Unraveling the Legacy of Haskell Brook Curry

Pioneer of Compositional Logic

Chentian Wu

2024 Spring MATH 473

University of Wisconsin-Madison

February 29, 2024

1. Introduction

Haskell Brook Curry ——an American mathematician and logician, is a pivotal figure in the history of **Mathematical Logic** and **Computer Science**. While not as widely celebrated as *Alan Turing*[1] nor as extensively influential as *Kurt Gödel*[2], Curry's contributions to humanity in the real historical context can rival both. If the Turing Machine[3] is considered the fundamental model of modern computer programming languages, then the notion of Computable Functions[4] represents another equivalent model alongside the Turing machine. Combinatory logic[5], the result of Curry's lifelong dedication, has become a shining star in theoretical computer science research, with functions as its fundamental elements. Perhaps unbeknownst to many, Curry stands as the only mathematician and logician whose entire name has been used in computer language nomenclature:

Haskell: the most popular functional programming language today;

Brook: a stream processing programming language developed at Stanford University, geared towards graphics processing and parallel computing, designed based on the ANSI C standard;

Curry: represents a novel language - a blend of functional and logical paradigms, with its foundation rooted in Haskell. In a sense, Curry can be seen as a superset of Haskell.

2. Curry's Life

Curry entered Harvard University at the age of 16 to study medicine but later switched to mathematics. After completing his undergraduate stud-

ies, he worked for two years in the electrical engineering department at the Massachusetts Institute of Technology before returning to Harvard to study physics, where he obtained a master's degree. It was during this time that Curry developed an interest in mathematical logic after taking a course on *Principia Mathematica*[6] by *Whitehead*[7] and *Russell*[8]. This prompted him to pursue a doctoral degree in mathematics at Harvard.

Initially under the guidance of his mentor, George Birkoff, Curry researched differential equations. However, his fascination with mathematical logic persisted. In 1927, while at Princeton University, Curry discovered the research of Russian scholar *Moses Schönfinkel*[9] in the field of combinatory logic, which greatly intrigued him and solidified his commitment to the study of **Mathematical Logic**. He then traveled to Göttingen, Germany, where he collaborated with two scholars familiar with Schönfinkel's work, *Heinrich Behmann*[10] and *Paul Bernays*[11]. Under the supervision of his advisor, the renowned *David Hilbert*[12], Curry completed his doctoral thesis on combinatory logic.

Curry's career saw him teaching at Pennsylvania State College for 37 years, where he made significant contributions to the development of computer science. Following retirement, he briefly taught at the University of Amsterdam before returning to Pennsylvania State College until his passing.

3. Research of Haskell Curry

Like many mathematical logicians of his time, Curry focused on foundational issues in mathematics, aiming to establish a secure theoretical foundation for the discipline. In 1933, Curry learned of the **Kleene-Rosser Paradox**[13] through correspondence with *John Rosser*[14], another notable math-

Figure 1: Haskell Brooks Curry

ematical logician. This paradox, discovered jointly by Kleene and Rosser, demonstrated the incompleteness of several formal systems proposed by Alonzo Church and Curry himself. Despite the completion of this proof, Church, Kleene, and Rosser abandoned further research into the foundations of mathematics, whereas Curry persevered, declaring himself a "**deserter never of paradoxes**".

Curry placed all his hopes for the **Foundations of Mathematics** on **Combinatory Logic**, dedicating his life's work to its development. Although the concept of combinatory logic was initially proposed by Moses Schönfinkel, much of the research and advancements came from Curry, earning him recognition as one of its founders. From a modern perspective, **Combinatory Logic** and **λ-Calculus**[15] together form the theoretical basis of **Functional Programming Languages**, with λ-Calculus pioneered by Alonzo Church[16]. However, Church's reputation and the fame of λ-Calculus overshadowed Curry and combinatory logic for a considerable period.

Curry's greatest contribution, and the focus of his lifelong research, is combinatory logic. In addition to this, he also discovered the Curry Paradox, named after him, and the famous *Curry-Howard Isomorphism*, also known as the *Curry-Howard Correspondence*. Moreover, his name is used as a verb

—*currying* —which refers to the process of transforming a multi-parameter function into a series of single-parameter functions.

4. Currying and an Example

In mathematics and computer science, currying[17] is the technique of translating the evaluation of a function that takes multiple arguments (or a tuple of arguments) into evaluating a sequence of functions, each with a single argument. Since the logic part requires too much mathematical knowledge that I could barely cover them in 1500 words, I think using some simple programming codes here can make it easier to explain the main work of currying.

