topology-incompatible.

11.44.2.72 native void parma_polyhedra_library::Polyhedron::time_elapse_assign (Polyhedron *y*)

Assigns to `this` the result of computing the *time-elapse* between `this` and `y`.

Exceptions

Invalid_Argument_Exception Thrown if `this` and `y` are topology-incompatible or dimension-incompatible.

11.44.2.73 native void parma_polyhedra_library::Polyhedron::topological_closure_assign ()

Assigns to `this` its topological closure.

11.44.2.74 native String parma_polyhedra_library::Polyhedron::toString ()

Returns a string representing `this`.

11.44.2.75 native long parma_polyhedra_library::Polyhedron::total_memory_in_bytes ()

Returns the total size in bytes of the memory occupied by `this`.

11.44.2.76 native void parma_polyhedra_library::Polyhedron::unconstrain_space_dimension (Variable *var*)

Computes the *cylindrification* of `this` with respect to space dimension `var`, assigning the result to `this`.

Parameters

var The space dimension that will be unconstrained.

Exceptions

Invalid_Argument_Exception Thrown if `var` is not a space dimension of `this`.

11.44.2.77 native void parma_polyhedra_library::Polyhedron::unconstrain_space_dimensions (Variables_Set *vars*)

Computes the *cylindrification* of `this` with respect to the set of space dimensions `vars`, assigning the result to `this`.

Parameters

vars The set of space dimension that will be unconstrained.

Exceptions

Invalid_Argument_Exception Thrown if `this` is dimension-incompatible with one of the Variable objects contained in `vars`.

11.44.2.78 native void parma_polyhedra_library::Polyhedron::upper_bound_assign (Polyhedron *y*)

Assigns to `this` the upper bound of `this` and `y`.

Exceptions

Invalid_Argument_Exception Thrown if `this` and `y` are topology-incompatible or dimension-incompatible.

11.44.2.79 native void parma_polyhedra_library::Polyhedron::widening_assign (Polyhedron *y*, By_Reference< Integer > *tp*)

Assigns to `this` the result of computing the *H79-widening* between `this` and `y`.

Parameters

y A polyhedron that *must* be contained in `this`;

tp A reference to an unsigned variable storing the number of available tokens (to be used when applying the *widening with tokens* delay technique).

Exceptions

Invalid_Argument_Exception Thrown if `this` and `y` are topology-incompatible or dimension-incompatible.

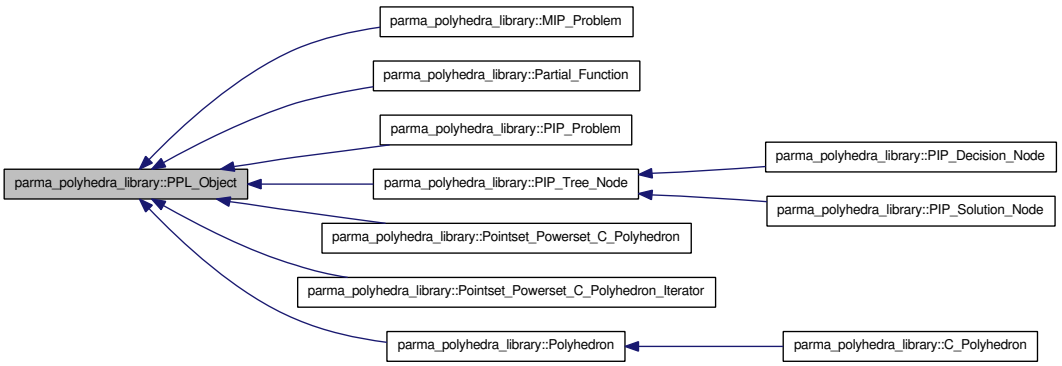
The documentation for this class was generated from the following file:

- Fake_Class_for_Doxygen.java

11.45 parma_polyhedra_library::PPL_Object Class Reference

Smart pointer to a PPL, C++ object.

Inheritance diagram for parma_polyhedra_library::PPL_Object:



Protected Member Functions

- PPL_Object ()

  Builds an object that points to 'null'.

Static Package Functions

- [static initializer]

Package Attributes

- long ptr

  Stores the value of the C++ pointer.

Static Private Member Functions

- static native void initIDs ()

11.45.1 Detailed Description

Smart pointer to a PPL, C++ object.

Definition at line 32 of file PPL_Object.java.

11.45.2 Constructor & Destructor Documentation

11.45.2.1 parma_polyhedra_library::PPL_Object::PPL_Object () [inline, protected]

Builds an object that points to 'null'.

Definition at line 38 of file PPL_Object.java.

References ptr.

11.45.3 Member Function Documentation

11.45.3.1 parma_polyhedra_library::PPL_Object::[static initializer] () [inline, static, package]

11.45.3.2 static native void parma_polyhedra_library::PPL_Object::initIDs () [static, private]

11.45.4 Member Data Documentation

11.45.4.1 long parma_polyhedra_library::PPL_Object::ptr [package]

Stores the value of the C++ pointer.

Definition at line 35 of file PPL_Object.java.

Referenced by PPL_Object().

The documentation for this class was generated from the following file:

- PPL_Object.java

11.46 parma_polyhedra_library::Timeout_Exception Class Reference

Exceptions caused by timeout expiring.

Public Member Functions

- Timeout_Exception (String s)

  Constructor.

11.46.1 Detailed Description

Exceptions caused by timeout expiring.

Definition at line 28 of file Timeout_Exception.java.

11.46.2 Constructor & Destructor Documentation

11.46.2.1 parma_polyhedra_library::Timeout_Exception::Timeout_Exception (String *s*) [inline]

Constructor.

Definition at line 30 of file Timeout_Exception.java.

The documentation for this class was generated from the following file:

- Timeout_Exception.java

11.47 Parma_Polyhedra_Library::Interfaces::Java::timeout_exception Class Reference

```
#include <ppl_java_common.defs.hh>
```

Public Member Functions

- void throw_me () const
- int priority () const

11.47.1 Detailed Description

Definition at line 122 of file ppl_java_common.defs.hh.

11.47.2 Member Function Documentation

11.47.2.1 int Parma_Polyhedra_Library::Interfaces::Java::timeout_exception::priority () const [inline]

Definition at line 128 of file ppl_java_common.defs.hh.

11.47.2.2 void Parma_Polyhedra_Library::Interfaces::Java::timeout_exception::throw_me () const [inline]

Definition at line 125 of file ppl_java_common.defs.hh.

The documentation for this class was generated from the following file:

- ppl_java_common.defs.hh

11.48 parma_polyhedra_library::Variable Class Reference

A dimension of the vector space.

Public Member Functions

- Variable (int i)

  Builds the variable corresponding to the Cartesian axis of index `i`.

- int id ()

  Returns the index of the Cartesian axis associated to `this`.

- int compareTo (Variable v)

  Returns a negative number if `this` *comes first than* `v`*, a zero if* `this` *equals* `v`*, a positive number if if* `this` *comes first than* `v`.

Static Package Functions

- [static initializer]

Static Private Member Functions

- static native void initIDs ()

Private Attributes

- int varid

  The index of the Cartesian axis.

11.48.1 Detailed Description

A dimension of the vector space. An object of the class Variable represents a dimension of the space, that is one of the Cartesian axes. Variables are used as basic blocks in order to build more complex linear expressions. Each variable is identified by a non-negative integer, representing the index of the corresponding Cartesian axis (the first axis has index 0).

Definition at line 38 of file Variable.java.

11.48.2 Constructor & Destructor Documentation

11.48.2.1 parma_polyhedra_library::Variable::Variable (int *i*) [inline]

Builds the variable corresponding to the Cartesian axis of index `i`.

Exceptions

RuntimeErrorException Thrown if `i` is has negative value.

Definition at line 47 of file Variable.java.

References varid.

11.48.3 Member Function Documentation

11.48.3.1 parma_polyhedra_library::Variable::[static initializer] () [inline, static, package]

11.48.3.2 int parma_polyhedra_library::Variable::compareTo (Variable *v*) [inline]

Returns a negative number if `this` comes first than `v`, a zero if `this` equals `v`, a positive number if if `this` comes first than `v`.

Definition at line 67 of file Variable.java.

References varid.

11.48.3.3 int parma_polyhedra_library::Variable::id () [inline]

Returns the index of the Cartesian axis associated to `this`.

Definition at line 58 of file Variable.java.

References varid.

Referenced by parma_polyhedra_library::Linear_Expression_Variable::Linear_Expression_Variable(), and parma_polyhedra_library::Linear_Expression_Variable::var_id().

11.48.3.4 static native void parma_polyhedra_library::Variable::initIDs () [static, private]

11.48.4 Member Data Documentation

11.48.4.1 int parma_polyhedra_library::Variable::varid [private]

The index of the Cartesian axis.

Definition at line 40 of file Variable.java.

Referenced by compareTo(), id(), and Variable().

The documentation for this class was generated from the following file:

- Variable.java

11.49 parma_polyhedra_library::Variables_Set Class Reference

A java.util.TreeSet of variables' indexes.

Public Member Functions

- Variables_Set ()

  Builds the empty set of variable indexes.

Static Package Functions

- [static initializer]

Static Private Member Functions

- static native void initIDs ()

11.49.1 Detailed Description

A java.util.TreeSet of variables' indexes.

Definition at line 27 of file Variables_Set.java.

11.49.2 Constructor & Destructor Documentation

11.49.2.1 parma_polyhedra_library::Variables_Set::Variables_Set () [inline]

Builds the empty set of variable indexes.

Definition at line 30 of file Variables_Set.java.

11.49.3 Member Function Documentation

11.49.3.1 parma_polyhedra_library::Variables_Set::[static initializer] () [inline, static, package]

11.49.3.2 static native void parma_polyhedra_library::Variables_Set::initIDs () [static, private]

The documentation for this class was generated from the following file:

- Variables_Set.java

12 File Documentation

12.1 Artificial_Parameter.java File Reference

Classes

- class parma_polyhedra_library::Artificial_Parameter

Namespaces

- namespace parma_polyhedra_library
  The PPL Java interface package.

12.2 Artificial_Parameter_Sequence.java File Reference

Classes

- class parma_polyhedra_library::Artificial_Parameter_Sequence

A sequence of artificial parameters.

Namespaces

- namespace parma_polyhedra_library

  The PPL Java interface package.

12.3 Bounded_Integer_Type_Overflow.java File Reference

Namespaces

- namespace parma_polyhedra_library

  The PPL Java interface package.

Enumerations

- enum parma_polyhedra_library::Bounded_Integer_Type_Overflow { parma_polyhedra_library::OVERFLOW_WRAPS, parma_polyhedra_library::OVERFLOW_UNDEFINED, parma_polyhedra_library::OVERFLOW_IMPOSSIBLE }

  Overflow behavior of bounded integer types.

12.4 Bounded_Integer_Type_Representation.java File Reference

Namespaces

- namespace parma_polyhedra_library

  The PPL Java interface package.

Enumerations

- enum parma_polyhedra_library::Bounded_Integer_Type_Representation { parma_polyhedra_library::UNSIGNED, parma_polyhedra_library::SIGNED_2_COMPLEMENT }

  Representation of bounded integer types.

12.5 Bounded_Integer_Type_Width.java File Reference

Namespaces

- namespace parma_polyhedra_library

  The PPL Java interface package.

Enumerations

- enum parma_polyhedra_library::Bounded_Integer_Type_Width {

  parma_polyhedra_library::BITS_8, parma_polyhedra_library::BITS_16, parma_polyhedra_library::BITS_32, parma_polyhedra_library::BITS_64,

  parma_polyhedra_library::BITS_128 }

  Widths of bounded integer types.

12.6 By_Reference.java File Reference

Classes

- class parma_polyhedra_library::By_Reference< T >

  An utility class implementing mutable and non-mutable call-by-reference.

Namespaces

- namespace parma_polyhedra_library

  The PPL Java interface package.

12.7 Coefficient.java File Reference

Classes

- class parma_polyhedra_library::Coefficient

  A PPL coefficient.

Namespaces

- namespace parma_polyhedra_library

  The PPL Java interface package.

12.8 Complexity_Class.java File Reference

Namespaces

- namespace parma_polyhedra_library

  The PPL Java interface package.

Enumerations

- enum parma_polyhedra_library::Complexity_Class { parma_polyhedra_library::POLYNOMIAL_-COMPLEXITY, parma_polyhedra_library::SIMPLEX_COMPLEXITY, parma_polyhedra_-library::ANY_COMPLEXITY }

  Possible Complexities.

12.9 Congruence.java File Reference

Classes

- class parma_polyhedra_library::Congruence

  A linear congruence.

Namespaces

- namespace parma_polyhedra_library

  The PPL Java interface package.

12.10 Congruence_System.java File Reference

Classes

- class parma_polyhedra_library::Congruence_System

  A system of congruences.

Namespaces

- namespace parma_polyhedra_library

  The PPL Java interface package.

12.11 Constraint.java File Reference

Classes

- class parma_polyhedra_library::Constraint

  A linear equality or inequality.

Namespaces

- namespace parma_polyhedra_library

  The PPL Java interface package.

12.12 Constraint_System.java File Reference

Classes

- class parma_polyhedra_library::Constraint_System

  A system of constraints.

Namespaces

- namespace parma_polyhedra_library

  The PPL Java interface package.

12.13 Control_Parameter_Name.java File Reference

Namespaces

- namespace parma_polyhedra_library

  The PPL Java interface package.

Enumerations

- enum parma_polyhedra_library::Control_Parameter_Name { parma_polyhedra_library::PRICING }

  Names of MIP problems' control parameters.

12.14 Control_Parameter_Value.java File Reference

Namespaces

- namespace parma_polyhedra_library

  The PPL Java interface package.

Enumerations

- enum parma_polyhedra_library::Control_Parameter_Value { parma_polyhedra_library::PRICING_STEEPEST_EDGE_FLOAT, parma_polyhedra_library::PRICING_STEEPEST_EDGE_EXACT, parma_polyhedra_library::PRICING_TEXTBOOK }

  Possible values for MIP problem's control parameters.

12.15 Degenerate_Element.java File Reference

Namespaces

- namespace parma_polyhedra_library

  The PPL Java interface package.

Enumerations

- enum parma_polyhedra_library::Degenerate_Element { parma_polyhedra_library::UNIVERSE, parma_polyhedra_library::EMPTY }

  Kinds of degenerate abstract elements.

12.16 Domain_Error_Exception.java File Reference

Classes

- class parma_polyhedra_library::Domain_Error_Exception

  Exceptions caused by domain errors.

Namespaces

- namespace parma_polyhedra_library

  The PPL Java interface package.

12.17 Fake_Class_for_Doxygen.java File Reference

Classes

- class parma_polyhedra_library::Fake_Class_For_Doxygen
- class parma_polyhedra_library::Polyhedron

  The Java base class for (C and NNC) convex polyhedra.

- class parma_polyhedra_library::C_Polyhedron

  A topologically closed convex polyhedron.

- class parma_polyhedra_library::Pointset_Powerset_C_Polyhedron

  A powerset of C_Polyhedron objects.

- class parma_polyhedra_library::Pointset_Powerset_C_Polyhedron_Iterator

  An iterator class for the disjuncts of a Pointset_Powerset_C_Polyhedron.

Namespaces

- namespace parma_polyhedra_library
  The PPL Java interface package.

12.18 fdl.dox File Reference

12.19 Generator.java File Reference

Classes

- class parma_polyhedra_library::Generator
  A line, ray, point or closure point.

Namespaces

- namespace parma_polyhedra_library
  The PPL Java interface package.

12.20 Generator_System.java File Reference

Classes

- class parma_polyhedra_library::Generator_System
  A system of generators.

Namespaces

- namespace parma_polyhedra_library
  The PPL Java interface package.

12.21 Generator_Type.java File Reference

Namespaces

- namespace parma_polyhedra_library
  The PPL Java interface package.

Enumerations

- enum parma_polyhedra_library::Generator_Type { parma_polyhedra_library::LINE, parma_polyhedra_library::RAY, parma_polyhedra_library::POINT, parma_polyhedra_library::CLOSURE_POINT }

The generator type.

12.22 gpl.dox File Reference

12.23 Grid_Generator.java File Reference

Classes

- class parma_polyhedra_library::Grid_Generator

  A grid line, parameter or grid point.

Namespaces

- namespace parma_polyhedra_library

  The PPL Java interface package.

12.24 Grid_Generator_System.java File Reference

Classes

- class parma_polyhedra_library::Grid_Generator_System

  A system of grid generators.

Namespaces

- namespace parma_polyhedra_library

  The PPL Java interface package.

12.25 Grid_Generator_Type.java File Reference

Namespaces

- namespace parma_polyhedra_library

  The PPL Java interface package.

Enumerations

- enum parma_polyhedra_library::Grid_Generator_Type { parma_polyhedra_library::LINE, parma_-polyhedra_library::PARAMETER, parma_polyhedra_library::POINT }

  The grid generator type.

12.26 Invalid_Argument_Exception.java File Reference

Classes

- class parma_polyhedra_library::Invalid_Argument_Exception
  Exceptions caused by invalid arguments.

Namespaces

- namespace parma_polyhedra_library
  The PPL Java interface package.

12.27 IO.java File Reference

Classes

- class parma_polyhedra_library::IO
  A class collecting I/O functions.

Namespaces

- namespace parma_polyhedra_library
  The PPL Java interface package.

