# The `.ray.txt` Programming Language

A grant proposal to
through
.

OrbitMines Research

7 September 2025

## Please select a call

### Thematic call

In the list of current calls below, please indicate the call topic you are responding to. Note that some larger funds (like NGI0 Core and NGI0 Commons Fund) (part of the Next Generation Internet initiative) will have some special scope or conditions. You'd better have a look at them before you submit a proposal. If in doubt, submit to our open call and mention in the application that you are okay with us allocating your proposal to the most suitable fund.

Open Call

## Contact information


### Your name (max. 100 characters)

Fadi Shawki

### Email address

fadi.shawki@orbitmines.com

### Phone number

+31 6 84704186

### Organisation (max. 100 characters)

OrbitMines

### Country

The Netherlands


## General project information

### Project name (max. 100 characters)

The `.ray.txt` Programming Language

### Website / wiki


Abstract: Can you explain the whole project and its expected outcome(s).

Please be short and to the point in your answers; focus primarily on the what and how, not so much on the why. Add longer descriptions as attachments (see below). If English isn't your first language, don't worry — our reviewers don't care about spelling errors, only about great ideas. We apologise for the inconvenience of having to submit in English. On the up side, you can be as technical as you need to be (but you don't have to). Do stay concrete. Use plain text in your reply only, if you need any HTML to make your point please include this as attachment.
(you have 1200 characters)

At some point in the future, programming languages will have to evolve to be something not *just* text-based. Something which is more fluid and can combine the traditional text-based interfaces with more interactive interfaces. This project is a text-based programming language in anticipation of that future. So in one sentence: A programming language with the syntax of Python, with the functionality of Lean, and interactivity of the Wolfram Language.

Currently, I have a triplet of files which contain all design notes regarding the language - some 2000 lines long -: (the two files here https://github.com/orbitmines/ray/tree/main/ar.ray/ray.ray.txt and https://github.com/orbitmines/ray/blob/main/ar.ray/ray.ts/src/ray.ts).

The project at the start will be implementing all those notes within the syntax of the language. Then writing a compiler and runtime. Then working on the ecosystem for interactivity.

What makes the language unique is that it treats everything like a structure similar to a hypergraph. As an example: It implements something like a Graph as a refinement of Hypergraph, and Array as a refinement of Graph. Then something like a binary string: A bit is an Array of length 2. Which finds itself defined at points in another Array which is the binary string.

Like that the intermediate representation used in the compiler, is the same as the language itself; as the control-flow graph is just another hypergraph.

You also get the unique property: that checking the equality of two objects simply become checking the equality of hypergraphs (and custom defined equivalences). And it's automatically defined for all objects in the language.

If at the end of this project I can start working on an IDE (as the next project) in the language itself. I'd consider it a success.

(Some more details of specific features of the language I've put in the other sections)

## Have you been involved with projects or organisations relevant to this project before? And if so, can you tell us a bit about your contributions?

(Optional) This can help us determine if you are the right person to undertake this effort

I have not been involved with other projects related to this project before. Except for a minor bug-fix, this is the first time I'm contributing to an open source project.

# Requested support

## Requested Amount

(between 5000 and 50000)

24.000 €

## Explain what the requested budget will be used for?

Does the project have other funding sources, both past and present? (If you want, you can in addition attach a budget at the bottom of the form). Explain costs for hardware, human labor (including rates used), travel cost to technical meetings, etc.

Other than my own, the project has no other funding sources.

The budget will be used to sustain myself for over a year to work on the project. A breakdown would be an estimate of 2.000 € /month. So x12 = 24.000 €.

## Compare your own project with existing or historical efforts.

(e.g. what is new, more thorough or otherwise different)

Back in 2023 I spent two months studying many existing projects (https://github.com/orbitmines/library/blob/main/phases/Project%20Index.md) in the space of programming languages and theorem-provers.

There are several design patterns which exist in other languages, but no solution exists which combines all of them:

(1) By design there's some overlap with languages like Python, Ruby and TypeScript in the ways of syntax.

(2) Languages like Agda, Lean, ... Idris implement features like dependent types, higher inductive types, types like "this function terminates" and (interactive) theorem proving. Though there's overlap there, to my knowledge no language exists which treats all objects as spaces like graphs (though higher inductive types are a step in that direction).

(3) Barely any interactive theorem provers exist like Chyp (https://github.com/akissinger/chyp) or the ZX-calculus theorem provers, which are based on a form of (hyper)graph rewriting.

(4) The ability to use abstract interpretation directly in the language. This is always only reserved for the compiler implementation of the language and not exposed to the language. (An example of this might be calling a function which accepts a number with a range of numbers)

(5) Similarly, compilers and something like Lean, have proofs of equivalence of functions. But there's no native support for saying "this function can be implemented in these ways" let the compiler choose one it thinks is better. (Say one implementation for the specification, another for efficiency) An example of what having that enabled in the .ray.txt language: recursively defining all boolean operations in terms of each-other, then telling the compiler:

"There are multiple ways in which you can implement all boolean operations, based on the system you're compiling to you can choose how to whether to implement them or use some primitive."

(6) As for interactivity with a programming language (which would be the next step in the evolution of this programming language). Something like the Wolfram Language is a canonical example of possibilities in that space.

### What are significant technical challenges you expect to solve during the project, if any?)

(optional but recommended)

(1) One of the big challenges is a design problem: How do you intuitively encode a structure like a (hyper)graph in text-based format. - In order to define types which are some form of a hypergraph. (This will likely be some form of recursive definition like inductive types/enums)

(2) Another technical challenge is the building of the compiler; to something like JavaScript (for running in the browser), C++ or directly integrating with compiler technology like LLVM. As it is the first time I'm doing something like this, figuring that out will be a challenge.

(3) As working with theorem provers is quite new to me, implementing that functionality into the language will be a challenge. (As is working with proofs in possibly infinite spaces)

(4) The technical challenge of writing some of the algorithms for working with hypergraphs.

(5) One feature of the interactive theorem prover I'd like, and part of language called function intensionality (so checking whether the implementations of two functions mean the same thing) is something I'm still unsure about how to implement.

### Describe the ecosystem of the project, and how you will engage with relevant actors and promote the outcomes?

(E.g. which actors will you involve? Who should run or deploy your solution to make it a success?)

Like any programming language the ecosystem will require a package manager once people start using it, but initially this is not a requirement for setting up the project.

It will at the start be a game of getting programming language enthusiasts interested in the language. It'll be similar yet different enough from existing programming languages that it should spike some interest. So a start of posting it on forums and seeing if there's interest is a start.

Once the text-based programming language is in place, a big part of the ecosystem will be a custom IDE/and IntelliJ/VS Code plugins built on top of it. Which will provide the basis for its future ecosystem: A move away from only being a text-based programming language.

## Footnotes & References