Tutorial IV: Contributing to Sage

Benjamin Hutz

Department of Mathematical Sciences
Florida Institute of Technology

November 8, 2013
Sage-Days 55

Ways to Contribute

`http://www.sagemath.org/doc/faq/faq-contribute.html`

1. Adding new functionality
2. Fixing bugs
3. Reviewing tickets
4. Improving documentation
5. Writing tutorials (or translating existing tutorials)

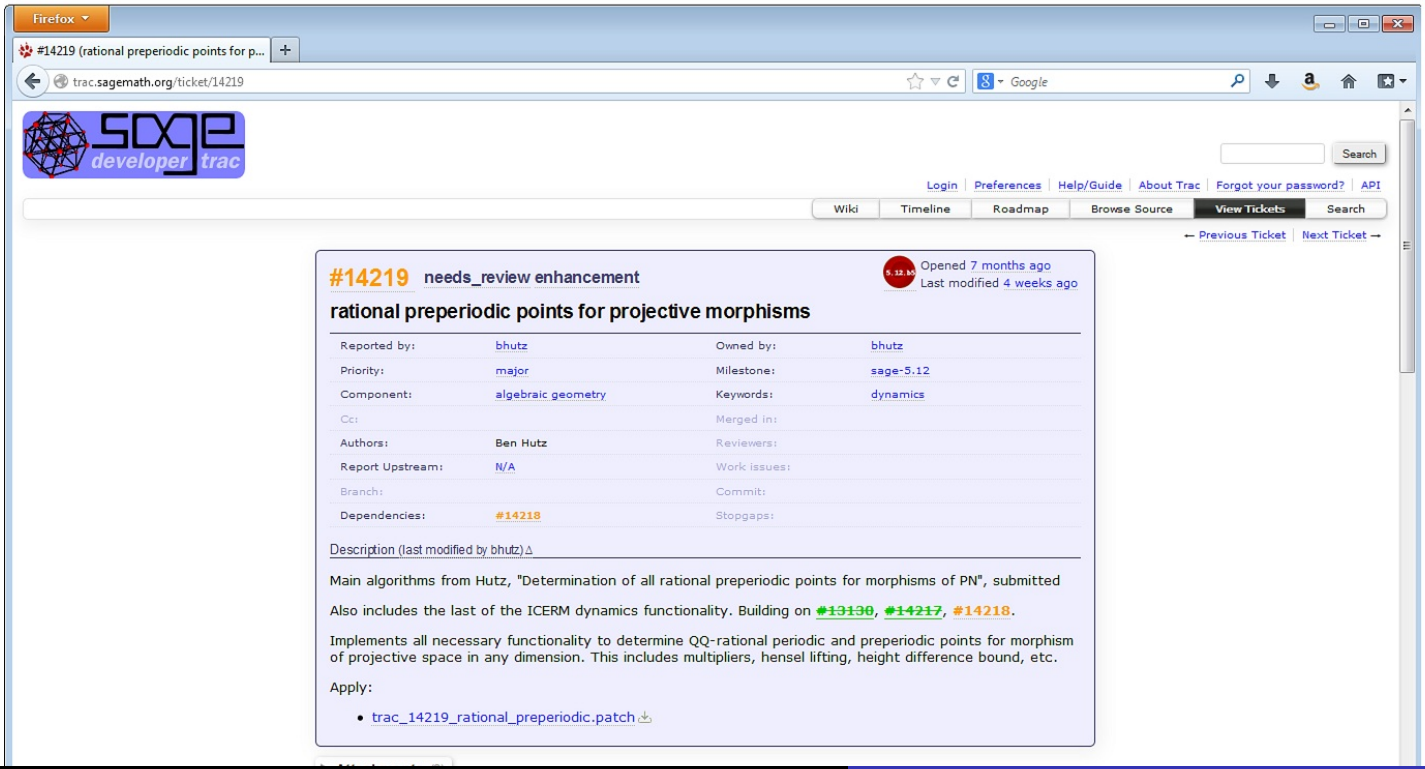
The Trac server: http://trac.sagemath.org/

Both the Sage development model and the technology in Sage itself are distinguished by an extremely strong emphasis on openness, community, cooperation, and collaboration: we are building the car, not reinventing the wheel.
The Trac server lists both bugs in Sage and ideas for features to add to Sage.