12.28 Length_Error_Exception.java File Reference

Classes

- class parma_polyhedra_library::Length_Error_Exception
  Exceptions caused by too big length/size values.

Namespaces

- namespace parma_polyhedra_library
  The PPL Java interface package.

12.29 Linear_Expression.java File Reference

Classes

- class parma_polyhedra_library::Linear_Expression
  A linear expression.

Namespaces

- namespace parma_polyhedra_library

  The PPL Java interface package.

12.30 Linear_Expression_Coefficient.java File Reference

Classes

- class parma_polyhedra_library::Linear_Expression_Coefficient

  A linear expression built from a coefficient.

Namespaces

- namespace parma_polyhedra_library

  The PPL Java interface package.

12.31 Linear_Expression_Difference.java File Reference

Classes

- class parma_polyhedra_library::Linear_Expression_Difference

  The difference of two linear expressions.

Namespaces

- namespace parma_polyhedra_library

  The PPL Java interface package.

12.32 Linear_Expression_Sum.java File Reference

Classes

- class parma_polyhedra_library::Linear_Expression_Sum

  The sum of two linear expressions.

Namespaces

- namespace parma_polyhedra_library

  The PPL Java interface package.

12.33 Linear_Expression_Times.java File Reference

Classes

- class parma_polyhedra_library::Linear_Expression_Times
  The product of a linear expression and a coefficient.

Namespaces

- namespace parma_polyhedra_library
  The PPL Java interface package.

12.34 Linear_Expression_Unary_Minus.java File Reference

Classes

- class parma_polyhedra_library::Linear_Expression_Unary_Minus
  The negation of a linear expression.

Namespaces

- namespace parma_polyhedra_library
  The PPL Java interface package.

12.35 Linear_Expression_Variable.java File Reference

Classes

- class parma_polyhedra_library::Linear_Expression_Variable
  A linear expression built from a variable.

Namespaces

- namespace parma_polyhedra_library
  The PPL Java interface package.

12.36 Logic_Error_Exception.java File Reference

Classes

- class parma_polyhedra_library::Logic_Error_Exception
  Exceptions due to errors in low-level routines.

Namespaces

- namespace parma_polyhedra_library
  The PPL Java interface package.

12.37 MIP_Problem.java File Reference

Classes

- class parma_polyhedra_library::MIP_Problem
  A Mixed Integer (linear) Programming problem.

Namespaces

- namespace parma_polyhedra_library
  The PPL Java interface package.

12.38 MIP_Problem_Status.java File Reference

Namespaces

- namespace parma_polyhedra_library
  The PPL Java interface package.

Enumerations

- enum parma_polyhedra_library::MIP_Problem_Status { parma_polyhedra_library::UNFEASIBLE_MIP_PROBLEM, parma_polyhedra_library::UNBOUNDED_MIP_PROBLEM, parma_polyhedra_library::OPTIMIZED_MIP_PROBLEM }
  Possible outcomes of the MIP_Problem solver.

12.39 Optimization_Mode.java File Reference

Namespaces

- namespace parma_polyhedra_library
  The PPL Java interface package.

Enumerations

- enum parma_polyhedra_library::Optimization_Mode { parma_polyhedra_library::MINIMIZATION, parma_polyhedra_library::MAXIMIZATION }
  Possible optimization modes.

12.40 Overflow_Error_Exception.java File Reference

Classes

- class parma_polyhedra_library::Overflow_Error_Exception
  Exceptions due to overflow errors.

Namespaces

- namespace parma_polyhedra_library
  The PPL Java interface package.

12.41 Pair.java File Reference

Classes

- class parma_polyhedra_library::Pair< K, V >
  A pair of values of type K and V.

Namespaces

- namespace parma_polyhedra_library
  The PPL Java interface package.

12.42 Parma_Polyhedra_Library.java File Reference

Classes

- class parma_polyhedra_library::Parma_Polyhedra_Library
  A class collecting library-level functions.

Namespaces

- namespace parma_polyhedra_library
  The PPL Java interface package.

12.43 Partial_Function.java File Reference

Classes

- class parma_polyhedra_library::Partial_Function
  A partial function on space dimension indices.

Namespaces

- namespace parma_polyhedra_library

  The PPL Java interface package.

12.44 PIP_Decision_Node.java File Reference

Classes

- class parma_polyhedra_library::PIP_Decision_Node

  An internal node of the PIP solution tree.

Namespaces

- namespace parma_polyhedra_library

  The PPL Java interface package.

12.45 PIP_Problem.java File Reference

Classes

- class parma_polyhedra_library::PIP_Problem

  A Parametric Integer Programming problem.

Namespaces

- namespace parma_polyhedra_library

  The PPL Java interface package.

12.46 PIP_Problem_Control_Parameter_Name.java File Reference

Namespaces

- namespace parma_polyhedra_library

  The PPL Java interface package.

Enumerations

- enum parma_polyhedra_library::PIP_Problem_Control_Parameter_Name { parma_polyhedra_library::CUTTING_STRATEGY, parma_polyhedra_library::PIVOT_ROW_STRATEGY }

  Names of PIP problems' control parameters.

12.47 PIP_Problem_Control_Parameter_Value.java File Reference

Namespaces

- namespace parma_polyhedra_library

  The PPL Java interface package.

Enumerations

- enum parma_polyhedra_library::PIP_Problem_Control_Parameter_Value {

  parma_polyhedra_library::CUTTING_STRATEGY_FIRST, parma_polyhedra_library::CUTTING_STRATEGY_DEEPEST, parma_polyhedra_library::CUTTING_STRATEGY_ALL, parma_polyhedra_library::PIVOT_ROW_STRATEGY_FIRST,

  parma_polyhedra_library::PIVOT_ROW_STRATEGY_MAX_COLUMN }

  Possible values for PIP problems' control parameters.

12.48 PIP_Problem_Status.java File Reference

Namespaces

- namespace parma_polyhedra_library

  The PPL Java interface package.

Enumerations

- enum parma_polyhedra_library::PIP_Problem_Status { parma_polyhedra_library::UNFEASIBLE_PIP_PROBLEM, parma_polyhedra_library::OPTIMIZED_PIP_PROBLEM }

  Possible outcomes of the PIP_Problem solver.

12.49 PIP_Solution_Node.java File Reference

Classes

- class parma_polyhedra_library::PIP_Solution_Node

  A leaf node of the PIP solution tree.

Namespaces

- namespace parma_polyhedra_library

  The PPL Java interface package.

12.50 PIP_Tree_Node.java File Reference

Classes

- class parma_polyhedra_library::PIP_Tree_Node
  A node of the PIP solution tree.

Namespaces

- namespace parma_polyhedra_library
  The PPL Java interface package.

12.51 Poly_Con_Relation.java File Reference

Classes

- class parma_polyhedra_library::Poly_Con_Relation
  The relation between a polyhedron and a constraint.

Namespaces

- namespace parma_polyhedra_library
  The PPL Java interface package.

12.52 Poly_Gen_Relation.java File Reference

Classes

- class parma_polyhedra_library::Poly_Gen_Relation
  The relation between a polyhedron and a generator.

Namespaces

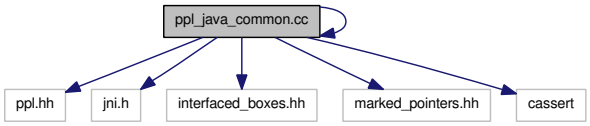
- namespace parma_polyhedra_library
  The PPL Java interface package.

12.53 ppl_java_common.cc File Reference

```
#include "ppl_java_common.defs.hh"
#include "ppl.hh"
#include <jni.h>
```

```
#include "interfaced_boxes.hh"
#include "marked_pointers.hh"
#include <cassert>
```

Include dependency graph for ppl_java_common.cc:



This graph shows which files directly or indirectly include this file:

Namespaces

- namespace Parma_Polyhedra_Library
- namespace Parma_Polyhedra_Library::Interfaces
- namespace Parma_Polyhedra_Library::Interfaces::Java

Functions

- void Parma_Polyhedra_Library::Interfaces::Java::handle_exception (JNIEnv *env, const std::overflow_error &e)
- void Parma_Polyhedra_Library::Interfaces::Java::handle_exception (JNIEnv *env, const std::invalid_argument &e)
- void Parma_Polyhedra_Library::Interfaces::Java::handle_exception (JNIEnv *env, const std::logic_error &e)
- void Parma_Polyhedra_Library::Interfaces::Java::handle_exception (JNIEnv *env, const std::length_error &e)
- void Parma_Polyhedra_Library::Interfaces::Java::handle_exception (JNIEnv *env, const std::domain_error &e)
- void Parma_Polyhedra_Library::Interfaces::Java::handle_exception (JNIEnv *env, const std::bad_alloc &)
- void Parma_Polyhedra_Library::Interfaces::Java::handle_exception (JNIEnv *env, const std::exception &e)
- void Parma_Polyhedra_Library::Interfaces::Java::handle_exception (JNIEnv *env, const timeout_exception &)
- void Parma_Polyhedra_Library::Interfaces::Java::handle_exception (JNIEnv *env, const deterministic_timeout_exception &)
- void Parma_Polyhedra_Library::Interfaces::Java::handle_exception (JNIEnv *env)
- void Parma_Polyhedra_Library::Interfaces::Java::reset_timeout ()
- void Parma_Polyhedra_Library::Interfaces::Java::reset_deterministic_timeout ()
- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_poly_gen_relation (JNIEnv *env, Poly_Gen_Relation &r)

  Builds a Java parma_polyhedra_library::Poly_Gen_Relation from C++ Poly_Gen_Relation `r`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_poly_con_relation (JNIEnv *env, Poly_Con_Relation &r)

  Builds a Java parma_polyhedra_library::Poly_Con_Relation from C++ Poly_Con_Relation `r`.

- Congruence Parma_Polyhedra_Library::Interfaces::Java::build_cxx_congruence (JNIEnv *env, jobject j_cg)

  Builds a C++ Congruence from Java parma_polyhedra_library::Congruence `j_cg`.

- PIP_Tree_Node::Artificial_Parameter Parma_Polyhedra_Library::Interfaces::Java::build_cxx_artificial_parameter (JNIEnv *env, jobject j_ap)

  Builds a C++ Artificial_Parameter from Java parma_polyhedra_library::Artificial_Parameter `j_artificial_parameter`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::bool_to_j_boolean_class (JNIEnv *env, const bool value)

  Builds a Java Boolean from C++ bool `value`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::j_long_to_j_long_class (JNIEnv *env, jlong value)

  Builds a Java Long from Java long `value`.

- jlong Parma_Polyhedra_Library::Interfaces::Java::j_long_class_to_j_long (JNIEnv *env, jobject j_long)

  Returns the Java long stored in Java Long `j_long`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::j_int_to_j_integer (JNIEnv *env, jint value)

  Builds a Java Integer from Java int `value`.

- jint Parma_Polyhedra_Library::Interfaces::Java::j_integer_to_j_int (JNIEnv *env, jobject j_integer)

  Returns the Java int stored in Java Integer `j_integer`.

- Variables_Set Parma_Polyhedra_Library::Interfaces::Java::build_cxx_variables_set (JNIEnv *env, jobject v_set)

  Builds a C++ Variables_Set from Java parma_polyhedra_library::Variables_Set `v_set`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_variables_set (JNIEnv *env, const Variables_Set &v_set)

  Builds a Java parma_polyhedra_library::Variables_Set from C++ Variables_Set `v_set`.

- Bounded_Integer_Type_Overflow Parma_Polyhedra_Library::Interfaces::Java::build_cxx_bounded_overflow (JNIEnv *env, jobject j_bounded_overflow)

  Builds a C++ Bounded_Integer_Type_Overflow from Java parma_polyhedra_library::Bounded_Integer_Type_Overflow `j_bounded_overflow`.

- Bounded_Integer_Type_Representation Parma_Polyhedra_Library::Interfaces::Java::build_cxx_bounded_rep (JNIEnv *env, jobject j_bounded_rep)

  Builds a C++ Bounded_Integer_Type_Representation from Java parma_polyhedra_library::Bounded_Integer_Type_Representation `j_bounded_rep`.

- Bounded_Integer_Type_Width Parma_Polyhedra_Library::Interfaces::Java::build_cxx_bounded_-width (JNIEnv *env, jobject j_bounded_width)

  Builds a C++ Bounded_Integer_Type_Width from Java parma_polyhedra_library::Bounded_Integer_-Type_Width `j_bounded_width`.

- Relation_Symbol Parma_Polyhedra_Library::Interfaces::Java::build_cxx_relsym (JNIEnv *env, jobject j_relsym)

  Builds a C++ Relation_Symbol from Java parma_polyhedra_library::Relation_Symbol `j_relsym`.

- Optimization_Mode Parma_Polyhedra_Library::Interfaces::Java::build_cxx_optimization_mode (JNIEnv *env, jobject j_opt_mode)

  Builds a C++ Optimization_Mode from Java parma_polyhedra_library::Optimization_Mode `j_opt_-mode`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_mip_status (JNIEnv *env, const MIP_Problem_Status &mip_status)

  Builds a Java parma_polyhedra_library::MIP_Problem_Status from C++ MIP_Problem_Status `mip_-status`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_pip_status (JNIEnv *env, const PIP_Problem_Status &pip_status)

  Builds a Java parma_polyhedra_library::PIP_Problem_Status from C++ PIP_Problem_Status `pip_-status`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_optimization_mode (JNIEnv *env, const Optimization_Mode &opt_mode)

  Builds a Java parma_polyhedra_library::Optimization_Mode from C++ Optimization_Mode `opt_mode`.

- MIP_Problem::Control_Parameter_Name Parma_Polyhedra_Library::Interfaces::Java::build_cxx_-control_parameter_name (JNIEnv *env, jobject j_cp_name)

  Builds a C++ MIP_Problem::Control_Parameter_Name from Java parma_polyhedra_library::Control_-Parameter_Name `j_cp_name`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_control_parameter_name (JNIEnv *env, const MIP_Problem::Control_Parameter_Name &cp_name)

  Builds a Java parma_polyhedra_library::Control_Parameter_Name from C++ MIP_Problem::Control_-Parameter_Name `cp_name`.

- MIP_Problem::Control_Parameter_Value Parma_Polyhedra_Library::Interfaces::Java::build_cxx_-control_parameter_value (JNIEnv *env, jobject j_cp_value)

  Builds a C++ MIP_Problem::Control_Parameter_Value from Java parma_polyhedra_library::Control_-Parameter_Value `j_cp_value`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_control_parameter_value (JNIEnv *env, const MIP_Problem::Control_Parameter_Value &cp_value)

  Builds a Java parma_polyhedra_library::Control_Parameter_Value from C++ MIP_Problem::Control_-Parameter_Value `cp_value`.

- PIP_Problem::Control_Parameter_Name Parma_Polyhedra_Library::Interfaces::Java::build_cxx_-pip_problem_control_parameter_name (JNIEnv *env, jobject j_cp_name)

  Builds a C++ PIP_Problem::Control_Parameter_Name from Java parma_polyhedra_library::PIP_-Problem_Control_Parameter_Name `j_cp_name`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_pip_problem_java_control_parameter_name (JNIEnv *env, const PIP_Problem::Control_Parameter_Name &cp_name)
- PIP_Problem::Control_Parameter_Value Parma_Polyhedra_Library::Interfaces::Java::build_cxx_pip_problem_control_parameter_value (JNIEnv *env, jobject j_cp_value)

  Builds a C++ PIP_Problem::Control_Parameter_Value from Java parma_polyhedra_library::PIP_Problem_Control_Parameter_Value `j_cp_value`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_pip_problem_control_parameter_value (JNIEnv *env, const PIP_Problem::Control_Parameter_Value &cp_value)

  Builds a Java parma_polyhedra_library::Control_Parameter_Value from C++ PIP_Problem::Control_Parameter_Value `cp_value`.

- Constraint Parma_Polyhedra_Library::Interfaces::Java::build_cxx_constraint (JNIEnv *env, jobject j_constraint)

  Builds a C++ Constraint from Java parma_polyhedra_library::Constraint `j_constraint`.

- Linear_Expression Parma_Polyhedra_Library::Interfaces::Java::build_cxx_linear_expression (JNIEnv *env, jobject j_le)

  Builds a C++ Linear_Expression from Java parma_polyhedra_library::Linear_Expression `j_le`.

- Generator Parma_Polyhedra_Library::Interfaces::Java::build_cxx_generator (JNIEnv *env, jobject j_g)

  Builds a C++ Generator from Java parma_polyhedra_library::Generator `j_g`.

- Grid_Generator Parma_Polyhedra_Library::Interfaces::Java::build_cxx_grid_generator (JNIEnv *env, jobject j_g)

  Builds a C++ Grid_Generator from Java parma_polyhedra_library::Grid_Generator `j_g`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_linear_expression_coefficient (JNIEnv *env, const Coefficient &coeff)

  Builds a Java parma_polyhedra_library::Linear_Expression_Coefficient from C++ Coefficient `coeff`.

- void Parma_Polyhedra_Library::Interfaces::Java::set_generator (JNIEnv *env, jobject dst, jobject src)

  Sets Java parma_polyhedra_library::Generator `dst` *to have the same value as* `src`.

