Perl 6: Project: Part 4

Elliot Chandler (Wallace)

10 October 2017

1. Data types

Perl 6 includes a large amount of non-primitive types in the language itself: 33 compound types, 56 "domain-specific" types, and 113 exception types. This includes common compound types such as arrays, lists, and hashes, as well as various other types for a range of applications, such as for documentation, concurrent programming, and I/O for various operating systems. Perl 6 includes a large amount of capabilities within the language core itself. That allows the base language installation to be quite functional. It means that it is not necessary to install as many additional libraries to have a practical base of tools [1].

2. Standard library

Perl 6 does not itself have a specified standard library *per se*, but the Rakudo Star distribution (the primary way to obtain an implementation of the language) does include a variety of additional *modules* (46 of them) [2]. Rakudo Star includes tools for performance debugging and profiling, such as `debugger-ui-commandline`, `grammar-debugger` and `Grammar-Profiler-Simple`. It provides the `Terminal-ANSIColor` module for creating colored output from command-line programs, and `test-mock` and `Test-META` for creating unit tests. Rakudo Star provides tools for data interchange and serialization, in the form of seven modules for working with JSON, and one for XML, as well as a database interface module (`DBIlish`). For graphics, Rakudo Star comes with the `svg` and `svg-plot` modules. It also includes several such as `perl6-http-easy` and `perl6-http-status` for network interoperation. Perl 6 includes a package manager module, also, called `zef`, which provides a command-line interface for managing the packages installed in a system [2].

3. Semantics of expression evaluation

Perl 6 makes a distinction between "statements" and "expressions". Expressions in Perl 6 are statements that return a value. A program in Perl 6 semantically represents a list of statements (things for the computer to do). To make a statement act as an expression, the `do` keyword can be used, which will cause the statement given to represent its value in the given context. Perl 6 generally evaluates statements in the order in which they are presented in the program (following the natural control flow), although internally it will divide up statements to be run concurrently in multiple threads if the result is the same [1].

4. Type coercion

By default, Perl 6 when encountering a variable that does not have a specified type will convert it to the necessary type when it is used in a context where its current type is not applicable. To override this behavior, types can be specified for each object, allowing strict type checking. When a specific type is requested for a variable, assignments to that variable will not be coerced, and explicit casting is required [1].

5. Semantics of assignment statements

Perl 6 has complex assignment semantics, similar to Java, because it uses containers for some values, but values can also exist without containers. Containers are object-oriented wrappers around the native (native to the Perl 6 virtual machine) types. The documentation writes that "The assignment operator asks the container on the left to store the value on its right" [1]. Because of this, the actual effect of assigning a value to a name depends on the type of container that has been created by variable declaration. In practice, this is largely transparent to the user, and the context of a reference to a variable will affect whether the container is used or the value within the container is used. If the user wants control over this, the language provides some constructs for controlling these issues [1], but in my experience the defaults have generally worked well.

A list can contain containers or values. Because of this, it is not always possible to assign to list elements, because lists are immutable and values cannot be changed. However, if the items in a list are containers, they can be changed because the container itself will stay the same. That is because the containers themselves do not change; rather, what they point to changes. Unlike lists, arrays only hold containers. Consequently, their contents can be changed [1].

References

[1] Perl 6 Documentation, "Perl 6 documentation: https://docs.perl6.org/," 19 Sept. 2017. The official Perl 6 documentation is very helpful for getting an understanding of the language.

[2] Rakudo organization on GitHub, "Rakudo Star: star/modules at master: https://github.com/rakudo/star/tree/master/modules," 2017.

[3] Perl 6 Documentation, "FAQ: Frequently asked questions about Perl 6: https://docs.perl6.org/language/faq," 19 Sept. 2017. I used this article to learn some basic information about Perl 6.

[4] R. Holloway, "The must-have features for Perl 6: https://opensource.com/life/16/9/perl-6-features," 23 Sept. 2016. This article provided a resource for learning about some of the things that distinguish and differentiate Perl 6 from other languages.

[5] E. Miller, "A review of Perl 6: https://www.evanmiller.org/a-review-of-perl-6.html," 13 Aug. 2017. This article discusses Perl 6's merits and weaknesses.

[6] jnthnwrthngtn, "Grammar::Tracer and Grammar::Debugger: https://perl6advent.wordpress.com/2011/12/02/grammartracer-and-grammardebugger/," *Perl 6 Advent Calendar*, 2 Dec. 2011. Debugging grammars is hard out of the box.

[7] M. Lenz, *Perl 6 Fundamentals: A Primer with Examples, Projects, and Case*

Studies. 2017. This book provides a general introduction to Perl 6. It seems quite accessible to me.

[8] M. Lenz, "How to debug a Perl 6 grammar: `https://perlgeek.de/en/article/debug-a-perl-6-grammar`," 1 Sept. 2010. This is more information on debugging Perl 6 grammars.

[9] Stack Overflow, "Floating point inaccuracy examples: `https://stackoverflow.com/questions/2100490/floating-point-inaccuracy-examples`," 2010–2013. A citation for the claim regarding floating points being a little odd.

[10] Hacker News, "Why hasn't Perl 6 taken off yet?: `https://news.ycombinator.com/item?id=12888784`," Nov. 2016. This is where I learned that Perl 6 Rationals aren't arbitrary-precision.

[11] The Perl 6 Programming Language, "The Perl 6 programming language: `https://perl6.org/`," n.d. This home page for the Perl 6 language introduces it and its culture. Retrieved 20 Sept. 2017.

[12] L. Rosenfeld and A. B. Downey, "Think Perl 6," 2017. I used this book as a supplementary resource for getting an overview of Perl 6.

[13] Perl 6 project, "NQP — Not Quite Perl (6): README.pod: `https://github.com/perl6/nqp`," 31 May 2017. This is where I got information about how Perl 6 is implemented, and its portability.

[14] Perl 6 Documentation, "Grammars: `https://docs.perl6.org/language/grammars`," 19 Sept. 2017. This documentation page was

one of my primary sources for getting an understanding of the scope and capabilities of grammars in Perl 6.

[15] Wikipedia, "Perl 6: `https://en.wikipedia.org/wiki/Perl_6`," I used Wikipedia's article on Perl 6 as a way to find other articles and books to look at (as well as Fogler Library and Web search).

Appendix

The appendices are not included here due to length. Tables of the composite types, domain-specific types, and exception types are available in the Perl 6 documentation [1]. A list of modules included with Rakudo Star is available on request from this author, based on the information from [2].