The website contains

1. the tickets!!!
2. user guides
   1. old - http://www.sagemath.org/doc/developer/
   2. git - http://sagemath.github.io/git-developer-guide/
3. FAQs

A Ticket


Firefox
#14219 (rational preperiodic points for p...
trac.sagemath.org/ticket/14219
Google
sage developer trac
Search
Login | Preferences | Help/Guide | About Trac | Forgot your password? | API
Wiki | Timeline | Roadmap | Browse Source | View Tickets | Search
← Previous Ticket | Next Ticket →
#14219 needs_review enhancement
Opened 7 months ago
Last modified 4 weeks ago
rational preperiodic points for projective morphisms
Reported by: bhutz
Owned by: bhutz
Priority: major
Milestone: sage-5.12
Component: algebraic geometry
Keywords: dynamics
Cc:
Merged in:
Authors: Ben Hutz
Reviewers:
Report Upstream: N/A
Work issues:
Branch:
Commit:
Dependencies: #14218
Stopgaps:
Description (last modified by bhutz)
Main algorithms from Hutz, "Determination of all rational preperiodic points for morphisms of PN", submitted
Also includes the last of the ICERM dynamics functionality. Building on #13130, #14217, #14218.
Implements all necessary functionality to determine QQ-rational periodic and preperiodic points for morphism of projective space in any dimension. This includes multipliers, hensel lifting, height difference bound, etc.
Apply:
trac_14219_rational_preperiodic.patch

Reviewing a ticket

Adam's talk.

Setting up Mercurial

In your home directory you need a mercurial configuration file '.hgrc' that includes

```
[ui]
username = Benjamin Hutz <bhutz@fit.edu>

[extensions]
#Enable the Mercurial queue extension.
hgext.mq =
```

From the command line you can use mercurial that is included with sage with

sage -hg command

common mercurial commands

1. qapplied - show the queue of applied of patches
2. quanpplied - show the queue of unapplied patches
3. qpop - move one patch from the applied to the unapplied
4. qpush - move one patch from the unapplied to the applied
5. qimport - import a patch file to the unapplied queue
6. qrefresh - save current changes to the current patch
7. qnew - create a new patch
8. export - export a patch to a file.

Starting a new patch

1. First create a 'clone'of your main sage so that you can always revert back.

   sage -clone clonename

2. Create a new patch

   hg qnew trac_#_description.patch

3. Modify the code

coding conventions

```
    def my function (self,parameters):
        r"""
        description

        INPUT:

        - ``name`` -- type, description, default

        OUTPUT:

        - description

        EXAMPLES::

            sage: example
            output
        """
        source code
```

doc tests!

The documentation comes directly from the comment section.

Whenever you run sage -t what you are running are the tests contained in the EXAMPLES: section.

Exporting the patch

1. When everything is working correctly you can save your changes to the patch with

   hg qrefresh

2. You can then export your patch so that others can use it

   hg export tip $>$ path/filename

More on queues

There are two main mercurial queues associated to any sage clone.

1. 'hg qapplied'
2. 'hg qunapplied'

To move patches between the two you use the commands

1. 'hg qpush'
2. 'hg qpop'

Be aware that these really are queues (FILO) so you are limited in how you move patches around.