- void Parma_Polyhedra_Library::Interfaces::Java::set_pair_element (JNIEnv *env, jobject dst_pair, int arg, jobject src)

  Assigns `src` *to one of the fields of parma_polyhedra_library::Pair object* `dst_pair`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::get_pair_element (JNIEnv *env, int arg, jobject pair)

  Returns one of the fields of the parma_polyhedra_library::Pair object `pair`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_constraint (JNIEnv *env, const Constraint &c)

  Builds a Java parma_polyhedra_library::Constraint from C++ Constraint `c`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_congruence (JNIEnv *env, const Congruence &cg)

Builds a Java parma_polyhedra_library::Congruence from C++ Congruence `cg`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_generator (JNIEnv *env, const Generator &g)

  Builds a Java parma_polyhedra_library::Generator from C++ Generator `g`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_grid_generator (JNIEnv *env, const Grid_Generator &g)

  Builds a Java parma_polyhedra_library::Grid_Generator from C++ Grid_Generator `g`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_constraint_system (JNIEnv *env, const Constraint_System &cs)

  Builds a Java parma_polyhedra_library::Constraint_System from C++ Constraint_System `cs`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_generator_system (JNIEnv *env, const Generator_System &gs)

  Builds a Java parma_polyhedra_library::Generator_System from C++ Generator_System `gs`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_grid_generator_system (JNIEnv *env, const Grid_Generator_System &gs)

  Builds a Java parma_polyhedra_library::Grid_Generator_System from C++ Grid_Generator_System `gs`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_congruence_system (JNIEnv *env, const Congruence_System &cgs)

  Builds a Java parma_polyhedra_library::Congruence_System from C++ Congruence_System `cgs`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_artificial_parameter (JNIEnv *env, const PIP_Tree_Node::Artificial_Parameter &ap)

  Builds a Java parma_polyhedra_library::Artificial_Parameter from C++ Artificial_Parameter `ap`.

Variables

- Java_Class_Cache Parma_Polyhedra_Library::Interfaces::Java::cached_classes

  The cached class references.

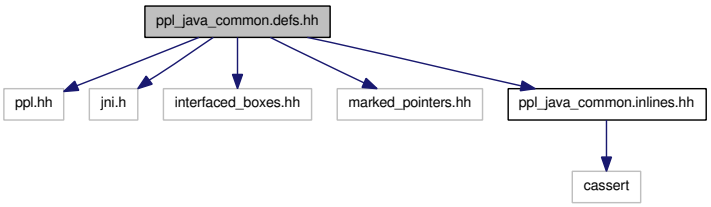
- Java_FMID_Cache Parma_Polyhedra_Library::Interfaces::Java::cached_FMIDs

  The field and method ID cache.

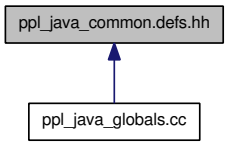
12.54 ppl_java_common.defs.hh File Reference

```
#include "ppl.hh"
#include <jni.h>
#include "interfaced_boxes.hh"
#include "marked_pointers.hh"
#include "ppl_java_common.inlines.hh"
```

Include dependency graph for ppl_java_common.defs.hh:



This graph shows which files directly or indirectly include this file:



Classes

- struct Parma_Polyhedra_Library::Interfaces::Java::Java_ExceptionOccurred
- class Parma_Polyhedra_Library::Interfaces::Java::timeout_exception
- class Parma_Polyhedra_Library::Interfaces::Java::deterministic_timeout_exception
- class Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache

  A cache for global references to Java classes.

- struct Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache

  A cache for field and method IDs of Java classes.

Namespaces

- namespace Parma_Polyhedra_Library
- namespace Parma_Polyhedra_Library::Interfaces
- namespace Parma_Polyhedra_Library::Interfaces::Java

Defines

- #define PPL_NO_AUTOMATIC_INITIALIZATION
- #define CATCH_ALL
- #define CHECK_EXCEPTION_ASSERT(env) assert(!env->ExceptionOccurred())
- #define CHECK_EXCEPTION_THROW(env)
- #define CHECK_EXCEPTION_RETURN(env, val)
- #define CHECK_EXCEPTION_RETURN_VOID(env)
- #define CHECK_RESULT_ABORT(env, cond)
- #define CHECK_RESULT_ASSERT(env, cond) assert(cond)
- #define CHECK_RESULT_THROW(env, cond)
- #define CHECK_RESULT_RETURN(env, cond, val)
- #define CHECK_RESULT_RETURN_VOID(env, cond)

Functions

- void Parma_Polyhedra_Library::Interfaces::Java::reset_timeout ()
- void Parma_Polyhedra_Library::Interfaces::Java::reset_deterministic_timeout ()
- void Parma_Polyhedra_Library::Interfaces::Java::handle_exception (JNIEnv *env, const std::logic_error &e)
- void Parma_Polyhedra_Library::Interfaces::Java::handle_exception (JNIEnv *env, const std::invalid_argument &e)
- void Parma_Polyhedra_Library::Interfaces::Java::handle_exception (JNIEnv *env, const std::domain_error &e)
- void Parma_Polyhedra_Library::Interfaces::Java::handle_exception (JNIEnv *env, const std::overflow_error &e)
- void Parma_Polyhedra_Library::Interfaces::Java::handle_exception (JNIEnv *env, const std::length_error &e)
- void Parma_Polyhedra_Library::Interfaces::Java::handle_exception (JNIEnv *env, const std::bad_alloc &)
- void Parma_Polyhedra_Library::Interfaces::Java::handle_exception (JNIEnv *env, const std::exception &e)
- void Parma_Polyhedra_Library::Interfaces::Java::handle_exception (JNIEnv *env, const timeout_exception &)
- void Parma_Polyhedra_Library::Interfaces::Java::handle_exception (JNIEnv *env, const deterministic_timeout_exception &)
- void Parma_Polyhedra_Library::Interfaces::Java::handle_exception (JNIEnv *env)
- template<typename U , typename V >
  U Parma_Polyhedra_Library::Interfaces::Java::jtype_to_unsigned (const V &value)

  Builds an unsigned C++ number from the Java native number `value`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::bool_to_j_boolean_class (JNIEnv *env, const bool value)

  Builds a Java Boolean from C++ bool `value`.

- jint Parma_Polyhedra_Library::Interfaces::Java::j_integer_to_j_int (JNIEnv *env, jobject j_integer)

  Returns the Java int stored in Java Integer `j_integer`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::j_int_to_j_integer (JNIEnv *env, jint value)

  Builds a Java Integer from Java int `value`.

- jlong Parma_Polyhedra_Library::Interfaces::Java::j_long_class_to_j_long (JNIEnv *env, jobject j_long)

  Returns the Java long stored in Java Long `j_long`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::j_long_to_j_long_class (JNIEnv *env, jlong value)

  Builds a Java Long from Java long `value`.

- bool Parma_Polyhedra_Library::Interfaces::Java::is_java_marked (JNIEnv *env, jobject ppl_object)

  Returns `true` *if and only if the Java object* `ppl_object` *refers to a C++ object.*

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_poly_gen_relation (JNIEnv *env, Poly_Gen_Relation &r)

Builds a Java parma_polyhedra_library::Poly_Gen_Relation from C++ Poly_Gen_Relation `r`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_poly_con_relation (JNIEnv *env, Poly_Con_Relation &r)

  Builds a Java parma_polyhedra_library::Poly_Con_Relation from C++ Poly_Con_Relation `r`.

- Variables_Set Parma_Polyhedra_Library::Interfaces::Java::build_cxx_variables_set (JNIEnv *env, jobject v_set)

  Builds a C++ Variables_Set from Java parma_polyhedra_library::Variables_Set `v_set`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_variables_set (JNIEnv *env, const Variables_Set &v_set)

  Builds a Java parma_polyhedra_library::Variables_Set from C++ Variables_Set `v_set`.

- Relation_Symbol Parma_Polyhedra_Library::Interfaces::Java::build_cxx_relsym (JNIEnv *env, jobject j_relsym)

  Builds a C++ Relation_Symbol from Java parma_polyhedra_library::Relation_Symbol `j_relsym`.

- Bounded_Integer_Type_Overflow Parma_Polyhedra_Library::Interfaces::Java::build_cxx_bounded_overflow (JNIEnv *env, jobject j_bounded_overflow)

  Builds a C++ Bounded_Integer_Type_Overflow from Java parma_polyhedra_library::Bounded_Integer_Type_Overflow `j_bounded_overflow`.

- Bounded_Integer_Type_Width Parma_Polyhedra_Library::Interfaces::Java::build_cxx_bounded_width (JNIEnv *env, jobject j_bounded_width)

  Builds a C++ Bounded_Integer_Type_Width from Java parma_polyhedra_library::Bounded_Integer_Type_Width `j_bounded_width`.

- Bounded_Integer_Type_Representation Parma_Polyhedra_Library::Interfaces::Java::build_cxx_bounded_rep (JNIEnv *env, jobject j_bounded_rep)

  Builds a C++ Bounded_Integer_Type_Representation from Java parma_polyhedra_library::Bounded_Integer_Type_Representation `j_bounded_rep`.

- Optimization_Mode Parma_Polyhedra_Library::Interfaces::Java::build_cxx_optimization_mode (JNIEnv *env, jobject j_opt_mode)

  Builds a C++ Optimization_Mode from Java parma_polyhedra_library::Optimization_Mode `j_opt_mode`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_optimization_mode (JNIEnv *env, const Optimization_Mode &opt_mode)

  Builds a Java parma_polyhedra_library::Optimization_Mode from C++ Optimization_Mode `opt_mode`.

- MIP_Problem::Control_Parameter_Name Parma_Polyhedra_Library::Interfaces::Java::build_cxx_control_parameter_name (JNIEnv *env, jobject j_cp_name)

  Builds a C++ MIP_Problem::Control_Parameter_Name from Java parma_polyhedra_library::Control_Parameter_Name `j_cp_name`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_control_parameter_name (JNIEnv *env, const MIP_Problem::Control_Parameter_Name &cp_name)

  Builds a Java parma_polyhedra_library::Control_Parameter_Name from C++ MIP_Problem::Control_Parameter_Name `cp_name`.

- MIP_Problem::Control_Parameter_Value Parma_Polyhedra_Library::Interfaces::Java::build_cxx_control_parameter_value (JNIEnv *env, jobject j_cp_value)

  Builds a C++ MIP_Problem::Control_Parameter_Value from Java parma_polyhedra_library::Control_Parameter_Value `j_cp_value`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_control_parameter_value (JNIEnv *env, const MIP_Problem::Control_Parameter_Value &cp_value)

  Builds a Java parma_polyhedra_library::Control_Parameter_Value from C++ MIP_Problem::Control_Parameter_Value `cp_value`.

- PIP_Problem::Control_Parameter_Name Parma_Polyhedra_Library::Interfaces::Java::build_cxx_pip_problem_control_parameter_name (JNIEnv *env, jobject j_cp_name)

  Builds a C++ PIP_Problem::Control_Parameter_Name from Java parma_polyhedra_library::PIP_Problem_Control_Parameter_Name `j_cp_name`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_pip_problem_control_parameter_name (JNIEnv *env, const PIP_Problem::Control_Parameter_Name &cp_name)

  Builds a Java parma_polyhedra_library::PIP_Problem_Control_Parameter_Name from C++ PIP_Problem::Control_Parameter_Name `cp_name`.

- PIP_Problem::Control_Parameter_Value Parma_Polyhedra_Library::Interfaces::Java::build_cxx_pip_problem_control_parameter_value (JNIEnv *env, jobject j_cp_value)

  Builds a C++ PIP_Problem::Control_Parameter_Value from Java parma_polyhedra_library::PIP_Problem_Control_Parameter_Value `j_cp_value`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_pip_problem_control_parameter_value (JNIEnv *env, const PIP_Problem::Control_Parameter_Value &cp_value)

  Builds a Java parma_polyhedra_library::Control_Parameter_Value from C++ PIP_Problem::Control_Parameter_Value `cp_value`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_mip_status (JNIEnv *env, const MIP_Problem_Status &mip_status)

  Builds a Java parma_polyhedra_library::MIP_Problem_Status from C++ MIP_Problem_Status `mip_status`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_pip_status (JNIEnv *env, const PIP_Problem_Status &pip_status)

  Builds a Java parma_polyhedra_library::PIP_Problem_Status from C++ PIP_Problem_Status `pip_status`.

- Variable Parma_Polyhedra_Library::Interfaces::Java::build_cxx_variable (JNIEnv *env, jobject j_var)

  Builds a C++ Variable from Java parma_polyhedra_library::Variable `j_var`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_variable (JNIEnv *env, const Variable &var)

  Builds a Java parma_polyhedra_library::Variable from C++ Variable `var`.

- Coefficient Parma_Polyhedra_Library::Interfaces::Java::build_cxx_coeff (JNIEnv *env, jobject j_coeff)

  Builds a C++ Coefficient from Java parma_polyhedra_library::Coefficient `j_coeff`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_coeff (JNIEnv *env, const Coefficient &ppl_coeff)

  Builds a Java parma_polyhedra_library::Coefficient from C++ Coefficient `ppl_coeff`.

- Constraint Parma_Polyhedra_Library::Interfaces::Java::build_cxx_constraint (JNIEnv *env, jobject j_constraint)

  Builds a C++ Constraint from Java parma_polyhedra_library::Constraint `j_constraint`.

- PIP_Tree_Node::Artificial_Parameter Parma_Polyhedra_Library::Interfaces::Java::build_cxx_artificial_parameter (JNIEnv *env, jobject j_ap)

  Builds a C++ Artificial_Parameter from Java parma_polyhedra_library::Artificial_Parameter `j_artificial_parameter`.

- Linear_Expression Parma_Polyhedra_Library::Interfaces::Java::build_cxx_linear_expression (JNIEnv *env, jobject j_le)

  Builds a C++ Linear_Expression from Java parma_polyhedra_library::Linear_Expression `j_le`.

- Congruence Parma_Polyhedra_Library::Interfaces::Java::build_cxx_congruence (JNIEnv *env, jobject j_cg)

  Builds a C++ Congruence from Java parma_polyhedra_library::Congruence `j_cg`.

- Generator Parma_Polyhedra_Library::Interfaces::Java::build_cxx_generator (JNIEnv *env, jobject j_g)

  Builds a C++ Generator from Java parma_polyhedra_library::Generator `j_g`.

- Grid_Generator Parma_Polyhedra_Library::Interfaces::Java::build_cxx_grid_generator (JNIEnv *env, jobject j_g)

  Builds a C++ Grid_Generator from Java parma_polyhedra_library::Grid_Generator `j_g`.

- Grid_Generator_System Parma_Polyhedra_Library::Interfaces::Java::build_cxx_grid_generator_system (JNIEnv *env, jobject j_gs)

  Builds a C++ Grid_Generator_System from Java parma_polyhedra_library::Grid_Generator_System `j_gs`.

- Constraint_System Parma_Polyhedra_Library::Interfaces::Java::build_cxx_constraint_system (JNIEnv *env, jobject j_cs)

  Builds a C++ Constraint_System from Java parma_polyhedra_library::Constraint_System `j_cs`.

- PIP_Tree_Node::Artificial_Parameter_Sequence Parma_Polyhedra_Library::Interfaces::Java::build_cxx_artificial_parameter_sequence (JNIEnv *env, jobject j_aps)

  Builds a C++ Artificial_Parameter_Sequence from Java parma_polyhedra_library::Artificial_Parameter_Sequence `j_aps`.

- Generator_System Parma_Polyhedra_Library::Interfaces::Java::build_cxx_generator_system (JNIEnv *env, jobject j_gs)

  Builds a C++ Generator_System from Java parma_polyhedra_library::Generator_System `j_gs`.

- Congruence_System Parma_Polyhedra_Library::Interfaces::Java::build_cxx_congruence_system (JNIEnv *env, jobject j_cgs)

  Builds a C++ Congruence_System from Java parma_polyhedra_library::Congruence_System `j_cgs`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_constraint (JNIEnv *env, const Constraint &c)

  Builds a Java parma_polyhedra_library::Constraint from C++ Constraint `c`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_artificial_parameter (JNIEnv *env, const PIP_Tree_Node::Artificial_Parameter &ap)

  Builds a Java parma_polyhedra_library::Artificial_Parameter from C++ Artificial_Parameter `ap`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_congruence (JNIEnv *env, const Congruence &cg)

  Builds a Java parma_polyhedra_library::Congruence from C++ Congruence `cg`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_generator (JNIEnv *env, const Generator &g)

  Builds a Java parma_polyhedra_library::Generator from C++ Generator `g`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_grid_generator (JNIEnv *env, const Grid_Generator &g)

  Builds a Java parma_polyhedra_library::Grid_Generator from C++ Grid_Generator `g`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_constraint_system (JNIEnv *env, const Constraint_System &cs)

  Builds a Java parma_polyhedra_library::Constraint_System from C++ Constraint_System `cs`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_artificial_parameter_sequence (JNIEnv *env, const PIP_Tree_Node::Artificial_Parameter_Sequence &aps)

  Builds a Java parma_polyhedra_library::Artificial_Parameter_Sequence from C++ Artificial_Parameter_Sequence `aps`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_grid_generator_system (JNIEnv *env, const Grid_Generator_System &gs)

  Builds a Java parma_polyhedra_library::Grid_Generator_System from C++ Grid_Generator_System `gs`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_generator_system (JNIEnv *env, const Generator_System &gs)

  Builds a Java parma_polyhedra_library::Generator_System from C++ Generator_System `gs`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_congruence_system (JNIEnv *env, const Congruence_System &cgs)

  Builds a Java parma_polyhedra_library::Congruence_System from C++ Congruence_System `cgs`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_linear_expression_coefficient (JNIEnv *env, const Coefficient &coeff)

  Builds a Java parma_polyhedra_library::Linear_Expression_Coefficient from C++ Coefficient `coeff`.

