Practical Course – Contributing to an Open-Source Project

Preliminary Report on Rust-Lang/Chalk

Florian Sextl[1]

[1]Technical University of Munich, sextl@in.tum.de

1 Overview

This report is meant to be a moderately detailed study about the chalk project in regard to how it is developed and how it implements different aspects of Free/Libre and Open Source Software. It will concentrate on the project's history, how it is governed, how contributions are managed and finally to what extend it is free and open source.

All sources referenced throughout the report can be found solely online and are, thus, object to future changes. Albeit such changes may occur, most sources are kept under version control or archived in some other way.

2 About Chalk

Chalk is a project under the governance of the Rust Language community[1] (Rust-Lang for short in the following) [27]. The main repository can be found at [12]. The chalk project aims at developing a "library that implements the Rust [programming language's] trait system" [31, chapter *What is Chalk?*] as a model against which correct usage of traits can be easily checked. In the Rust programming language traits describe properties about types, e.g. whether they are printable, have a fixed size in memory or can be accessed safely from various threads. Thus, a model of the the trait system is key to reasoning about Rust code and to compile it in a correct and efficient manner.

To this end, the Rust compiler rustc [14] currently uses a different model for trait checking based on a resolution method. This method has certain shortcomings and, in turn, led to the chalk project, which was started to improve upon the current model by lowering the Rust code into a logic program that can then be queried against in a

[1]The term Rust Language community describes both the open source community developing the Rust Programming Language and the entirety of all users of this programming language. It also used as a synonym for the GitHub organization that hosts all projects directly related to the Rust development.

PROLOG-like fashion [31, 28]. The goal of the chalk developers is to have this functionality in an independent library which can then be referenced from both rustc and other language analysis tools. To make developing this library easier, the project repository consists of both the experimental library's basic structure, a Read-Eval-Print Loop application, an alternative query engine and support libraries.

Despite its experimental status, chalk's main library can already be used in the nightly build of rustc[2] and is utilized by the rust-analyzer project, as well. Due to the experimental nature of chalk and its rather often changing interfaces, integrating code in these systems is also part of the development of the chalk project [3, inter alia].

3 History of Chalk

Although the first commit to the chalk repository was on 2015-07-26 [6], actual development began with the first blog post by Niko Matsakis[3] about his vision of chalk [15]. Work on the integration into rustc started at least in 2018 [8], whereas the current usable integration seems to originate from a commit in early 2020 [7]. Another usable integration exists for the rust-analyzer compiler frontend and language server [26] since at least 2019 [9]. The exact commits responsible for the integration of chalk in both rustc and rust-analyzer are rather difficult to determine as both projects have a large commit history and underwent several changes to the source code structure. On top of the rather experimental integrations, chalk is currently in the process of becoming feature complete in regard to the old resolution solver and the goal of the current sprint (2020 sprint 4) is to extract the necessary lowering of Rust types into an own shared type library that can be included in both chalk and rustc to have a common ground [1, 32].

Up to this point, over 70 contributors worked on the project, of which only 4 contributed more than a hundred commits to the master branch [6]. This relatively low number of contributors stands in a stark contrast to the over 3000 contributors of the rustc project and the overall number of contributing members of the Rust-Lang community. In fact, most work for chalk is done by members of the traits working group (wg-traits for short), who oversee the development and integration of chalk. The reason for this rather centralistic development seems to lie mostly in the rather unconventional approach and little necessity to exchange the currently working trait checking system. As chalk's integration in rustc evolves, a broader interest and participation might arise. This thesis is supported by the development of further logic-based checkers for usage within the Rust compiler [13, inter alia] and the ever rising number of contributions to the chalk project.

[2] As a command line flag, e.g. `rustc -Z chalk=yes`

[3] Nicholas D. Matsakis (GitHub user nikomatsakis) is a member of the Rust-Lang core team and team leader of both the compiler and the language subteams.

4 Governance and Funding

Governance of the chalk project is directly intertwined with governance of the Rust-Lang project and the Rust language community. This connection arose, as the main chalk repository got moved first from being Niko Matsakis' private project to the Rust lang nursery group that supports aspiring Rust projects and later became an official Rust-Lang community project[4]. Chalks now parent project, the Rust-Lang project, was originally developed at Mozilla Research and, later on, handed over to the independent Rust-Lang community. After this shift to an own independent governance, the next step, the Rust community is currently in the process of realizing, is the establishment an own Rust Foundation [30] to handle legal and monetary responsibilities completely autonomous.

