REPORT ON OpenDreamKit DELIVERABLE D4.1

Python/Cython bindings for PARI and its integration in Sage

LUCA DE FEO, VINCENT DELECROIX, AND JEROEN DEMEYER

Due on	01/03/2016 (M6)
Delivered on	10/02/2017
Lead	CNRS (CNRS)
Progress on and finalization of this deliverable has been tracked publicly at: `https://github.com/OpenDreamKit/OpenDreamKit/issues/83`	

DELIVERABLE DESCRIPTION, AS TAKEN FROM GITHUB ISSUE #83 ON 2017-02-09

- **WP4:** User Interfaces
- **Lead Institution:** CNRS
- **Due:** 2016-02-29 (month 6)
- **Nature:** Other
- **Task:** T4.12 (#80)
- **Proposal:** p.47.
- **Final report**

The SageMath project includes many different subsystems, mostly written in C/C++. For each subsystem, Sage provides a low-level interface, usually written in Cython, through which the higher level components access the subsystem. The mathematical community would immensely benefit if the low-level interfaces were maintained outside of the Sage project, as separate Python packages. Indeed such decoupling would enable other Python projects to build upon those externalized interfaces, thus helping to improve them, and share maintenance effort.

Of all the subsystems, the case of PARI is of particular interest. Since its inception, Sage has had a low-level Cython interface to PARI, which has evolved over time in a highly coupled way. Around 2012, some Sage developers forked this interface into a project they called CyPari. One of the primary goals of the CyPari fork was to make independent Python bindings to PARI for use in a project called Snappy. Over time, the CyPari fork has diverged from the Sage/PARI interface, and has gotten behind in terms of functionality.

The goal of this deliverable is to reconcile the fork by externalizing the Sage/PARI interface into an independent package, maintained by the Sage community, which may ultimately replace CyPari inside Snappy. The task happened to be more difficult than originally thought. The high level of coupling between Sage internals and the PARI interface makes it very delicate to pull the latter out of the SageMath codebase. The process of making this possible has led to a great amount of refactoring inside the Sage project, which is summarized in Trac ticket 20238.

Because of the high degree of coupling, and thanks to the availability of Snappy, this deliverable constitutes a highly valuable case study for future externalizations of low-level interfaces in SageMath. To bring this deliverable to completion, we have decided to split it in several steps:

- ✓ Move SageMath's C signalling api to a separate Python/Cython package. The package is called cysignals, and is integrated to SageMath 7.1.

- ✓ Decouple SageMath's PARI interface from the coercion model. This has been achieved in SageMath 7.4.
- ✓ Clean up the interface API, by removing unneeded object orientation and external dependencies. This has been achieved, and is integrated to SageMath 7.5.
- [ ] Move SageMath's PARI interface to a separate Python/Cython package depending on cysignals. The package is called CyPari2, and will replace the old PARI interface starting from SageMath 7.6.

The CyPari2 package is not ready to replace the PyPi package CyPari yet. The most important missing functionality is Windows compatibility. A full replacement to CyPari is the goal of deliverable D4.10 #84.

CONTENTS

1. RATIONALE

The SAGE project includes many different subsystems, mostly written in C/C++. For each subsystem, SAGE provides a low-level interface, usually written in CYTHON, through which the higher level components access the subsystem. The mathematical community would immensely benefit if the low-level interfaces were maintained outside of the SAGE project, as separate PYTHON packages. Indeed such decoupling would enable other PYTHON projects to build upon those externalized interfaces, thus helping improve them, and share maintenance effort.

Of all the subsystems, the case of PARI is of particular interest. Since its inception, SAGE has had a low-level CYTHON interface to PARI, which has evolved over time in a highly coupled way. Around 2012, some SAGE developers forked this interface into a project they called CyPari. One of the primary goals of the CyPari fork was to make independent PYTHON bindings to PARI for use in a project called Snappy. Over time, the CyPari fork has diverged from the SAGE/PARI interface, and has gotten behind in terms of functionality.

The goal of this deliverable was to reconcile the fork by externalizing the SAGE/PARI interface into an independent package, maintained by the SAGE community, which may ultimately replace CyPari inside Snappy.

2. WORK ACCOMPLISHED

Completing this deliverable happened to be more difficult than originally planned. The high level of coupling between SAGE internals and the PARI interface makes it very delicate to pull the latter out of the SAGE codebase. Initially planned to be delivered in month 6, it was only completed in month 16.