- void Parma_Polyhedra_Library::Interfaces::Java::set_generator (JNIEnv *env, jobject dst, jobject src)

  Sets Java parma_polyhedra_library::Generator `dst` *to have the same value as* `src`.

- void Parma_Polyhedra_Library::Interfaces::Java::set_coefficient (JNIEnv *env, jobject dst, jobject src)

Sets Java Coefficient `dst` *to have the same value as* `src`.

- void Parma_Polyhedra_Library::Interfaces::Java::set_by_reference (JNIEnv *env, jobject by_ref_dst, jobject src)

  Modifies parma_polyhedra_library::By_Reference object `by_ref_dst` *so that it references object* `src`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::get_by_reference (JNIEnv *env, jobject by_reference)

  Returns the object referenced by parma_polyhedra_library::By_Reference object `by_reference`.

- void Parma_Polyhedra_Library::Interfaces::Java::set_pair_element (JNIEnv *env, jobject dst_pair, int arg, jobject src)

  Assigns `src` *to one of the fields of parma_polyhedra_library::Pair object* `dst_pair`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::get_pair_element (JNIEnv *env, int arg, jobject pair)

  Returns one of the fields of the parma_polyhedra_library::Pair object `pair`.

- void * Parma_Polyhedra_Library::Interfaces::Java::get_ptr (JNIEnv *env, jobject ppl_object)

  Returns a pointer to the C++ object wrapped by `ppl_object`.

- template<typename T >
  void Parma_Polyhedra_Library::Interfaces::Java::set_ptr (JNIEnv *env, jobject ppl_object, const T *address, bool to_be_marked=false)

  Sets the pointer of the underlying C++ object in the Java object.

- template<typename R >
  jobject Parma_Polyhedra_Library::Interfaces::Java::build_linear_expression (JNIEnv *env, const R &r)

  Builds a Java parma_polyhedra_library::Linear_Expression from the C++ constraint/congruence `r`.

12.54.1 Define Documentation

12.54.1.1 #define CATCH_ALL

Definition at line 38 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Artificial_1Parameter_ascii_1dump(), Java_parma_1polyhedra_1library_Congruence_1System_ascii_1dump(), Java_parma_1polyhedra_1library_Congruence_ascii_1dump(), Java_parma_1polyhedra_1library_Constraint_1System_ascii_1dump(), Java_parma_1polyhedra_1library_Constraint_ascii_1dump(), Java_parma_1polyhedra_1library_Generator_1System_ascii_1dump(), Java_parma_1polyhedra_1library_Generator_ascii_1dump(), Java_parma_1polyhedra_1library_Grid_1Generator_1System_ascii_1dump(), Java_parma_1polyhedra_1library_Grid_1Generator_ascii_1dump(), Java_parma_1polyhedra_1library_IO_wrap_1string(), Java_parma_1polyhedra_1library_Linear_1Expression_all_1homogeneous_1terms_1are_1zero(), Java_parma_1polyhedra_1library_Linear_1Expression_ascii_1dump(), Java_parma_1polyhedra_1library_Linear_1Expression_is_1zero(), Java_parma_1polyhedra_1library_MIP_1Problem_add_1constraint(), Java_parma_1polyhedra_1library_MIP_1Problem_add_1constraints(), Java_parma_1polyhedra_1library_MIP_1Problem_add_1space_1dimensions_1and_1embed(), Java_parma_

1polyhedra_1library_MIP_1Problem_add_1to_1integer_1space_1dimensions(), Java_parma_-1polyhedra_1library_MIP_1Problem_ascii_1dump(), Java_parma_1polyhedra_1library_MIP_-1Problem_build_1cpp_1object__J(), Java_parma_1polyhedra_1library_MIP_1Problem_build_1cpp_-1object__JLparma_1polyhedra_1library_Constraint_1System_2Lparma_1polyhedra_1library_Linear_-1Expression_2Lparma_1polyhedra_1library_Optimization_1Mode_2(), Java_parma_1polyhedra_-1library_MIP_1Problem_clear(), Java_parma_1polyhedra_1library_MIP_1Problem_constraints(), Java_parma_1polyhedra_1library_MIP_1Problem_evaluate_1objective_1function(), Java_parma_-1polyhedra_1library_MIP_1Problem_feasible_1point(), Java_parma_1polyhedra_1library_MIP_-1Problem_get_1control_1parameter(), Java_parma_1polyhedra_1library_MIP_1Problem_integer_-1space_1dimensions(), Java_parma_1polyhedra_1library_MIP_1Problem_is_1satisfiable(), Java_-parma_1polyhedra_1library_MIP_1Problem_max_1space_1dimension(), Java_parma_1polyhedra_-1library_MIP_1Problem_objective_1function(), Java_parma_1polyhedra_1library_MIP_1Problem_-OK(), Java_parma_1polyhedra_1library_MIP_1Problem_optimal_1value(), Java_parma_1polyhedra_-1library_MIP_1Problem_optimization_1mode(), Java_parma_1polyhedra_1library_MIP_1Problem_-optimizing_1point(), Java_parma_1polyhedra_1library_MIP_1Problem_set_1control_1parameter(), Java_parma_1polyhedra_1library_MIP_1Problem_set_1objective_1function(), Java_parma_1polyhedra_-1library_MIP_1Problem_set_1optimization_1mode(), Java_parma_1polyhedra_1library_MIP_-1Problem_solve(), Java_parma_1polyhedra_1library_MIP_1Problem_space_1dimension(), Java_-parma_1polyhedra_1library_MIP_1Problem_total_1memory_1in_1bytes(), Java_parma_1polyhedra_-1library_Parma_1Polyhedra_1Library_irrational_1precision(), Java_parma_1polyhedra_1library_-Parma_1Polyhedra_1Library_reset_1deterministic_1timeout(), Java_parma_1polyhedra_1library_-Parma_1Polyhedra_1Library_reset_1timeout(), Java_parma_1polyhedra_1library_Parma_1Polyhedra_-1Library_restore_1pre_1PPL_1rounding(), Java_parma_1polyhedra_1library_Parma_1Polyhedra_-1Library_set_1deterministic_1timeout(), Java_parma_1polyhedra_1library_Parma_1Polyhedra_-1Library_set_1irrational_1precision(), Java_parma_1polyhedra_1library_Parma_1Polyhedra_1Library_-set_1rounding_1for_1PPL(), Java_parma_1polyhedra_1library_Parma_1Polyhedra_1Library_set_-1timeout(), Java_parma_1polyhedra_1library_Partial_1Function_build_1cpp_1object(), Java_parma_-1polyhedra_1library_Partial_1Function_has_1empty_1codomain(), Java_parma_1polyhedra_1library_-Partial_1Function_insert(), Java_parma_1polyhedra_1library_Partial_1Function_max_1in_1codomain(), Java_parma_1polyhedra_1library_PIP_1Decision_1Node_child_1node(), Java_parma_1polyhedra_-1library_PIP_1Problem_add_1constraint(), Java_parma_1polyhedra_1library_PIP_1Problem_add_-1constraints(), Java_parma_1polyhedra_1library_PIP_1Problem_add_1space_1dimensions_1and_-1embed(), Java_parma_1polyhedra_1library_PIP_1Problem_add_1to_1parameter_1space_1dimensions(), Java_parma_1polyhedra_1library_PIP_1Problem_ascii_1dump(), Java_parma_1polyhedra_1library_-PIP_1Problem_build_1cpp_1object__J(), Java_parma_1polyhedra_1library_PIP_1Problem_build_-1cpp_1object__JLparma_1polyhedra_1library_Constraint_1System_2Lparma_1polyhedra_1library_-Variables_1Set_2(), Java_parma_1polyhedra_1library_PIP_1Problem_constraint_1at_1index(), Java_-parma_1polyhedra_1library_PIP_1Problem_constraints(), Java_parma_1polyhedra_1library_PIP_-1Problem_external_1memory_1in_1bytes(), Java_parma_1polyhedra_1library_PIP_1Problem_get_-1big_1parameter_1dimension(), Java_parma_1polyhedra_1library_PIP_1Problem_get_1pip_1problem_-1control_1parameter(), Java_parma_1polyhedra_1library_PIP_1Problem_is_1satisfiable(), Java_parma_-1polyhedra_1library_PIP_1Problem_max_1space_1dimension(), Java_parma_1polyhedra_1library_-PIP_1Problem_number_1of_1constraints(), Java_parma_1polyhedra_1library_PIP_1Problem_number_-1of_1parameter_1space_1dimensions(), Java_parma_1polyhedra_1library_PIP_1Problem_OK(), Java_-parma_1polyhedra_1library_PIP_1Problem_optimizing_1solution(), Java_parma_1polyhedra_1library_-PIP_1Problem_parameter_1space_1dimensions(), Java_parma_1polyhedra_1library_PIP_1Problem_-set_1big_1parameter_1dimension(), Java_parma_1polyhedra_1library_PIP_1Problem_set_1pip_-1problem_1control_1parameter(), Java_parma_1polyhedra_1library_PIP_1Problem_solution(), Java_-parma_1polyhedra_1library_PIP_1Problem_solve(), Java_parma_1polyhedra_1library_PIP_1Problem_-space_1dimension(), Java_parma_1polyhedra_1library_PIP_1Problem_total_1memory_1in_1bytes(), Java_parma_1polyhedra_1library_PIP_1Tree_1Node_artificials(), Java_parma_1polyhedra_1library_-PIP_1Tree_1Node_as_1decision(), Java_parma_1polyhedra_1library_PIP_1Tree_1Node_as_1solution(), Java_parma_1polyhedra_1library_PIP_1Tree_1Node_constraints(), Java_parma_1polyhedra_1library_-PIP_1Tree_1Node_number_1of_1artificials(), and Java_parma_1polyhedra_1library_PIP_1Tree_1Node_-

OK().

12.54.1.2 #define CHECK_EXCEPTION_ASSERT(env) assert(!env->ExceptionOccurred())

Definition at line 72 of file ppl_java_common.defs.hh.

Referenced by Parma_Polyhedra_Library::Interfaces::Java::build_cxx_system().

12.54.1.3 #define CHECK_EXCEPTION_RETURN(env, val)

Value:

```
do {                                    \
    if (env->ExceptionOccurred())         \
      return val;                         \
  } while (0)
```

Definition at line 79 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_MIP_1Problem_constraints(), Java_parma_-1polyhedra_1library_PIP_1Problem_constraints(), and Java_parma_1polyhedra_1library_PIP_1Tree_-1Node_artificials().

12.54.1.4 #define CHECK_EXCEPTION_RETURN_VOID(env)

Value:

```
do {                                    \
    if (env->ExceptionOccurred())         \
      return;                             \
  } while (0)
```

Definition at line 84 of file ppl_java_common.defs.hh.

12.54.1.5 #define CHECK_EXCEPTION_THROW(env)

Value:

```
do {                                    \
    if (env->ExceptionOccurred())         \
      throw Java_ExceptionOccurred();     \
  } while (0)
```

Definition at line 74 of file ppl_java_common.defs.hh.

Referenced by Parma_Polyhedra_Library::Interfaces::Java::build_cxx_coeff(), Parma_Polyhedra_-Library::Interfaces::Java::build_cxx_system(), and Parma_Polyhedra_Library::Interfaces::Java::build_-linear_expression().

12.54.1.6 #define CHECK_RESULT_ABORT(env, cond)

Value:

```
do {                                    \
    if (!(cond))                          \
      abort();                            \
  } while (0)
```

Definition at line 89 of file ppl_java_common.defs.hh.

12.54.1.7 #define CHECK_RESULT_ASSERT(env, cond) assert(cond)

Definition at line 94 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_Artificial_1Parameter_1Sequence_initIDs(), Java_parma_1polyhedra_1library_Artificial_1Parameter_initIDs(), Java_parma_1polyhedra_1library_Bounded_1Integer_1Type_1Overflow_initIDs(), Java_parma_1polyhedra_1library_Bounded_1Integer_1Type_1Representation_initIDs(), Java_parma_1polyhedra_1library_Bounded_1Integer_1Type_1Width_initIDs(), Java_parma_1polyhedra_1library_By_1Reference_initIDs(), Java_parma_1polyhedra_1library_Coefficient_initIDs(), Java_parma_1polyhedra_1library_Complexity_1Class_initIDs(), Java_parma_1polyhedra_1library_Congruence_1System_initIDs(), Java_parma_1polyhedra_1library_Congruence_initIDs(), Java_parma_1polyhedra_1library_Constraint_1System_initIDs(), Java_parma_1polyhedra_1library_Constraint_initIDs(), Java_parma_1polyhedra_1library_Degenerate_1Element_initIDs(), Java_parma_1polyhedra_1library_Generator_1System_initIDs(), Java_parma_1polyhedra_1library_Generator_1Type_initIDs(), Java_parma_1polyhedra_1library_Generator_initIDs(), Java_parma_1polyhedra_1library_Grid_1Generator_1System_initIDs(), Java_parma_1polyhedra_1library_Grid_1Generator_1Type_initIDs(), Java_parma_1polyhedra_1library_Grid_1Generator_initIDs(), Java_parma_1polyhedra_1library_Linear_1Expression_1Coefficient_initIDs(), Java_parma_1polyhedra_1library_Linear_1Expression_1Difference_initIDs(), Java_parma_1polyhedra_1library_Linear_1Expression_1Sum_initIDs(), Java_parma_1polyhedra_1library_Linear_1Expression_1Times_initIDs(), Java_parma_1polyhedra_1library_Linear_1Expression_1Unary_1Minus_initIDs(), Java_parma_1polyhedra_1library_Linear_1Expression_1Variable_initIDs(), Java_parma_1polyhedra_1library_Linear_1Expression_initIDs(), Java_parma_1polyhedra_1library_MIP_1Problem_1Status_initIDs(), Java_parma_1polyhedra_1library_Optimization_1Mode_initIDs(), Java_parma_1polyhedra_1library_Pair_initIDs(), Java_parma_1polyhedra_1library_PIP_1Decision_1Node_child_1node(), Java_parma_1polyhedra_1library_PIP_1Problem_1Status_initIDs(), Java_parma_1polyhedra_1library_PIP_1Problem_optimizing_1solution(), Java_parma_1polyhedra_1library_PIP_1Problem_solution(), Java_parma_1polyhedra_1library_PIP_1Tree_1Node_as_1decision(), Java_parma_1polyhedra_1library_PIP_1Tree_1Node_as_1solution(), Java_parma_1polyhedra_1library_Poly_1Con_1Relation_initIDs(), Java_parma_1polyhedra_1library_Poly_1Gen_1Relation_initIDs(), Java_parma_1polyhedra_1library_PPL_1Object_initIDs(), Java_parma_1polyhedra_1library_Relation_1Symbol_initIDs(), Java_parma_1polyhedra_1library_Variable_initIDs(), and Java_parma_1polyhedra_1library_Variables_1Set_initIDs().

12.54.1.8 #define CHECK_RESULT_RETURN(env, cond, val)

Value:

```
do {                                    \
    if (!(cond))                          \
      return val;                         \
  } while (0)
```

Definition at line 101 of file ppl_java_common.defs.hh.

Referenced by Java_parma_1polyhedra_1library_IO_wrap_1string(), Java_parma_1polyhedra_1library_MIP_1Problem_constraints(), Java_parma_1polyhedra_1library_MIP_1Problem_objective_1function(),

Java_parma_1polyhedra_1library_PIP_1Decision_1Node_child_1node(), Java_parma_1polyhedra_-1library_PIP_1Problem_constraints(), Java_parma_1polyhedra_1library_PIP_1Problem_optimizing_-1solution(), Java_parma_1polyhedra_1library_PIP_1Problem_solution(), Java_parma_1polyhedra_-1library_PIP_1Tree_1Node_artificials(), Java_parma_1polyhedra_1library_PIP_1Tree_1Node_as_-1decision(), Java_parma_1polyhedra_1library_PIP_1Tree_1Node_as_1solution(), and Java_parma_-1polyhedra_1library_PIP_1Tree_1Node_constraints().

12.54.1.9 #define CHECK_RESULT_RETURN_VOID(env, cond)

Value:

```
do {                                        \
    if (!(cond))                              \
      return;                                 \
  } while (0)
```

Definition at line 106 of file ppl_java_common.defs.hh.

12.54.1.10 #define CHECK_RESULT_THROW(env, cond)

Value:

```
do {                                    \
    if (!(cond))                          \
      throw Java_ExceptionOccurred();     \
  } while (0)
```

Definition at line 96 of file ppl_java_common.defs.hh.

Referenced by Parma_Polyhedra_Library::Interfaces::Java::build_cxx_coeff(), Parma_Polyhedra_-Library::Interfaces::Java::build_java_coeff(), Parma_Polyhedra_Library::Interfaces::Java::build_java_-variable(), and Parma_Polyhedra_Library::Interfaces::Java::build_linear_expression().

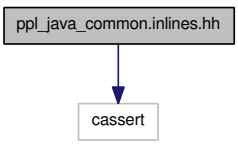
12.54.1.11 #define PPL_NO_AUTOMATIC_INITIALIZATION

Definition at line 27 of file ppl_java_common.defs.hh.