Until this foundation is in place, the Mozilla Foundation acts as a legal representative [22] and provides resources and sponsoring to the Rust-Lang community and thereby also to the chalk project. Other than Mozilla, several influential technology companies such as Amazon Web Services, Google Cloud and Microsoft Azure provide infrastructure and employ developers specifically to contribute to Rust-Lang projects [20, 4]. The infrastructure provided by these sponsors is accessible to all projects under the governance of the Rust-Lang community and, as such, chalk is a beneficiary and does not need its own resources.

Albeit all sponsors may have a justified interest in the direction of development for the Rust programming language, all decisions are made by the community by consensus. To be more precise, all decisions that would introduce "substantial" [11] changes to one or more Rust-Lang projects are subject to the RFC (Request for Comments) process. An RFC contains a summary of the topic at hand, an exhaustive motivation why this should be done, a sufficient implementation design and further auxiliary information [18]. RFCs can be filed by anyone and are then subject to a consensus decision-making process. Depending on which part of and to what degree the Rust-Lang community would be influenced this process is managed by either the Rust-Lang core team or the corresponding subteam. Each RFCs that is accepted gets a so called shepherd assigned, a subteam member that is responsible for overseeing the RFCs lifetime.

The actual team hierarchy is rather flat with the core team responsible for the general direction of the Rust-Lang development and the subteams responsible for their specified field [10]. Despite the core team having no designated leader, every subteam consists of both members and at least two team leaders that are responsible for organizing the team's affairs. The subteams can also form working groups for dedicated topics that need work outside the regular scope of their field and which are assembled from subteam members. The chalk project in particular is managed by the wg-traits, which is subordinate to both the compiler and the language subteams. As long as chalk is not feature complete in regard to the current resolution method, the compiler team is in charge of the design and actual work. Therefore, the compiler team's internal two-leveled mem-

[4]The exact dates for these moves are not publicly available as repository moves are not recorded in the GitHub history.

ber hierarchy is applied to classify contributors working on the chalk project as well. The top level members are called "full members" whereas the lower level members are called "contributors" [21]. The first kind is allowed to vote on both RFCs regarding the compiler as well as contributors becoming full members and can also be assigned to shepherd RFCs. On top of that, full members have full access to the infrastructure used by their team. Contributors on the other hand have limited privileges and limited access to the infrastructure, yet are also allowed to vote on new contributors. Whereas discussions and voting on RFCs is done in public, team promotions are discussed entirely in a private mailing list.

5 Contributing and Communication

All contributions to the chalk project are subject to the Rust Code of Conduct [19]. This code of conduct is based on the contributor covenant v1.3.0 and the Node.js Policy on Trolling and describes the rules that apply for, inter alia, all contributions to Rust-Lang projects. The actual contribution process for chalk is described both in the chalk book [31, in chapter *Contribution guide*] and a separate CONTRIBUTING file [5], which appears to be an older version of the book chapter. Despite some minor differences, both guides stipulate the GitHub fork and pull model as the standard contribution workflow and cover more details about required steps and information sources. New pull requests are then handled by the the Rust-Lang infrastructure; the highfive bot assigns reviewers if none were stated explicitly in the pull request, the bors bot then handles automatic checks and merges the pull request after all reviewers approved it [23].

Another bot, which is scripted as a GitHub action, ensure, that changes to the solver library's parts are released once a week automatically to the Rust package repository `crates.io`. The automatic release cycle became necessary to keep the integration of chalk in the rust-analyzer up-to-date after that project switched to an own automatic release cycle [2]. Other than by the automatic GitHub action, manual releases with bugfixes and the like can be initiated by full members of the working group at any time.

Although chalk is released at least once a week automatically, the development process is scheduled in another way. The wg-traits plan implementation of new features and more drastic changes to the codebase in six weeks long sprints, mirroring the release frequency of rustc. These sprints are planned in sprint meetings held as video conferences and are archived on YouTube [e.g. 1, the recording of the latest sprint meeting for 2020 sprint 4]. Other than in these sprint meetings, all members and contributors partake in weekly meetings hosted over the Rust lang Zulipchat platform. These weeklies are there to check on the current status of development, delegate urgent bugfixes as well as other tasks and allow the members to exchange on their problems and new ideas. Although the weekly meetings are meant to actually happen each week, the last meeting as of today (2020-12-01) was on 2020-11-17. The Rust-Lang Zulipchat platform accommodates discussions around all aspects of the Rust programming language and especially about implementation details of the compiler and related projects. There are several subteams who have an own stream and host most of their communication over the