What a patch file actually looks like

Put in a link from trac.sagemath.org

git

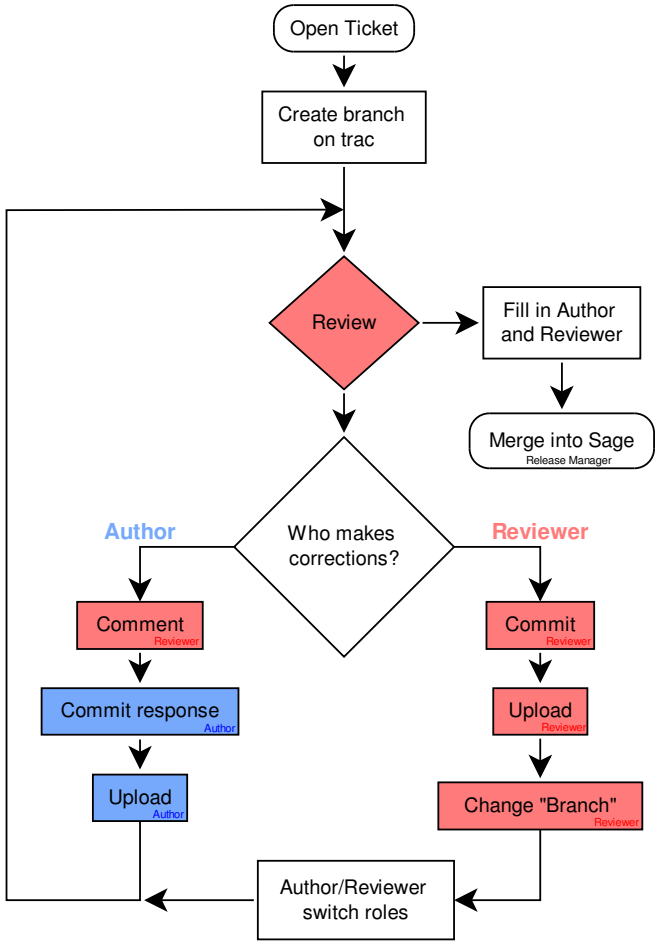
git is a source code management system.

1. Strong support for non-linear (and distributed) development
2. Efficient handling of large projects
3. Cryptographic authentication of history
4. Toolkit-based design

Branches

1. git is based on branches. Given a starting point (or base) you can create a new branch which will include your changes to the base.
2. git has a robust merge system to combine changes from multiple branches together.

Workflow


Open Ticket
Create branch on trac
Review
Fill in Author and Reviewer
Merge into Sage
Release Manager
Who makes corrections?
Author
Reviewer
Comment
Reviewer
Commit response
Author
Upload
Author
Commit
Reviewer
Upload
Reviewer
Change "Branch"
Reviewer
Author/Reviewer switch roles

sage development scripts

git scripts to make writing, reviewing, and collaborating on trac-tickets easier.

```
abandon             Abandon a ticket or branch.
checkout            Checkout another branch (ticket)
clean               Restore the working directory to the most recent commit.
comment             Add a comment to "ticket" on trac.
commit              Create a commit from the pending changes on the current branch.
create-ticket       Create a new ticket on trac.
diff                Show how the current file system differs from "base".
edit-ticket         Edit the description of "ticket" on trac.
gather              Create a new branch "branch" with "tickets_or_remote_branches"
                    applied.
help                show help message and exit
import-patch        Import a patch into the current branch.
merge               Merge changes from "ticket_or_branch" into the current branch.
needs-info          Set a ticket on trac to "needs_info".
needs-review        Set a ticket on trac to "needs_review".
needs-work          Set a ticket on trac to "needs_work".
positive-review     Set a ticket on trac to "positive_review".
prune-tickets       Remove branches for tickets that are already merged into master.
pull                Pull "ticket_or_remote_branch" to "branch".
push                Push the current branch to the Sage repository.
remote-status       Show information about the status of "ticket".
set-remote          Set the remote branch to push to.
show-dependencies   Show the dependencies of "ticket".
tickets             Print the tickets currently in your local repository.
upload-ssh-key      Upload "public_key" to gitolite through the trac interface.
vanilla             Return to a clean version of Sage.
```