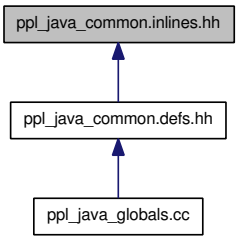
12.55 ppl_java_common.inlines.hh File Reference

```
#include <cassert>
```

Include dependency graph for ppl_java_common.inlines.hh:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace Parma_Polyhedra_Library
- namespace Parma_Polyhedra_Library::Interfaces
- namespace Parma_Polyhedra_Library::Interfaces::Java

Functions

- template<typename U , typename V >
  U Parma_Polyhedra_Library::Interfaces::Java::jtype_to_unsigned (const V &value)

  Builds an unsigned C++ number from the Java native number `value`.

- template<typename T >
  void Parma_Polyhedra_Library::Interfaces::Java::set_ptr (JNIEnv *env, jobject ppl_object, const T *address, bool to_be_marked=false)

  Sets the pointer of the underlying C++ object in the Java object.

- void * Parma_Polyhedra_Library::Interfaces::Java::get_ptr (JNIEnv *env, jobject ppl_object)

  Returns a pointer to the C++ object wrapped by `ppl_object`.

- bool Parma_Polyhedra_Library::Interfaces::Java::is_java_marked (JNIEnv *env, jobject ppl_object)

  Returns `true` *if and only if the Java object* `ppl_object` *refers to a C++ object.*

- void Parma_Polyhedra_Library::Interfaces::Java::set_coefficient (JNIEnv *env, jobject dst, jobject src)

  Sets Java Coefficient `dst` *to have the same value as* `src`.

- void Parma_Polyhedra_Library::Interfaces::Java::set_by_reference (JNIEnv *env, jobject by_ref_dst, jobject src)

  Modifies parma_polyhedra_library::By_Reference object `by_ref_dst` *so that it references object* `src`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::get_by_reference (JNIEnv *env, jobject by_reference)

  Returns the object referenced by parma_polyhedra_library::By_Reference object `by_reference`.

- template<typename R >
  jobject Parma_Polyhedra_Library::Interfaces::Java::build_linear_expression (JNIEnv *env, const R &r)

Builds a Java parma_polyhedra_library::Linear_Expression from the C++ constraint/congruence `r`.

- Variable Parma_Polyhedra_Library::Interfaces::Java::build_cxx_variable (JNIEnv *env, jobject j_var)

  Builds a C++ Variable from Java parma_polyhedra_library::Variable `j_var`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_variable (JNIEnv *env, const Variable &var)

  Builds a Java parma_polyhedra_library::Variable from C++ Variable `var`.

- Coefficient Parma_Polyhedra_Library::Interfaces::Java::build_cxx_coeff (JNIEnv *env, jobject j_coeff)

  Builds a C++ Coefficient from Java parma_polyhedra_library::Coefficient `j_coeff`.

- jobject Parma_Polyhedra_Library::Interfaces::Java::build_java_coeff (JNIEnv *env, const Coefficient &ppl_coeff)

  Builds a Java parma_polyhedra_library::Coefficient from C++ Coefficient `ppl_coeff`.

- template<typename System , typename Elem_Builder >
  System Parma_Polyhedra_Library::Interfaces::Java::build_cxx_system (JNIEnv *env, jobject j_iterable, Elem_Builder build_cxx_elem)
- Congruence_System Parma_Polyhedra_Library::Interfaces::Java::build_cxx_congruence_system (JNIEnv *env, jobject j_cgs)

  Builds a C++ Congruence_System from Java parma_polyhedra_library::Congruence_System `j_cgs`.

- Constraint_System Parma_Polyhedra_Library::Interfaces::Java::build_cxx_constraint_system (JNIEnv *env, jobject j_cs)

  Builds a C++ Constraint_System from Java parma_polyhedra_library::Constraint_System `j_cs`.

- Generator_System Parma_Polyhedra_Library::Interfaces::Java::build_cxx_generator_system (JNIEnv *env, jobject j_gs)

  Builds a C++ Generator_System from Java parma_polyhedra_library::Generator_System `j_gs`.

- Grid_Generator_System Parma_Polyhedra_Library::Interfaces::Java::build_cxx_grid_generator_system (JNIEnv *env, jobject j_gs)

  Builds a C++ Grid_Generator_System from Java parma_polyhedra_library::Grid_Generator_System `j_gs`.

12.56 ppl_java_globals.cc File Reference

```
#include "ppl_java_common.defs.hh"
#include "parma_polyhedra_library_Artificial_Parameter.h"
#include "parma_polyhedra_library_Artificial_Parameter_Sequence.h"
#include "parma_polyhedra_library_Bounded_Integer_Type_Overflow.h"
#include "parma_polyhedra_library_Bounded_Integer_Type_-
Representation.h"
#include "parma_polyhedra_library_Bounded_Integer_Type_Width.h"
```

```
#include "parma_polyhedra_library_By_Reference.h"
#include "parma_polyhedra_library_Coefficient.h"
#include "parma_polyhedra_library_Complexity_Class.h"
#include "parma_polyhedra_library_Congruence.h"
#include "parma_polyhedra_library_Congruence_System.h"
#include "parma_polyhedra_library_Constraint.h"
#include "parma_polyhedra_library_Constraint_System.h"
#include "parma_polyhedra_library_Degenerate_Element.h"
#include "parma_polyhedra_library_Generator.h"
#include "parma_polyhedra_library_Generator_System.h"
#include "parma_polyhedra_library_Generator_Type.h"
#include "parma_polyhedra_library_Grid_Generator.h"
#include "parma_polyhedra_library_Grid_Generator_System.h"
#include "parma_polyhedra_library_Grid_Generator_Type.h"
#include "parma_polyhedra_library_IO.h"
#include "parma_polyhedra_library_Linear_Expression.h"
#include "parma_polyhedra_library_Linear_Expression_Coefficient.h"
#include "parma_polyhedra_library_Linear_Expression_Difference.h"
#include "parma_polyhedra_library_Linear_Expression_Sum.h"
#include "parma_polyhedra_library_Linear_Expression_Times.h"
#include "parma_polyhedra_library_Linear_Expression_Unary_Minus.h"
#include "parma_polyhedra_library_Linear_Expression_Variable.h"
#include "parma_polyhedra_library_MIP_Problem.h"
#include "parma_polyhedra_library_MIP_Problem_Status.h"
#include "parma_polyhedra_library_Optimization_Mode.h"
#include "parma_polyhedra_library_Pair.h"
#include "parma_polyhedra_library_Parma_Polyhedra_Library.h"
#include "parma_polyhedra_library_Partial_Function.h"
#include "parma_polyhedra_library_PIP_Problem.h"
#include "parma_polyhedra_library_PIP_Problem_Status.h"
#include "parma_polyhedra_library_PIP_Decision_Node.h"
#include "parma_polyhedra_library_PIP_Solution_Node.h"
#include "parma_polyhedra_library_PIP_Tree_Node.h"
#include "parma_polyhedra_library_Poly_Con_Relation.h"
#include "parma_polyhedra_library_Poly_Gen_Relation.h"
#include "parma_polyhedra_library_PPL_Object.h"
```

```
#include "parma_polyhedra_library_Relation_Symbol.h"
#include "parma_polyhedra_library_Variable.h"
#include "parma_polyhedra_library_Variables_Set.h"
```

Include dependency graph for ppl_java_globals.cc:

Functions

- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Parma_1Polyhedra_1Library_-initialize_1library (JNIEnv ∗env, jclass)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Parma_1Polyhedra_1Library_-finalize_1library (JNIEnv ∗env, jclass)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_By_1Reference_initIDs (JNIEnv ∗env, jclass j_by_ref_class)
- JNIEXPORT jint JNICALL Java_parma_1polyhedra_1library_Coefficient_bits (JNIEnv ∗, jclass)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Coefficient_initIDs (JNIEnv ∗env, jclass j_coeff_class)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Complexity_1Class_initIDs (JNIEnv ∗env, jclass j_complexity_class)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Congruence_initIDs (JNIEnv ∗env, jclass j_congruence_class)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Congruence_1System_initIDs (JNIEnv ∗env, jclass j_con_sys_class)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Constraint_initIDs (JNIEnv ∗env, jclass j_constraint_class)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Constraint_1System_initIDs (JNIEnv ∗env, jclass j_con_sys_class)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Degenerate_1Element_initIDs (JNIEnv ∗env, jclass j_degenerate_class)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Generator_initIDs (JNIEnv ∗env, jclass j_generator_class)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Generator_1System_initIDs (JNIEnv ∗env, jclass j_gen_sys_class)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Generator_1Type_initIDs (JNIEnv ∗env, jclass j_gen_type_class)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Grid_1Generator_initIDs (JNIEnv ∗env, jclass j_grid_generator_class)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Grid_1Generator_1System_initIDs (JNIEnv ∗env, jclass j_gen_sys_class)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Grid_1Generator_1Type_initIDs (JNIEnv ∗env, jclass j_grid_gen_type_class)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Linear_1Expression_initIDs (JNIEnv ∗env, jclass j_le_class)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Linear_1Expression_1Coefficient_-initIDs (JNIEnv ∗env, jclass j_le_coeff_class)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Linear_1Expression_1Difference_-initIDs (JNIEnv ∗env, jclass j_le_diff_class)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Linear_1Expression_1Sum_initIDs (JNIEnv ∗env, jclass j_le_sum_class)

- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Linear_1Expression_1Times_-initIDs (JNIEnv *env, jclass j_le_times_class)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Linear_1Expression_1Unary_-1Minus_initIDs (JNIEnv *env, jclass j_le_uminus_class)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Linear_1Expression_1Variable_-initIDs (JNIEnv *env, jclass j_le_var_class)
- JNIEXPORT jboolean JNICALL Java_parma_1polyhedra_1library_Linear_1Expression_is_1zero (JNIEnv *env, jobject j_this)
- JNIEXPORT jboolean JNICALL Java_parma_1polyhedra_1library_Linear_1Expression_all_-1homogeneous_1terms_1are_1zero (JNIEnv *env, jobject j_this)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_MIP_1Problem_1Status_initIDs (JNIEnv *env, jclass j_mip_status_class)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_1Status_initIDs (JNIEnv *env, jclass j_mip_status_class)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Optimization_1Mode_initIDs (JNIEnv *env, jclass j_opt_mode_class)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Pair_initIDs (JNIEnv *env, jclass j_pair_class)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Poly_1Con_1Relation_initIDs (JNIEnv *env, jclass j_poly_con_relation_class)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Poly_1Gen_1Relation_initIDs (JNIEnv *env, jclass j_poly_gen_relation_class)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_PPL_1Object_initIDs (JNIEnv *env, jclass j_ppl_object_class)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Relation_1Symbol_initIDs (JNIEnv *env, jclass j_rel_sym_class)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Bounded_1Integer_1Type_-1Overflow_initIDs (JNIEnv *env, jclass j_bounded_overflow_class)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Bounded_1Integer_1Type_-1Representation_initIDs (JNIEnv *env, jclass j_bounded_rep_class)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Bounded_1Integer_1Type_-1Width_initIDs (JNIEnv *env, jclass j_bounded_width_class)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Variable_initIDs (JNIEnv *env, jclass j_variable_class)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Variables_1Set_initIDs (JNIEnv *env, jclass j_vset_class)
- JNIEXPORT jint JNICALL Java_parma_1polyhedra_1library_Parma_1Polyhedra_1Library_-version_1major (JNIEnv *, jclass)
- JNIEXPORT jint JNICALL Java_parma_1polyhedra_1library_Parma_1Polyhedra_1Library_-version_1minor (JNIEnv *, jclass)
- JNIEXPORT jint JNICALL Java_parma_1polyhedra_1library_Parma_1Polyhedra_1Library_-version_1revision (JNIEnv *, jclass)
- JNIEXPORT jint JNICALL Java_parma_1polyhedra_1library_Parma_1Polyhedra_1Library_-version_1beta (JNIEnv *, jclass)
- JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Parma_1Polyhedra_1Library_-version (JNIEnv *env, jclass)
- JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Parma_1Polyhedra_1Library_-banner (JNIEnv *env, jclass)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Parma_1Polyhedra_1Library_set_-1rounding_1for_1PPL (JNIEnv *env, jclass)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Parma_1Polyhedra_1Library_-restore_1pre_1PPL_1rounding (JNIEnv *env, jclass)

- JNIEXPORT jobject JNICALL Java_parma_1polyhedra_1library_MIP_1Problem_optimizing_-1point (JNIEnv *env, jobject j_this_mip_problem)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_MIP_1Problem_optimal_1value (JNIEnv *env, jobject j_this_mip_problem, jobject j_coeff_num, jobject j_coeff_den)
- JNIEXPORT jboolean JNICALL Java_parma_1polyhedra_1library_MIP_1Problem_OK (JNIEnv *env, jobject j_this_mip_problem)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_MIP_1Problem_build_1cpp_-1object__J (JNIEnv *env, jobject j_this_mip_problem, jlong j_dim)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_MIP_1Problem_build_1cpp_-1object__JLparma_1polyhedra_1library_Constraint_1System_2Lparma_1polyhedra_1library_-Linear_1Expression_2Lparma_1polyhedra_1library_Optimization_1Mode_2 (JNIEnv *env, jobject j_this_mip_problem, jlong j_dim, jobject j_cs, jobject j_le, jobject j_opt_mode)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_MIP_1Problem_build_1cpp_-1object__Lparma_1polyhedra_1library_MIP_1Problem_2 (JNIEnv *env, jobject j_this, jobject j_y)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_MIP_1Problem_free (JNIEnv *env, jobject j_this)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_MIP_1Problem_finalize (JNIEnv *env, jobject j_this)
- JNIEXPORT jlong JNICALL Java_parma_1polyhedra_1library_MIP_1Problem_total_1memory_-1in_1bytes (JNIEnv *env, jobject j_this_mip_problem)
- JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_MIP_1Problem_toString (JNIEnv *env, jobject j_this)
- JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_MIP_1Problem_ascii_1dump (JNIEnv *env, jobject j_this)
- JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Linear_1Expression_toString (JNIEnv *env, jobject j_this)
- JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Linear_1Expression_ascii_-1dump (JNIEnv *env, jobject j_this)
- JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Generator_toString (JNIEnv *env, jobject g)
- JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Generator_ascii_1dump (JNIEnv *env, jobject j_this)
- JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Constraint_toString (JNIEnv *env, jobject c)
- JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Constraint_ascii_1dump (JNIEnv *env, jobject j_this)
- JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Grid_1Generator_toString (JNIEnv *env, jobject g)
- JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Grid_1Generator_ascii_1dump (JNIEnv *env, jobject j_this)
- JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Congruence_toString (JNIEnv *env, jobject g)
- JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Congruence_ascii_1dump (JNIEnv *env, jobject j_this)
- JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Grid_1Generator_1System_-toString (JNIEnv *env, jobject ggs)
- JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Grid_1Generator_1System_-ascii_1dump (JNIEnv *env, jobject j_this)
- JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Generator_1System_toString (JNIEnv *env, jobject gs)

- JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Generator_1System_ascii_1dump (JNIEnv *env, jobject j_this)
- JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Constraint_1System_toString (JNIEnv *env, jobject cs)
- JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Constraint_1System_ascii_1dump (JNIEnv *env, jobject j_this)
- JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Congruence_1System_toString (JNIEnv *env, jobject cgs)
- JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Congruence_1System_ascii_1dump (JNIEnv *env, jobject j_this)
- JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_IO_wrap_1string (JNIEnv *env, jclass, jstring str, jint indent_depth, jint preferred_first_line_length, jint preferred_line_length)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_build_1cpp_1object__J (JNIEnv *env, jobject j_this_pip_problem, jlong j_dim)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_build_1cpp_1object__JLparma_1polyhedra_1library_Constraint_1System_2Lparma_1polyhedra_1library_Variables_1Set_2 (JNIEnv *env, jobject j_this_pip_problem, jlong j_dim, jobject j_cs, jobject j_vars)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_build_1cpp_1object__Lparma_1polyhedra_1library_PIP_1Problem_2 (JNIEnv *env, jobject j_this, jobject j_y)
- JNIEXPORT jboolean JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_OK (JNIEnv *env, jobject j_this_pip_problem)
- JNIEXPORT jlong JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_total_1memory_1in_1bytes (JNIEnv *env, jobject j_this_pip_problem)
- JNIEXPORT jlong JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_external_1memory_1in_1bytes (JNIEnv *env, jobject j_this_pip_problem)
- JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_toString (JNIEnv *env, jobject j_this)
- JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_ascii_1dump (JNIEnv *env, jobject j_this)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_free (JNIEnv *env, jobject j_this)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_PIP_1_problem_finalize (JNIEnv *env, jobject j_this)
- JNIEXPORT jlong JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_max_1space_1dimension (JNIEnv *env, jobject j_this_pip_problem)
- JNIEXPORT jlong JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_space_1dimension (JNIEnv *env, jobject j_this_pip_problem)
- JNIEXPORT jlong JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_get_1big_1parameter_1dimension (JNIEnv *env, jobject j_this_pip_problem)
- JNIEXPORT jobject JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_parameter_1space_1dimensions (JNIEnv *env, jobject j_this_pip_problem)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_set_1big_1parameter_1dimension (JNIEnv *env, jobject j_this_pip_problem, jlong j_dim)
- JNIEXPORT jlong JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_number_1of_1parameter_1space_1dimensions (JNIEnv *env, jobject j_this_pip_problem)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_add_1space_1dimensions_1and_1embed (JNIEnv *env, jobject j_this_pip_problem, jlong j_dim_vars, jlong j_dim_pars)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_add_1to_1parameter_1space_1dimensions (JNIEnv *env, jobject j_this_pip_problem, jobject j_vars)