Zulipchat platform. Both the compiler subteam and the traits working group manage their communication in this way. Other than Zulipchat, where the developers interact mostly with users, a second important communication platform for the chalk project is GitHub, which is mostly used for inter developer communication. Following the GitHub fork and pull model, changes to the codebase are peer reviewed and discussed in the corresponding comment threads on the pull requests whereas bug reports and feature requests are documented as GitHub issues. In general, all communication about actual work on the chalk project is openly available and thus in compliance with best practices for open source projects. The only kind of private, non-available communication is the internal mailing list used for discussing promotions in the team. Albeit the open availability of all on-topic information, it is rather difficult to find information on how specific decision were made. Both the Zulipchat stream and the GitHub repository link to relevant parts of each other, yet the information itself is spread over many different websites and documents - even some focusing seemingly only on rustc - with only little options to search all of them. It is rather easy to follow the development process once one is accustomed to all these sources but difficult to find out about all of these in the first place and even harder to research the exact history of the project as was necessary for this report.

6 Licenses

Chalk is doubly licensed under both the MIT and the Apache 2.0 licenses. This approach is common to all Rust-Lang projects [24] and results in strong permissions with only little conditions. In particular, all of chalks code and documentation is free for commercial and private use and may be distributed with or without modifications as long as the original license and copyright notices are left unchanged. These permissions even hold if parts or all of chalk is licensed under different terms or is used as part of a bigger closed-source project. Due to the double licensing with the Apache 2.0 license, users are also granted an explicit patent grant.

This specific choice of licenses is especially interesting as Rust was originally under the governance of the Mozilla Foundation, the parent organization of the Mozilla free software community. Most Mozilla projects are actually licensed under the weak copyleft Mozilla Public License[5]. In comparison to other compilers and related tools, the way licenses are used for the Rust-Lang is somewhat standard yet not too common. Many tools are licensed under the GNU General Public License or a GPL-conform license, whereas the MIT and Apache licenses are less commonly used [33].

The rather permissive way of licensing Rust-Lang projects may be an important factor to the ever growing interest in the Rust programming language from both developers [cf. 29] and technology companies [4]. Especially companies are often reluctant to support projects that are licensed under a copyleft license such as the GPL.

[5] e.g. the Firefox Browser [16]

7 Conclusion - How much FLOSS is Chalk?

Due to the licenses alone, chalk is an open source and free software project. In addition to this rather formal classification, chalk also satisfies some more practical open source traits. First and foremost it is hosted as a public repository on GitHub and developed by a changing community of distributors. As stated before, this community communicates openly about the development process and is open to new contributors. To help these new members getting to know the project, the chalk repository contains both a number of dedicated issues for beginners as well as a rather short README file that refers to the chalk book for more detailed information. This delegation of information is quite common for Rust-Lang projects and, therefore, despite it being unusual in comparison to other language projects, not too difficult to understand and orient in. Furthermore, Rust-Lang's code of conduct has a strong focus on creating an inclusive and open community space and is asserted by a dedicated, platform-independent moderation team [25]. It therefore ensures an environment suited for a diverse community and the open interactions needed for well functioning open source projects. Last but not least, chalk puts all best practices for open source maintainers [17] into practice and can, as a result, be seen as a good open source project.

References

[1] *Archived 2020 Sprint 4 Meeting*. 2020-10-30. URL: https://youtu.be/MHp_otI28UU (visited on 2020-11-24).

[2] *Archived discussion regarding automatic releases for chalk*. URL: https://zulip-archive.rust-lang.org/144729wgtraits/37430publishing.html (visited on 2020-11-27).

[3] *Archived discussions regarding chalk integration in rustc*. URL: https://zulip-archive.rust-lang.org/144729wgtraits/17410reintegratingchalkintorustc.html (visited on 2020-11-24).

[4] Matt Asay. *Why AWS loves Rust, and how we'd like to help*. URL: https://aws.amazon.com/de/blogs/opensource/why-aws-loves-rust-and-how-wed-like-to-help (visited on 2020-11-26).

[5] *Chalk CONTRIBUTING.md File*. URL: https://github.com/rust-lang/chalk/blob/master/CONTRIBUTING.md (visited on 2020-11-27).