The process of making this deliverable possible led to a great amount of refactoring inside the SAGE project. As summarized in Trac ticket 20238 (see appendix), it required close to 50 *tickets*, which fell in the following categories:

- Moving SAGE's C signalling api to a separate PYTHON/CYTHON package called cysignals.
- Decoupling SAGE's PARI interface from the *coercion model*.
- Upgrading the PARI interface to the latest upstream version (2.8.0).
- Cleaning up the PARI interface API, by removing unneeded object orientation and external dependencies.
- Moving the PARI interface to a separate PYTHON/CYTHON package cysignals, depending on cysignals.

The end results of this work are the packages cysignals and CyPari2, both installable in a pure PYTHON environment via the standard tool `pip`. Starting from version 7.6, installation via `pip` will also be SAGE's default way of providing the PARI interface.

3. AFTERMATH

The CyPari2 package is not ready to replace CyPari yet. The most important missing functionality is Windows compatibility. A full replacement to CyPari is the goal of deliverable D4.10: "Second version of the PARI Python/Cython bindings".

Because of the high degree of coupling, and thanks to the availability of a third party tool depending on it in Snappy, this deliverable constitutes a highly valuable case study for future externalizations of low-level interfaces in SAGE. Recently, the SAGE developer community has increasingly paid attention to interoperability, as highlighted by the many related discussions on the sage-devel mailing list. We are confident that the experience gained in developing this deliverable will be an accelerator for making SAGE a more modular and interoperable software distribution.

APPENDIX A. SAGE'S TRAC TICKET #20238

http://trac.sagemath.org/ticket/20238

#20238 new task

Opened 11 months ago
Last modified 2 days ago

Move the Sage <-> PARI interface to a stand-alone package CyPari

Reported by:	jdemeyer	Owned by:	
Priority:	major	Milestone:	~~sage-7.5~~
Component:	packages: standard	Keywords:	Pari
Cc:	defeo, slelievre, vdelecroix, mmarco	Merged in:	
Authors:		Reviewers:	
Report Upstream:	N/A	Work issues:	
Branch:		Commit:	
Dependencies:		Stopgaps:	

Description (last modified by jdemeyer) Δ

Needs cysignals:

- ~~#20002~~: Move interrupt.pyx to package cysignals
- ~~#20210~~: Move memory functions to cysignals

Cleanup of PARI interface:

- ~~#20205~~: Clean up factoring PARI interface
- ~~#20213~~: Replace pari_catch_sig_on by sig_on
- ~~#20216~~: Deprecate PARI nth_prime, prime_list, primes_up_to_n
- ~~#20217~~: Remove redundant functions from pari_instance.pyx
- ~~#20219~~: Remove redundant functions from gen.pyx
- ~~#20224~~: Auto-generated PARI functions sometimes return 0 instead of None
- ~~#20226~~: Implement conversion PARI <-> Python int/long without GMP/MPIR
- ~~#20241~~: Separate Sage-specific components from generic C-interface in PariInstance
- ~~#20257~~: Deprecate undocumented arguments to PARI functions
- ~~#20352~~: Initialize PARI constants in `PariInstance.__init__`
- ~~#20473~~: Remove global pari_instance variable
- ~~#20486~~: Remove deprecated PARI code
- ~~#21703~~: Interface PARI precision in bits
- ~~#21806~~: Allow multiple instances of the `PariInstance` object
- ~~#21807~~: Move `gentoobj` and rename it to `gentosage`
- ~~#21808~~: Change `gen.python()` to return Python objects
- ~~#21809~~: Pythonize deprecation warnings in PARI interface
- ~~#21810~~: Move calculation of PARI stack size out of `__init__`
- ~~#22183~~: Rename `PariInstance` -> `Pari`
- ~~#22185~~: Rename `gen` -> `Gen`
- ~~#22195~~: Allow compiling PARI interface without `anal.h`
- #22165: Stop using deprecated PARI function `polred()`
- ~~#22210~~: Remove obsolete special case in PARI `Gen.eval()`
- ~~#22221~~: Fix dependency on PARI headers
- ~~#22222~~: Remove `pari_instance` global in `gen.pyx`
- #22319: Implement `__index__` for PARI `Gen`
- #22321: `Gen.__init__()` does not work as expected

PARI upgrades:

- ~~#20581~~: Upgrade PARI to latest master
- ~~#21005~~: Update PARI to version 2.8.0
- ~~#21756~~: Update PARI to version 2.9.1

Disclaimer: this report, together with its annexes and the reports for the earlier deliverables, is self contained for auditing and reviewing purposes. Hyperlinks to external resources are meant as a convenience for casual readers wishing to follow our progress; such links have been checked for correctness at the time of submission of the deliverable, but there is no guarantee implied that they will remain valid.