- JNIEXPORT jlong JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_number_1of_-1constraints (JNIEnv *env, jobject j_this_pip_problem)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_add_1constraint (JNIEnv *env, jobject j_this_pip_problem, jobject j_c)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_add_1constraints (JNIEnv *env, jobject j_this_pip_problem, jobject j_cs)
- JNIEXPORT jboolean JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_is_1satisfiable (JNIEnv *env, jobject j_this_pip_problem)
- JNIEXPORT jobject JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_solve (JNIEnv *env, jobject j_this_pip_problem)
- JNIEXPORT jobject JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_solution (JNIEnv *env, jobject j_this_pip_problem)
- JNIEXPORT jobject JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_optimizing_-1solution (JNIEnv *env, jobject j_this_pip_problem)
- JNIEXPORT jobject JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_get_1pip_-1problem_1control_1parameter (JNIEnv *env, jobject j_this_pip_problem, jobject j_cpn)
- JNIEXPORT jobject JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_constraint_1at_-1index (JNIEnv *env, jobject j_this_pip_problem, jlong j_index)
- JNIEXPORT jobject JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_constraints (JNIEnv *env, jobject j_this_pip_problem)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_set_1pip_-1problem_1control_1parameter (JNIEnv *env, jobject j_this_pip_problem, jobject j_cpv)
- JNIEXPORT jboolean JNICALL Java_parma_1polyhedra_1library_PIP_1Tree_1Node_OK (JNIEnv *env, jobject j_this_pip_tree)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_PIP_1Tree_1Node_free (JNIEnv *env, jobject j_this)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_PIP_1Tree_1Node_finalize (JNIEnv *env, jobject j_this)
- JNIEXPORT jobject JNICALL Java_parma_1polyhedra_1library_PIP_1Tree_1Node_constraints (JNIEnv *env, jobject j_this_pip_node)
- JNIEXPORT jobject JNICALL Java_parma_1polyhedra_1library_PIP_1Tree_1Node_as_1solution (JNIEnv *env, jobject j_this)
- JNIEXPORT jobject JNICALL Java_parma_1polyhedra_1library_PIP_1Tree_1Node_as_1decision (JNIEnv *env, jobject j_this)
- JNIEXPORT jlong JNICALL Java_parma_1polyhedra_1library_PIP_1Tree_1Node_number_1of_-1artificials (JNIEnv *env, jobject j_this)
- JNIEXPORT jobject JNICALL Java_parma_1polyhedra_1library_PIP_1Tree_1Node_artificials (JNIEnv *env, jobject j_this_pip_node)
- JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_PIP_1Tree_1Node_toString (JNIEnv *env, jobject j_this)
- JNIEXPORT jobject JNICALL Java_parma_1polyhedra_1library_PIP_1Decision_1Node_child_-1node (JNIEnv *env, jobject j_this, jboolean j_branch)
- JNIEXPORT jobject JNICALL Java_parma_1polyhedra_1library_PIP_1Solution_1Node_-parametric_1values (JNIEnv *env, jobject j_this, jobject j_var)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Artificial_1Parameter_initIDs (JNIEnv *env, jclass j_artificial_parameter_class)
- JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Artificial_1Parameter_ascii_-1dump (JNIEnv *env, jobject j_this)
- JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Artificial_1Parameter_toString (JNIEnv *env, jobject j_this)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Artificial_1Parameter_1Sequence_-initIDs (JNIEnv *env, jclass j_aps_class)

- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Partial_1Function_build_1cpp_-1object (JNIEnv ∗env, jobject j_this_pfunc)
- JNIEXPORT jboolean JNICALL Java_parma_1polyhedra_1library_Partial_1Function_has_-1empty_1codomain (JNIEnv ∗env, jobject j_this_pfunc)
- JNIEXPORT jlong JNICALL Java_parma_1polyhedra_1library_Partial_1Function_max_1in_-1codomain (JNIEnv ∗env, jobject j_this_pfunc)
- JNIEXPORT jlong JNICALL Java_parma_1polyhedra_1library_Partial_1Function_maps (JNIEnv ∗env, jobject j_this_pfunc, jlong j_i)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Partial_1Function_insert (JNIEnv ∗env, jobject j_this_pfunc, jlong i, jlong j)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Partial_1Function_free (JNIEnv ∗env, jobject j_this)
- JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Partial_1Function_finalize (JNIEnv ∗env, jobject j_this)

12.56.1 Function Documentation

12.56.1.1 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_-Artificial_1Parameter_1Sequence_initIDs (JNIEnv ∗ *env*, jclass *j_aps_class*)

Definition at line 2112 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Artificial_Parameter_-Sequence_add_ID, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Artificial_-Parameter_Sequence_init_ID, Parma_Polyhedra_Library::Interfaces::Java::cached_FMIDs, and CHECK_RESULT_ASSERT.

12.56.1.2 JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Artificial_-1Parameter_ascii_1dump (JNIEnv ∗ *env*, jobject *j_this*)

Definition at line 2087 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_artificial_parameter(), and CATCH_-ALL.

12.56.1.3 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Artificial_-1Parameter_initIDs (JNIEnv ∗ *env*, jclass *j_artificial_parameter_class*)

Definition at line 2067 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Artificial_Parameter_-den_ID, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Artificial_Parameter_init_ID, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Artificial_Parameter_le_ID, Parma_-Polyhedra_Library::Interfaces::Java::cached_FMIDs, and CHECK_RESULT_ASSERT.

12.56.1.4 JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Artificial_-1Parameter_toString (JNIEnv * *env*, jobject *j_this*)

Definition at line 2101 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_artificial_parameter().

12.56.1.5 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_-Bounded_1Integer_1Type_1Overflow_initIDs (JNIEnv * *env*, jclass *j_bounded_overflow_class*)

Definition at line 607 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Bounded_-Integer_Type_Overflow_ordinal_ID, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Bounded_Integer_Type_Overflow_OVERFLOW_IMPOSSIBLE_ID, Parma_Polyhedra_-Library::Interfaces::Java::Java_FMID_Cache::Bounded_Integer_Type_Overflow_OVERFLOW_-UNDEFINED_ID, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Bounded_Integer_-Type_Overflow_OVERFLOW_WRAPS_ID, Parma_Polyhedra_Library::Interfaces::Java::cached_-FMIDs, and CHECK_RESULT_ASSERT.

12.56.1.6 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_-Bounded_1Integer_1Type_1Representation_initIDs (JNIEnv * *env*, jclass *j_bounded_rep_class*)

Definition at line 629 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Bounded_Integer_-Type_Representation_ordinal_ID, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Bounded_Integer_Type_Representation_SIGNED_2_COMPLEMENT_ID, Parma_Polyhedra_-Library::Interfaces::Java::Java_FMID_Cache::Bounded_Integer_Type_Representation_UNSIGNED_ID, Parma_Polyhedra_Library::Interfaces::Java::cached_FMIDs, and CHECK_RESULT_ASSERT.

12.56.1.7 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_-Bounded_1Integer_1Type_1Width_initIDs (JNIEnv * *env*, jclass *j_bounded_width_class*)

Definition at line 647 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Bounded_Integer_Type_-Width_BITS_128_ID, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Bounded_-Integer_Type_Width_BITS_16_ID, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Bounded_Integer_Type_Width_BITS_32_ID, Parma_Polyhedra_Library::Interfaces::Java::Java_-FMID_Cache::Bounded_Integer_Type_Width_BITS_64_ID, Parma_Polyhedra_-Library::Interfaces::Java::Java_FMID_Cache::Bounded_Integer_Type_Width_BITS_8_ID, Parma_-Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Bounded_Integer_Type_Width_ordinal_ID, Parma_Polyhedra_Library::Interfaces::Java::cached_FMIDs, and CHECK_RESULT_ASSERT.

12.56.1.8 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_By_1Reference_initIDs (JNIEnv * *env*, jclass *j_by_ref_class*)

Definition at line 90 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::By_Reference_init_ID, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::By_Reference_obj_ID, Parma_Polyhedra_Library::Interfaces::Java::cached_FMIDs, and CHECK_RESULT_ASSERT.

12.56.1.9 JNIEXPORT jint JNICALL Java_parma_1polyhedra_1library_Coefficient_bits (JNIEnv *, jclass)

Definition at line 102 of file ppl_java_globals.cc.

12.56.1.10 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Coefficient_initIDs (JNIEnv * *env*, jclass *j_coeff_class*)

Definition at line 108 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Boolean, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Boolean_valueOf_ID, Parma_Polyhedra_Library::Interfaces::Java::cached_classes, Parma_Polyhedra_Library::Interfaces::Java::cached_FMIDs, CHECK_RESULT_ASSERT, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Coefficient_init_from_String_ID, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Coefficient_toString_ID, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Coefficient_value_ID, Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Integer, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Integer_intValue_ID, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Integer_valueOf_ID, Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Long, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Long_longValue_ID, and Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Long_valueOf_ID.

12.56.1.11 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Complexity_1Class_initIDs (JNIEnv * *env*, jclass *j_complexity_class*)

Definition at line 145 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::cached_FMIDs, CHECK_RESULT_ASSERT, and Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Complexity_Class_ordinal_ID.

12.56.1.12 JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Congruence_1System_ascii_1dump (JNIEnv * *env*, jobject *j_this*)

Definition at line 1465 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_congruence_system(), and CATCH_-ALL.

12.56.1.13 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Congruence_-1System_initIDs (JNIEnv * *env*, jclass *j_con_sys_class*)

Definition at line 178 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::cached_FMIDs, CHECK_RESULT_ASSERT, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Congruence_System_add_ID, and Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Congruence_System_init_ID.

12.56.1.14 JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Congruence_-1System_toString (JNIEnv * *env*, jobject *cgs*)

Definition at line 1455 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_congruence_system().

12.56.1.15 JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Congruence_ascii_-1dump (JNIEnv * *env*, jobject *j_this*)

Definition at line 1372 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_congruence(), and CATCH_ALL.

12.56.1.16 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Congruence_initIDs (JNIEnv * *env*, jclass *j_congruence_class*)

Definition at line 153 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::cached_FMIDs, CHECK_RESULT_-ASSERT, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Congruence_init_ID, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Congruence_lhs_ID, Parma_-Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Congruence_mod_ID, and Parma_Polyhedra_-Library::Interfaces::Java::Java_FMID_Cache::Congruence_rhs_ID.

12.56.1.17 JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Congruence_-toString (JNIEnv * *env*, jobject *g*)

Definition at line 1362 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_congruence().

12.56.1.18 JNIEXPORT jstring JNICALL Java_parma_1polyhedra_-1library_Constraint_1System_ascii_1dump (JNIEnv * *env*, jobject *j_this*)

Definition at line 1442 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_constraint_system(), and CATCH_-ALL.

12.56.1.19 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Constraint_-1System_initIDs (JNIEnv * *env*, jclass *j_con_sys_class*)

Definition at line 215 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::cached_classes, Parma_Polyhedra_-Library::Interfaces::Java::cached_FMIDs, CHECK_RESULT_ASSERT, Parma_Polyhedra_-Library::Interfaces::Java::Java_FMID_Cache::Constraint_System_add_ID, Parma_Polyhedra_-Library::Interfaces::Java::Java_FMID_Cache::Constraint_System_init_ID, Parma_Polyhedra_-Library::Interfaces::Java::Java_Class_Cache::Iterator, Parma_Polyhedra_Library::Interfaces::Java::Java_-FMID_Cache::System_Iterator_has_next_ID, Parma_Polyhedra_Library::Interfaces::Java::Java_-FMID_Cache::System_iterator_ID, and Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::System_Iterator_next_ID.

12.56.1.20 JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Constraint_-1System_toString (JNIEnv * *env*, jobject *cs*)

Definition at line 1431 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_constraint_system().

12.56.1.21 JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Constraint_ascii_-1dump (JNIEnv * *env*, jobject *j_this*)

Definition at line 1326 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_constraint(), and CATCH_ALL.

12.56.1.22 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Constraint_initIDs (JNIEnv * *env*, jclass *j_constraint_class*)

Definition at line 190 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::cached_FMIDs, CHECK_RESULT_-ASSERT, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Constraint_init_ID, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Constraint_kind_ID, Parma_-

Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Constraint_lhs_ID, and Parma_Polyhedra_-Library::Interfaces::Java::Java_FMID_Cache::Constraint_rhs_ID.

12.56.1.23 JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Constraint_-toString (JNIEnv * *env*, jobject *c*)

Definition at line 1316 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_constraint().

12.56.1.24 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Degenerate_-1Element_initIDs (JNIEnv * *env*, jclass *j_degenerate_class*)

Definition at line 240 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::cached_FMIDs, CHECK_RESULT_ASSERT, and Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Degenerate_Element_ordinal_ID.

12.56.1.25 JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Generator_-1System_ascii_1dump (JNIEnv * *env*, jobject *j_this*)

Definition at line 1418 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_generator_system(), and CATCH_-ALL.

12.56.1.26 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Generator_1System_-initIDs (JNIEnv * *env*, jclass *j_gen_sys_class*)

Definition at line 289 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::cached_FMIDs, CHECK_RESULT_ASSERT, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Generator_System_add_ID, and Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Generator_System_init_ID.

12.56.1.27 JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Generator_-1System_toString (JNIEnv * *env*, jobject *gs*)

Definition at line 1408 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_generator_system().

12.56.1.28 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Generator_1Type_-initIDs (JNIEnv * *env*, jclass *j_gen_type_class*)

Definition at line 301 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::cached_FMIDs, CHECK_RESULT_ASSERT, and Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Generator_Type_ordinal_ID.

12.56.1.29 JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Generator_ascii_-1dump (JNIEnv * *env*, jobject *j_this*)

Definition at line 1303 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_generator(), and CATCH_ALL.

12.56.1.30 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Generator_initIDs (JNIEnv * *env*, jclass *j_generator_class*)

Definition at line 248 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::cached_FMIDs, CHECK_RESULT_-ASSERT, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Generator_closure_-point_ID, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Generator_-div_ID, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Generator_gt_ID, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Generator_le_ID, Parma_-Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Generator_line_ID, Parma_Polyhedra_-Library::Interfaces::Java::Java_FMID_Cache::Generator_point_ID, and Parma_Polyhedra_-Library::Interfaces::Java::Java_FMID_Cache::Generator_ray_ID.

12.56.1.31 JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Generator_toString (JNIEnv * *env*, jobject *g*)

Definition at line 1293 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_generator().

12.56.1.32 JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_-Grid_1Generator_1System_ascii_1dump (JNIEnv * *env*, jobject *j_this*)

Definition at line 1395 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_grid_generator_system(), and CATCH_ALL.

12.56.1.33 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Grid_1Generator_-1System_initIDs (JNIEnv * *env*, jclass *j_gen_sys_class*)

Definition at line 346 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::cached_FMIDs, CHECK_RESULT_ASSERT, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Grid_Generator_System_add_ID, and Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Grid_Generator_System_init_ID.

12.56.1.34 JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_-Grid_1Generator_1System_toString (JNIEnv * *env*, jobject *ggs*)

Definition at line 1385 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_grid_generator_system().

12.56.1.35 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Grid_1Generator_-1Type_initIDs (JNIEnv * *env*, jclass *j_grid_gen_type_class*)

Definition at line 358 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::cached_FMIDs, CHECK_RESULT_ASSERT, and Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Grid_Generator_Type_ordinal_ID.

12.56.1.36 JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Grid_1Generator_-ascii_1dump (JNIEnv * *env*, jobject *j_this*)

Definition at line 1349 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_grid_generator(), and CATCH_ALL.

12.56.1.37 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Grid_1Generator_-initIDs (JNIEnv * *env*, jclass *j_grid_generator_class*)

Definition at line 310 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::cached_FMIDs, CHECK_RESULT_ASSERT, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Grid_Generator_div_ID, Parma_-Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Grid_Generator_grid_line_ID, Parma_-Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Grid_Generator_grid_point_ID, Parma_-Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Grid_Generator_gt_ID, Parma_Polyhedra_-Library::Interfaces::Java::Java_FMID_Cache::Grid_Generator_le_ID, and Parma_Polyhedra_-Library::Interfaces::Java::Java_FMID_Cache::Grid_Generator_parameter_ID.

12.56.1.38 JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Grid_1Generator_-toString (JNIEnv * *env*, jobject *g*)

Definition at line 1339 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_grid_generator().

12.56.1.39 JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_IO_wrap_1string (JNIEnv * *env*, jclass, jstring *str*, jint *indent_depth*, jint *preferred_first_line_length*, jint *preferred_line_length*)

Definition at line 1478 of file ppl_java_globals.cc.

References CATCH_ALL, and CHECK_RESULT_RETURN.

12.56.1.40 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_-Linear_1Expression_1Coefficient_initIDs (JNIEnv * *env*, jclass *j_le_coeff_class*)

Definition at line 383 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::cached_FMIDs, CHECK_RESULT_ASSERT, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Linear_Expression_Coefficient_-coeff_ID, and Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Linear_Expression_-Coefficient_init_ID.