[6] *Contribution history on GitHub repository of chalk*. URL: https://github.com/rust-lang/chalk/graphs/contributors (visited on 2020-11-24).

[7] *Early commit to the current usable chalk integration in the rustc repository*. 2020-05-07. URL: https://github.com/rust-lang/rust/commit/a24df5b3cdeb49334d6cde1c4d983b2354616824 (visited on 2020-11-24).

[8] *Early commit with a chalk integration in the rustc repository*. 2018-03-13. URL: https://github.com/rust-lang/rust/commit/3a50b41da4cbb135fc74cdc8ebf2b09edb396f87 (visited on 2020-11-24).

[9] *Early commit with chalk integration in the rust-analyzer repository.* 2019-04-05. URL: https://github.com/rust-analyzer/rust-analyzer/commit/b9c0c2abb79769852119dc9a595e63ee74eeba03 (visited on 2020-11-24).

[10] *General Governance Structure of the Rust Community.* URL: https://www.rust-lang.org/governance (visited on 2020-11-26).

[11] *GitHub repository containing the RFCs.* URL: https://github.com/rust-lang/rfcs (visited on 2020-11-26).

[12] *GitHub repository of chalk.* URL: https://github.com/rust-lang/chalk (visited on 2020-11-24).

[13] *GitHub repository of polonius.* URL: https://github.com/rust-lang/polonius (visited on 2020-11-26).

[14] *GitHub repository of rustc.* URL: https://github.com/rust-lang/rust (visited on 2020-11-24).

[15] Nicholas D. Matsakis. *Lowering Rust traits to logic.* 2017-01-26. URL: https://smallcultfollowing.com/babysteps/blog/2017/01/26/lowering-rust-traits-to-logic (visited on 2020-11-24).

[16] *Mozilla Firefox Licenses.* URL: https://hg.mozilla.org/mozilla-central/raw-file/tip/toolkit/content/license.html (visited on 2020-11-28).

[17] *Open Source Guide: Best Practices for Maintainers.* URL: https://opensource.guide/best-practices (visited on 2020-11-28).

[18] *RFC Template.* URL: https://github.com/rust-lang/rfcs/blob/master/0000-template.md (visited on 2020-11-26).

[19] *Rust Community Code of Conduct.* URL: https://www.rust-lang.org/policies/code-of-conduct (visited on 2020-11-26).

[20] *Rust Corporate Sponsors.* URL: https://www.rust-lang.org/sponsors (visited on 2020-11-26).

[21] *Rust Forge - Compiler Team Membership Description.* URL: https://forge.rust-lang.org/compiler/membership.html (visited on 2020-11-26).

[22] *Rust Forge - Legal counsel.* URL: https://forge.rust-lang.org/core/legal.html (visited on 2020-11-26).

[23] *Rust Forge - Review policies.* URL: https://forge.rust-lang.org/compiler/reviews.html (visited on 2020-11-27).

[24] *Rust Lang Licenses.* URL: https://www.rust-lang.org/policies/licenses (visited on 2020-11-28).

[25] *Rust Lang Moderation Team.* URL: https://www.rust-lang.org/governance/teams/moderation (visited on 2020-11-28).

[26] *rust-analyzer Website.* URL: https://rust-analyzer.github.io (visited on 2020-11-24).

[27] *rust-lang.org*. URL: https://rust-lang.org (visited on 2020-11-24).

[28] *Rustc Dev Guide*. URL: https://rustc-dev-guide.rust-lang.org/traits/chalk.html (visited on 2020-11-24).

[29] *Stack Overflow Developer Survey 2020*. URL: https://insights.stackoverflow.com/survey/2020#technology-most-loved-dreaded-and-wanted-languages-loved (visited on 2020-11-29).

[30] The Rust Core Team. *Laying the foundation for Rust's future*. URL: https://blog.rust-lang.org/2020/08/18/laying-the-foundation-for-rusts-future.html (visited on 2020-11-26).

[31] *The Chalk Book*. URL: https://rust-lang.github.io/chalk/book (visited on 2020-11-24).

[32] *Weekly Workgroup Meeting 2020-11-17*. 2020-11-17. URL: https://zulip-archive.rust-lang.org/144729wgtraits/36754meeting20201117.html (visited on 2020-11-24).

[33] *Wikipedia: Non-exhaustive list of compilers*. URL: https://en.wikipedia.org/wiki/List_of_compilers (visited on 2020-11-28).