12.56.1.41 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_-Linear_1Expression_1Difference_initIDs (JNIEnv * *env*, jclass *j_le_diff_class*)

Definition at line 398 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::cached_FMIDs, CHECK_RESULT_ASSERT, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Linear_Expression_Difference_lhs_ID, and Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Linear_Expression_Difference_-rhs_ID.

12.56.1.42 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Linear_1Expression_-1Sum_initIDs (JNIEnv * *env*, jclass *j_le_sum_class*)

Definition at line 412 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::cached_FMIDs, CHECK_RESULT_ASSERT, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Linear_Expression_Sum_lhs_ID, and Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Linear_Expression_Sum_rhs_ID.

12.56.1.43 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_-Linear_1Expression_1Times_initIDs (JNIEnv * *env*, jclass *j_le_times_class*)

Definition at line 426 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::cached_FMIDs, CHECK_RESULT_ASSERT, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Linear_Expression_Times_coeff_ID, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Linear_Expression_Times_init_from_-coeff_var_ID, and Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Linear_Expression_-Times_lin_expr_ID.

12.56.1.44 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_-Linear_1Expression_1Unary_1Minus_initIDs (JNIEnv * *env*, jclass *j_le_uminus_class*)

Definition at line 446 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::cached_FMIDs, CHECK_RESULT_ASSERT, and Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Linear_Expression_Unary_Minus_-arg_ID.

12.56.1.45 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_-Linear_1Expression_1Variable_initIDs (JNIEnv * *env*, jclass *j_le_var_class*)

Definition at line 456 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::cached_FMIDs, CHECK_RESULT_ASSERT, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Linear_Expression_Variable_init_ID, and Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Linear_Expression_Variable_var_-id_ID.

12.56.1.46 JNIEXPORT jboolean JNICALL Java_parma_1polyhedra_1library_Linear_-1Expression_all_1homogeneous_1terms_1are_1zero (JNIEnv * *env*, jobject *j_this*)

Definition at line 479 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_linear_expression(), and CATCH_-ALL.

12.56.1.47 JNIEXPORT jstring JNICALL Java_parma_1polyhedra_-1library_Linear_1Expression_ascii_1dump (JNIEnv * *env*, jobject *j_this*)

Definition at line 1280 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_linear_expression(), and CATCH_-ALL.

12.56.1.48 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Linear_1Expression_-initIDs (JNIEnv * *env*, jclass *j_le_class*)

Definition at line 367 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::cached_FMIDs, CHECK_RESULT_ASSERT, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Linear_Expression_sum_ID, and Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Linear_Expression_times_ID.

12.56.1.49 JNIEXPORT jboolean JNICALL Java_parma_1polyhedra_-1library_Linear_1Expression_is_1zero (JNIEnv * *env*, jobject *j_this*)

Definition at line 469 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_linear_expression(), and CATCH_-ALL.

12.56.1.50 JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Linear_-1Expression_toString (JNIEnv * *env*, jobject *j_this*)

Definition at line 1270 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_linear_expression().

12.56.1.51 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_MIP_1Problem_-1Status_initIDs (JNIEnv * *env*, jclass *j_mip_status_class*)

Definition at line 489 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::cached_FMIDs, CHECK_RESULT_ASSERT, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::MIP_Problem_Status_OPTIMIZED_-MIP_PROBLEM_ID, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::MIP_Problem_-Status_ordinal_ID, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::MIP_Problem_-Status_UNBOUNDED_MIP_PROBLEM_ID, and Parma_Polyhedra_Library::Interfaces::Java::Java_-FMID_Cache::MIP_Problem_Status_UNFEASIBLE_MIP_PROBLEM_ID.

12.56.1.52 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_MIP_-1Problem_add_1constraint (JNIEnv * *env*, jobject *j_this_mip_problem*, jobject *j_c*)

Definition at line 1034 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_constraint(), CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.53 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_MIP_-1Problem_add_1constraints (JNIEnv * *env*, jobject *j_this_mip_problem*, jobject *j_cs*)

Definition at line 1046 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_constraint_system(), CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.54 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_MIP_1Problem_-add_1space_1dimensions_1and_1embed (JNIEnv * *env*, jobject *j_this_mip_problem*, jlong *j_dim*)

Definition at line 1010 of file ppl_java_globals.cc.

References CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.55 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_MIP_1Problem_-add_1to_1integer_1space_1dimensions (JNIEnv * *env*, jobject *j_this_mip_problem*, jobject *j_vset*)

Definition at line 1022 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_variables_set(), CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.56 JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_MIP_1Problem_-ascii_1dump (JNIEnv * *env*, jobject *j_this*)

Definition at line 1256 of file ppl_java_globals.cc.

References CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.57 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_MIP_1Problem_-build_1cpp_1object__J (JNIEnv * *env*, jobject *j_this_mip_problem*, jlong *j_dim*)

Definition at line 1179 of file ppl_java_globals.cc.

References CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::set_ptr().

12.56.1.58 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_MIP_1Problem_-build_1cpp_1object__JLparma_1polyhedra_1library_Constraint_1System_2Lparma_-1polyhedra_1library_Linear_1Expression_2Lparma_1polyhedra_1library_-Optimization_1Mode_2 (JNIEnv * *env*, jobject *j_this_mip_problem*, jlong *j_dim*, jobject *j_cs*, jobject *j_le*, jobject *j_opt_mode*)

Definition at line 1190 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_constraint_system(), Parma_-Polyhedra_Library::Interfaces::Java::build_cxx_linear_expression(), Parma_Polyhedra_-Library::Interfaces::Java::build_cxx_optimization_mode(), CATCH_ALL, and Parma_Polyhedra_-Library::Interfaces::Java::set_ptr().

12.56.1.59 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_MIP_1Problem_-build_1cpp_1object__Lparma_1polyhedra_1library_MIP_1Problem_2 (JNIEnv * *env*, jobject *j_this*, jobject *j_y*)

Definition at line 1205 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::get_ptr(), and Parma_Polyhedra_-Library::Interfaces::Java::set_ptr().

12.56.1.60 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_MIP_1Problem_clear (JNIEnv * *env*, jobject *j_this_mip_problem*)

Definition at line 999 of file ppl_java_globals.cc.

References CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.61 JNIEXPORT jobject JNICALL Java_parma_1polyhedra_1library_MIP_1Problem_-constraints (JNIEnv * *env*, jobject *j_this_mip_problem*)

Definition at line 974 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_java_constraint(), Parma_Polyhedra_-Library::Interfaces::Java::cached_classes, Parma_Polyhedra_Library::Interfaces::Java::cached_-FMIDs, CATCH_ALL, CHECK_EXCEPTION_RETURN, CHECK_RESULT_RETURN, Parma_-Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Constraint_System, Parma_Polyhedra_-Library::Interfaces::Java::Java_FMID_Cache::Constraint_System_add_ID, Parma_Polyhedra_-Library::Interfaces::Java::Java_FMID_Cache::Constraint_System_init_ID, and Parma_Polyhedra_-Library::Interfaces::Java::get_ptr().

12.56.1.62 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_MIP_1Problem_-evaluate_1objective_1function (JNIEnv * *env*, jobject *j_this_mip_problem*, jobject *j_gen*, jobject *j_coeff_num*, jobject *j_coeff_den*)

Definition at line 1106 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_generator(), Parma_Polyhedra_Library::Interfaces::Java::build_java_coeff(), CATCH_ALL, Parma_Polyhedra_Library::Interfaces::Java::get_ptr(), and Parma_Polyhedra_Library::Interfaces::Java::set_coefficient().

12.56.1.63 JNIEXPORT jobject JNICALL Java_parma_1polyhedra_1library_MIP_1Problem_feasible_1point (JNIEnv * *env*, jobject *j_this_mip_problem*)

Definition at line 1123 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_java_generator(), CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.64 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_MIP_1Problem_finalize (JNIEnv * *env*, jobject *j_this*)

Definition at line 1225 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::get_ptr(), and Parma_Polyhedra_Library::Interfaces::Java::is_java_marked().

12.56.1.65 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_MIP_1Problem_free (JNIEnv * *env*, jobject *j_this*)

Definition at line 1214 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::get_ptr(), Parma_Polyhedra_Library::Interfaces::Java::is_java_marked(), and Parma_Polyhedra_Library::Interfaces::Java::set_ptr().

12.56.1.66 JNIEXPORT jobject JNICALL Java_parma_1polyhedra_1library_MIP_1Problem_get_1control_1parameter (JNIEnv * *env*, jobject *j_this_mip_problem*, jobject *j_cpn*)

Definition at line 942 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_control_parameter_name(), Parma_Polyhedra_Library::Interfaces::Java::build_java_control_parameter_value(), CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.67 JNIEXPORT jobject JNICALL Java_parma_1polyhedra_1library_MIP_1Problem_integer_1space_1dimensions (JNIEnv * *env*, jobject *j_this_mip_problem*)

Definition at line 890 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_java_variables_set(), CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.68 JNIEXPORT jboolean JNICALL Java_parma_1polyhedra_-1library_MIP_1Problem_is_1satisfiable (JNIEnv * *env*, jobject *j_this_mip_problem*)

Definition at line 1082 of file ppl_java_globals.cc.

References CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.69 JNIEXPORT jlong JNICALL Java_parma_1polyhedra_1library_-MIP_1Problem_max_1space_1dimension (JNIEnv * *env*, jobject *j_this_mip_problem*)

Definition at line 866 of file ppl_java_globals.cc.

References CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.70 JNIEXPORT jobject JNICALL Java_parma_1polyhedra_1library_-MIP_1Problem_objective_1function (JNIEnv * *env*, jobject *j_this_mip_problem*)

Definition at line 903 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_java_coeff(), Parma_-Polyhedra_Library::Interfaces::Java::build_linear_expression(), Parma_Polyhedra_-Library::Interfaces::Java::cached_classes, Parma_Polyhedra_Library::Interfaces::Java::cached_FMIDs, CATCH_ALL, CHECK_RESULT_RETURN, Parma_Polyhedra_Library::Interfaces::Java::get_ptr(), Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Linear_Expression_Coefficient, Parma_-Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Linear_Expression_Coefficient_init_ID, and Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Linear_Expression_sum_ID.

12.56.1.71 JNIEXPORT jboolean JNICALL Java_parma_1polyhedra_1library_MIP_1Problem_-OK (JNIEnv * *env*, jobject *j_this_mip_problem*)

Definition at line 1167 of file ppl_java_globals.cc.

References CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.72 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_MIP_1Problem_-optimal_1value (JNIEnv * *env*, jobject *j_this_mip_problem*, jobject *j_coeff_num*, jobject *j_coeff_den*)

Definition at line 1151 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_java_coeff(), CATCH_ALL, Parma_Polyhedra_Library::Interfaces::Java::get_ptr(), and Parma_Polyhedra_Library::Interfaces::Java::set_coefficient().

12.56.1.73 JNIEXPORT jobject JNICALL Java_parma_1polyhedra_1library_MIP_1Problem_optimization_1mode (JNIEnv * *env*, jobject *j_this_mip_problem*)

Definition at line 929 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_java_optimization_mode(), CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.74 JNIEXPORT jobject JNICALL Java_parma_1polyhedra_1library_MIP_1Problem_optimizing_1point (JNIEnv * *env*, jobject *j_this_mip_problem*)

Definition at line 1137 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_java_generator(), CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.75 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_MIP_1Problem_set_1control_1parameter (JNIEnv * *env*, jobject *j_this_mip_problem*, jobject *j_cpv*)

Definition at line 960 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_control_parameter_value(), CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.76 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_MIP_1Problem_set_1objective_1function (JNIEnv * *env*, jobject *j_this_mip_problem*, jobject *j_le*)

Definition at line 1058 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_linear_expression(), CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.77 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_MIP_1Problem_set_1optimization_1mode (JNIEnv * *env*, jobject *j_this_mip_problem*, jobject *j_opt_mode*)

Definition at line 1070 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_optimization_mode(), CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.78 JNIEXPORT jobject JNICALL Java_parma_1polyhedra_1library_MIP_1Problem_-solve (JNIEnv * *env*, jobject *j_this_mip_problem*)

Definition at line 1093 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_java_mip_status(), CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.79 JNIEXPORT jlong JNICALL Java_parma_1polyhedra_1library_MIP_1Problem_-space_1dimension (JNIEnv * *env*, jobject *j_this_mip_problem*)

Definition at line 878 of file ppl_java_globals.cc.

References CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.80 JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_MIP_1Problem_-toString (JNIEnv * *env*, jobject *j_this*)

Definition at line 1245 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.81 JNIEXPORT jlong JNICALL Java_parma_1polyhedra_1library_-MIP_1Problem_total_1memory_1in_1bytes (JNIEnv * *env*, jobject *j_this_mip_problem*)

Definition at line 1233 of file ppl_java_globals.cc.

References CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.82 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Optimization_-1Mode_initIDs (JNIEnv * *env*, jclass *j_opt_mode_class*)

Definition at line 529 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::cached_FMIDs, CHECK_RESULT_-ASSERT, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Optimization_Mode_-MAXIMIZATION_ID, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Optimization_-Mode_MINIMIZATION_ID, and Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_-Cache::Optimization_Mode_ordinal_ID.

12.56.1.83 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Pair_initIDs (JNIEnv * *env*, jclass *j_pair_class*)

Definition at line 547 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::cached_FMIDs, CHECK_RESULT_ASSERT, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Pair_first_ID, and Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Pair_second_ID.

12.56.1.84 JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Parma_1Polyhedra_1Library_banner (JNIEnv * *env*, jclass)

Definition at line 743 of file ppl_java_globals.cc.

12.56.1.85 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Parma_1Polyhedra_1Library_finalize_1library (JNIEnv * *env*, jclass)

Definition at line 82 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::cached_classes, and Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::clear_cache().

12.56.1.86 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Parma_1Polyhedra_1Library_initialize_1library (JNIEnv * *env*, jclass)

Definition at line 75 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::cached_classes, and Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::init_cache().

12.56.1.87 JNIEXPORT jint JNICALL Java_parma_1polyhedra_1library_Parma_1Polyhedra_1Library_irrational_1precision (JNIEnv * *env*, jclass)

Definition at line 767 of file ppl_java_globals.cc.

References CATCH_ALL.

12.56.1.88 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Parma_1Polyhedra_1Library_reset_1deterministic_1timeout (JNIEnv * *env*, jclass)

Definition at line 850 of file ppl_java_globals.cc.

References CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::reset_deterministic_timeout().

12.56.1.89 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_-Parma_1Polyhedra_1Library_reset_1timeout (JNIEnv * *env*, jclass)

Definition at line 811 of file ppl_java_globals.cc.

References CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::reset_timeout().

12.56.1.90 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Parma_-1Polyhedra_1Library_restore_1pre_1PPL_1rounding (JNIEnv * *env*, jclass)

Definition at line 758 of file ppl_java_globals.cc.

References CATCH_ALL.

12.56.1.91 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Parma_-1Polyhedra_1Library_set_1deterministic_1timeout (JNIEnv * *env*, jclass, jint *weight*)

Definition at line 827 of file ppl_java_globals.cc.

References CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::reset_deterministic_timeout().

12.56.1.92 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Parma_-1Polyhedra_1Library_set_1irrational_1precision (JNIEnv * *env*, jclass, jint *p*)

Definition at line 777 of file ppl_java_globals.cc.

References CATCH_ALL.

12.56.1.93 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_-Parma_1Polyhedra_1Library_set_1rounding_1for_1PPL (JNIEnv * *env*, jclass)

Definition at line 749 of file ppl_java_globals.cc.

References CATCH_ALL.

12.56.1.94 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_-Parma_1Polyhedra_1Library_set_1timeout (JNIEnv * *env*, jclass, jint *hsecs*)

Definition at line 787 of file ppl_java_globals.cc.

References CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::reset_timeout().

12.56.1.95 JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_Parma_-1Polyhedra_1Library_version (JNIEnv * *env*, jclass)

Definition at line 737 of file ppl_java_globals.cc.

12.56.1.96 JNIEXPORT jint JNICALL Java_parma_1polyhedra_1library_Parma_1Polyhedra_-1Library_version_1beta (JNIEnv *, jclass)

Definition at line 731 of file ppl_java_globals.cc.

12.56.1.97 JNIEXPORT jint JNICALL Java_parma_1polyhedra_1library_Parma_1Polyhedra_-1Library_version_1major (JNIEnv *, jclass)

Definition at line 713 of file ppl_java_globals.cc.

12.56.1.98 JNIEXPORT jint JNICALL Java_parma_1polyhedra_1library_Parma_1Polyhedra_-1Library_version_1minor (JNIEnv *, jclass)

Definition at line 719 of file ppl_java_globals.cc.

12.56.1.99 JNIEXPORT jint JNICALL Java_parma_1polyhedra_1library_Parma_1Polyhedra_-1Library_version_1revision (JNIEnv *, jclass)

Definition at line 725 of file ppl_java_globals.cc.

12.56.1.100 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_-Partial_1Function_build_1cpp_1object (JNIEnv * *env*, jobject *j_this_pfunc*)

Definition at line 2125 of file ppl_java_globals.cc.

References CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::set_ptr().

12.56.1.101 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Partial_1Function_-finalize (JNIEnv * *env*, jobject *j_this*)

Definition at line 2195 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::get_ptr(), and Parma_Polyhedra_-Library::Interfaces::Java::is_java_marked().

12.56.1.102 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Partial_1Function_-free (JNIEnv * *env*, jobject *j_this*)

Definition at line 2183 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::get_ptr(), Parma_Polyhedra_-Library::Interfaces::Java::is_java_marked(), and Parma_Polyhedra_Library::Interfaces::Java::set_ptr().

12.56.1.103 JNIEXPORT jboolean JNICALL Java_parma_1polyhedra_1library_-Partial_1Function_has_1empty_1codomain (JNIEnv * *env*, jobject *j_this_pfunc*)

Definition at line 2135 of file ppl_java_globals.cc.

References CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.104 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Partial_-1Function_insert (JNIEnv * *env*, jobject *j_this_pfunc*, jlong *i*, jlong *j*)

Definition at line 2172 of file ppl_java_globals.cc.

References CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.105 JNIEXPORT jlong JNICALL Java_parma_1polyhedra_1library_-Partial_1Function_maps (JNIEnv * *env*, jobject *j_this_pfunc*, jlong *j_i*)

Definition at line 2159 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.106 JNIEXPORT jlong JNICALL Java_parma_1polyhedra_1library_-Partial_1Function_max_1in_1codomain (JNIEnv * *env*, jobject *j_this_pfunc*)

Definition at line 2147 of file ppl_java_globals.cc.

References CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.107 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_PIP_1_problem_-finalize (JNIEnv * *env*, jobject *j_this*)

Definition at line 1605 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::get_ptr(), and Parma_Polyhedra_-Library::Interfaces::Java::is_java_marked().

12.56.1.108 JNIEXPORT jobject JNICALL Java_parma_1polyhedra_1library_PIP_-1Decision_1Node_child_1node (JNIEnv * *env*, jobject *j_this*, jboolean *j_branch*)

Definition at line 2033 of file ppl_java_globals.cc.

References CATCH_ALL, CHECK_RESULT_ASSERT, CHECK_RESULT_RETURN, Parma_-Polyhedra_Library::Interfaces::Java::get_ptr(), and Parma_Polyhedra_Library::Interfaces::Java::set_ptr().

12.56.1.109 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_-1Status_initIDs (JNIEnv * *env*, jclass *j_mip_status_class*)

Definition at line 511 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::cached_FMIDs, CHECK_RESULT_ASSERT, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::PIP_Problem_Status_OPTIMIZED_-PIP_PROBLEM_ID, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::PIP_Problem_-Status_ordinal_ID, and Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::PIP_Problem_-Status_UNFEASIBLE_PIP_PROBLEM_ID.

12.56.1.110 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_PIP_-1Problem_add_1constraint (JNIEnv * *env*, jobject *j_this_pip_problem*, jobject *j_c*)

Definition at line 1723 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_constraint(), CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.111 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_PIP_-1Problem_add_1constraints (JNIEnv * *env*, jobject *j_this_pip_problem*, jobject *j_cs*)

Definition at line 1735 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_constraint_system(), CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.112 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_-add_1space_1dimensions_1and_1embed (JNIEnv * *env*, jobject *j_this_pip_problem*, jlong *j_dim_vars*, jlong *j_dim_pars*)

Definition at line 1686 of file ppl_java_globals.cc.

References CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.113 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_-add_1to_1parameter_1space_1dimensions (JNIEnv * *env*, jobject *j_this_pip_problem*, jobject *j_vars*)

Definition at line 1699 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_variables_set(), CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.114 JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_-ascii_1dump (JNIEnv * *env*, jobject *j_this*)

Definition at line 1580 of file ppl_java_globals.cc.

References CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.115 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_-build_1cpp_1object__J (JNIEnv * *env*, jobject *j_this_pip_problem*, jlong *j_dim*)

Definition at line 1498 of file ppl_java_globals.cc.

References CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::set_ptr().

12.56.1.116 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_-build_1cpp_1object__JLparma_1polyhedra_1library_Constraint_1System_-2Lparma_1polyhedra_1library_Variables_1Set_2 (JNIEnv * *env*, jobject *j_this_pip_problem*, jlong *j_dim*, jobject *j_cs*, jobject *j_vars*)

Definition at line 1509 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_constraint_system(), Parma_Polyhedra_Library::Interfaces::Java::build_cxx_variables_set(), CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::set_ptr().

12.56.1.117 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_build_1cpp_1object__Lparma_1polyhedra_1library_PIP_1Problem_2 (JNIEnv * *env*, jobject *j_this*, jobject *j_y*)

Definition at line 1524 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::get_ptr(), and Parma_Polyhedra_Library::Interfaces::Java::set_ptr().

12.56.1.118 JNIEXPORT jobject JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_constraint_1at_1index (JNIEnv * *env*, jobject *j_this_pip_problem*, jlong *j_index*)

Definition at line 1832 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_java_constraint(), CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.119 JNIEXPORT jobject JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_constraints (JNIEnv * *env*, jobject *j_this_pip_problem*)

Definition at line 1846 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_java_constraint(), Parma_Polyhedra_Library::Interfaces::Java::cached_classes, Parma_Polyhedra_Library::Interfaces::Java::cached_FMIDs, CATCH_ALL, CHECK_EXCEPTION_RETURN, CHECK_RESULT_RETURN, Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Constraint_System, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Constraint_System_add_ID, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Constraint_System_init_ID, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.120 JNIEXPORT jlong JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_external_1memory_1in_1bytes (JNIEnv * *env*, jobject *j_this_pip_problem*)

Definition at line 1557 of file ppl_java_globals.cc.

References CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.121 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_free (JNIEnv * *env*, jobject *j_this*)

Definition at line 1594 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::get_ptr(), Parma_Polyhedra_Library::Interfaces::Java::is_java_marked(), and Parma_Polyhedra_Library::Interfaces::Java::set_ptr().

12.56.1.122 JNIEXPORT jlong JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_get_1big_1parameter_1dimension (JNIEnv * *env*, jobject *j_this_pip_problem*)

Definition at line 1637 of file ppl_java_globals.cc.

References CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.123 JNIEXPORT jobject JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_get_1pip_1problem_1control_1parameter (JNIEnv * *env*, jobject *j_this_pip_problem*, jobject *j_cpn*)

Definition at line 1814 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_pip_problem_control_parameter_name(), Parma_Polyhedra_Library::Interfaces::Java::build_java_pip_problem_control_parameter_value(), CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.124 JNIEXPORT jboolean JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_is_1satisfiable (JNIEnv * *env*, jobject *j_this_pip_problem*)

Definition at line 1747 of file ppl_java_globals.cc.

References CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.125 JNIEXPORT jlong JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_max_1space_1dimension (JNIEnv * *env*, jobject *j_this_pip_problem*)

Definition at line 1613 of file ppl_java_globals.cc.

References CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.126 JNIEXPORT jlong JNICALL Java_parma_1polyhedra_1library_-PIP_1Problem_number_1of_1constraints (JNIEnv * *env*, jobject *j_this_pip_problem*)

Definition at line 1711 of file ppl_java_globals.cc.

References CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.127 JNIEXPORT jlong JNICALL Java_parma_1polyhedra_1library_PIP_-1Problem_number_1of_1parameter_1space_1dimensions (JNIEnv * *env*, jobject *j_this_pip_problem*)

Definition at line 1674 of file ppl_java_globals.cc.

References CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.128 JNIEXPORT jboolean JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_-OK (JNIEnv * *env*, jobject *j_this_pip_problem*)

Definition at line 1533 of file ppl_java_globals.cc.

References CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.129 JNIEXPORT jobject JNICALL Java_parma_1polyhedra_1library_-PIP_1Problem_optimizing_1solution (JNIEnv * *env*, jobject *j_this_pip_problem*)

Definition at line 1792 of file ppl_java_globals.cc.

References CATCH_ALL, CHECK_RESULT_ASSERT, CHECK_RESULT_RETURN, Parma_-Polyhedra_Library::Interfaces::Java::get_ptr(), and Parma_Polyhedra_Library::Interfaces::Java::set_ptr().

12.56.1.130 JNIEXPORT jobject JNICALL Java_parma_1polyhedra_1library_-PIP_1Problem_parameter_1space_1dimensions (JNIEnv * *env*, jobject *j_this_pip_problem*)

Definition at line 1649 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_java_variables_set(), CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.131 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_-set_1big_1parameter_1dimension (JNIEnv * *env*, jobject *j_this_pip_problem*, jlong *j_dim*)

Definition at line 1662 of file ppl_java_globals.cc.

References CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.132 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_set_-1pip_1problem_1control_1parameter (JNIEnv * *env*, jobject *j_this_pip_problem*, jobject *j_cpv*)

Definition at line 1871 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_pip_problem_control_parameter_-value(), CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.133 JNIEXPORT jobject JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_-solution (JNIEnv * *env*, jobject *j_this_pip_problem*)

Definition at line 1770 of file ppl_java_globals.cc.

References CATCH_ALL, CHECK_RESULT_ASSERT, CHECK_RESULT_RETURN, Parma_-Polyhedra_Library::Interfaces::Java::get_ptr(), and Parma_Polyhedra_Library::Interfaces::Java::set_ptr().

12.56.1.134 JNIEXPORT jobject JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_-solve (JNIEnv * *env*, jobject *j_this_pip_problem*)

Definition at line 1758 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_java_pip_status(), CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.135 JNIEXPORT jlong JNICALL Java_parma_1polyhedra_1library_-PIP_1Problem_space_1dimension (JNIEnv * *env*, jobject *j_this_pip_problem*)

Definition at line 1625 of file ppl_java_globals.cc.

References CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.136 JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_PIP_1Problem_-toString (JNIEnv * *env*, jobject *j_this*)

Definition at line 1569 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.137 JNIEXPORT jlong JNICALL Java_parma_1polyhedra_1library_-PIP_1Problem_total_1memory_1in_1bytes (JNIEnv * *env*, jobject *j_this_pip_problem*)

Definition at line 1545 of file ppl_java_globals.cc.

References CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.138 JNIEXPORT jobject JNICALL Java_parma_1polyhedra_1library_PIP_-1Solution_1Node_parametric_1values (JNIEnv * *env*, jobject *j_this*, jobject *j_var*)

Definition at line 2058 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_cxx_variable(), Parma_-Polyhedra_Library::Interfaces::Java::build_linear_expression(), and Parma_Polyhedra_-Library::Interfaces::Java::get_ptr().

12.56.1.139 JNIEXPORT jobject JNICALL Java_parma_1polyhedra_1library_PIP_1Tree_-1Node_artificials (JNIEnv * *env*, jobject *j_this_pip_node*)

Definition at line 1995 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Artificial_Parameter_-Sequence, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Artificial_Parameter_-Sequence_add_ID, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Artificial_-Parameter_Sequence_init_ID, Parma_Polyhedra_Library::Interfaces::Java::build_java_artificial_-parameter(), Parma_Polyhedra_Library::Interfaces::Java::cached_classes, Parma_Polyhedra_-Library::Interfaces::Java::cached_FMIDs, CATCH_ALL, CHECK_EXCEPTION_RETURN, CHECK_-RESULT_RETURN, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.140 JNIEXPORT jobject JNICALL Java_parma_1polyhedra_-1library_PIP_1Tree_1Node_as_1decision (JNIEnv * *env*, jobject *j_this*)

Definition at line 1958 of file ppl_java_globals.cc.

References CATCH_ALL, CHECK_RESULT_ASSERT, CHECK_RESULT_RETURN, Parma_-Polyhedra_Library::Interfaces::Java::get_ptr(), and Parma_Polyhedra_Library::Interfaces::Java::set_ptr().

12.56.1.141 JNIEXPORT jobject JNICALL Java_parma_1polyhedra_-1library_PIP_1Tree_1Node_as_1solution (JNIEnv * *env*, jobject *j_this*)

Definition at line 1932 of file ppl_java_globals.cc.

References CATCH_ALL, CHECK_RESULT_ASSERT, CHECK_RESULT_RETURN, Parma_-Polyhedra_Library::Interfaces::Java::get_ptr(), and Parma_Polyhedra_Library::Interfaces::Java::set_ptr().

12.56.1.142 JNIEXPORT jobject JNICALL Java_parma_1polyhedra_-1library_PIP_1Tree_1Node_constraints (JNIEnv * *env*, jobject *j_this_pip_node*)

Definition at line 1915 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::build_java_constraint_system(), Parma_Polyhedra_Library::Interfaces::Java::cached_classes, Parma_Polyhedra_-Library::Interfaces::Java::cached_FMIDs, CATCH_ALL, CHECK_RESULT_RETURN, Parma_-Polyhedra_Library::Interfaces::Java::Java_Class_Cache::Constraint_System, Parma_Polyhedra_-Library::Interfaces::Java::Java_FMID_Cache::Constraint_System_init_ID, and Parma_Polyhedra_-Library::Interfaces::Java::get_ptr().

12.56.1.143 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_PIP_1Tree_1Node_-finalize (JNIEnv * *env*, jobject *j_this*)

Definition at line 1907 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::get_ptr(), and Parma_Polyhedra_-Library::Interfaces::Java::is_java_marked().

12.56.1.144 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_PIP_1Tree_1Node_-free (JNIEnv * *env*, jobject *j_this*)

Definition at line 1896 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::get_ptr(), Parma_Polyhedra_-Library::Interfaces::Java::is_java_marked(), and Parma_Polyhedra_Library::Interfaces::Java::set_ptr().

12.56.1.145 JNIEXPORT jlong JNICALL Java_parma_1polyhedra_1library_-PIP_1Tree_1Node_number_1of_1artificials (JNIEnv * *env*, jobject *j_this*)

Definition at line 1984 of file ppl_java_globals.cc.

References CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.146 JNIEXPORT jboolean JNICALL Java_parma_1polyhedra_1library_PIP_1Tree_-1Node_OK (JNIEnv * *env*, jobject *j_this_pip_tree*)

Definition at line 1884 of file ppl_java_globals.cc.

References CATCH_ALL, and Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.147 JNIEXPORT jstring JNICALL Java_parma_1polyhedra_1library_PIP_1Tree_-1Node_toString (JNIEnv * *env*, jobject *j_this*)

Definition at line 2022 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::get_ptr().

12.56.1.148 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Poly_1Con_-1Relation_initIDs (JNIEnv * *env*, jclass *j_poly_con_relation_class*)

Definition at line 559 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::cached_FMIDs, CHECK_RESULT_ASSERT, and Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Poly_Con_Relation_init_ID.

12.56.1.149 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Poly_1Gen_-1Relation_initIDs (JNIEnv * *env*, jclass *j_poly_gen_relation_class*)

Definition at line 568 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::cached_FMIDs, CHECK_RESULT_ASSERT, and Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Poly_Gen_Relation_init_ID.

12.56.1.150 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_PPL_1Object_-initIDs (JNIEnv * *env*, jclass *j_ppl_object_class*)

Definition at line 577 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::cached_FMIDs, CHECK_RESULT_ASSERT, and Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::PPL_Object_ptr_ID.

12.56.1.151 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Relation_1Symbol_-initIDs (JNIEnv * *env*, jclass *j_rel_sym_class*)

Definition at line 585 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::cached_FMIDs, CHECK_RESULT_ASSERT, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Relation_Symbol_EQUAL_ID, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Relation_Symbol_GREATER_OR_EQUAL_ID, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Relation_Symbol_GREATER_THAN_ID, and Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Relation_Symbol_ordinal_ID.

12.56.1.152 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Variable_initIDs (JNIEnv * *env*, jclass *j_variable_class*)

Definition at line 677 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::cached_FMIDs, CHECK_RESULT_ASSERT, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Variable_init_ID, and Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Variable_varid_ID.

12.56.1.153 JNIEXPORT void JNICALL Java_parma_1polyhedra_1library_Variables_1Set_initIDs (JNIEnv * *env*, jclass *j_vset_class*)

Definition at line 688 of file ppl_java_globals.cc.

References Parma_Polyhedra_Library::Interfaces::Java::cached_FMIDs, CHECK_RESULT_ASSERT, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Variables_Set_add_ID, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Variables_Set_init_ID, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Variables_Set_Iterator_has_next_ID, Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Variables_Set_iterator_ID, and Parma_Polyhedra_Library::Interfaces::Java::Java_FMID_Cache::Variables_Set_Iterator_next_ID.

12.57 PPL_Object.java File Reference

Classes

- class parma_polyhedra_library::PPL_Object
  Smart pointer to a PPL, C++ object.

Namespaces

- namespace parma_polyhedra_library
  The PPL Java interface package.

12.58 Relation_Symbol.java File Reference

Namespaces

- namespace parma_polyhedra_library
  The PPL Java interface package.

Enumerations

- enum parma_polyhedra_library::Relation_Symbol {

  parma_polyhedra_library::LESS_THAN, parma_polyhedra_library::LESS_OR_EQUAL, parma_polyhedra_library::EQUAL, parma_polyhedra_library::GREATER_OR_EQUAL,

  parma_polyhedra_library::GREATER_THAN }

  Relation symbols.

12.59 Timeout_Exception.java File Reference

Classes

- class parma_polyhedra_library::Timeout_Exception

  Exceptions caused by timeout expiring.

Namespaces

- namespace parma_polyhedra_library

  The PPL Java interface package.

12.60 Variable.java File Reference

Classes

- class parma_polyhedra_library::Variable

  A dimension of the vector space.

Namespaces

- namespace parma_polyhedra_library

  The PPL Java interface package.

12.61 Variables_Set.java File Reference

Classes

- class parma_polyhedra_library::Variables_Set

  A java.util.TreeSet of variables' indexes.

Namespaces

- namespace parma_polyhedra_library

  The PPL Java interface package.

Index