I'll firstly show the direct definition of currying: given a function

$$f : (X \times Y) \to Z$$

currying constructs a new function

$$g : X \to (Y \to Z)$$
$$(x)(y) \mapsto f(x, y)$$

for x of type X and y of type Y. We then also write

$$\text{curry}(f) = g$$

To illustrate currying, let's first introduce the concept of **types**.

In programming languages, expressions have values. We often say that an expression has a value of `1` or a value of `'a'`. Although they are stored in memory in binary format, they are conceptually distinct. To distinguish between them, we assign different types.

We refer to the value `1`'s type as `Int`, representing an integer, and the value `'a'`s type as `Char`, representing a character.

Thus, we establish a layer of abstraction on the concept of values, where different values have distinct properties based on their types.

In mathematical terms, types act as an equivalence relation on the set of values, inducing a partition of this set. Values of the same type form an equivalence class.

A function is used to transform one or more values into another value. Functions are also considered values and therefore have types.

For example, the addition function `add` takes two integer values and produces another integer value. Once we have defined the add function, we can call it like this:

```
1       add 1 2
2   =   3
```

Now, let's consider a question: what happens if we only provide one parameter to `add`? What is the result of `add 1`?

We can visualize it like this: consider add as a machine with two slots. If we provide values `1` and `2` to each slot respectively, the machine will produce the result `3`. Now, if only one slot is filled with `1`, what happens? Clearly, it's still a machine, but with only one slot filled. Therefore, `add 1` remains a function, but it only accepts one parameter. It returns the result of adding `1` to the provided parameter value. We can denote its type as:

```
1   add 1 :: Int -> Int
```

Let's reconsider the type of `add`. We observe that when `add` is given `1` as a parameter, it returns the function `add 1`. Therefore, we have an alternative perspective on `add`. It can be viewed as a machine with one slot: when provided with the parameter 1, it produces another machine with one slot, `add`

1. Hence, we can denote the type of add as:

```
add :: Int -> (Int -> Int)
```

For convenience in writing, we agree on right associativity for `->`, thus:

```
add :: Int -> Int -> Int
```

In fact, according to the notation provided above, Currying refers to a higher-order function as follows:

```
curry :: ((a, b) -> c) -> (a -> b -> c)
```

It transforms a function `f` into a function `g`, such that:

```
f :: ((a, b) -> c)
g :: a -> b -> c
```

where `g = curry f` and `f = uncurry g`. This ensures that for any `x` and `y`, it satisfies:

```
f (x, y) = g x y
```

You might wonder what's the point of curring. Actually, curring brings much flexibility and makes it easier for composition: curried functions can be easily composed together to create more complex functions. By partially applying functions and combining them, one can create new functions without modifying existing ones, leading to more flexible and expressive expression.

Curring is a fundamental concept in the future development of so-called functional programming languages, which are derived from the λ-Calculus. At that time, Currying had few applications in fields other than logic, but Curry's vision was ahead of its time. Today, decades later, the **Haskell** programming language based on Curring's functions has become one of the most popular programming languages in the world, widely used Used in various fields of engineering.

Figure 2: Haskell Programming Language

5. Conclusion

From medicine to applied mathematics, to analytics, and then to logic, Curry adjusted his research direction three times in his life. As an double-major student, I'm deeply inspired by Curry's perseverance and decisiveness. I now major in theoretical computer science and mathematics (especially logic), and hope that I could make a difference in these fields, like Dr. Curry did.

References

[1] Wikipedia contributors. Alan turing — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Alan_Turing&oldid=1210984284, 2024. [Online; accessed 29-February-2024].

[2] Wikipedia contributors. Kurt gödel — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Kurt_G%C3%B6del&oldid=1204990087, 2024. [Online; accessed 29-February-2024].

[3] Wikipedia contributors. Turing machine — Wikipedia, the free encyclopedia, 2024. [Online; accessed 29-February-2024].

[4] Wikipedia contributors. Computable function — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Computable_function&oldid=1208892479, 2024. [Online; accessed 29-February-2024].

[5] Wikipedia contributors. Combinatory logic — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Combinatory_logic&oldid=1210817185, 2024. [Online; accessed 29-February-2024].

[6] Alfred North Whitehead and Bertrand Russell. *Principia Mathematica*. Cambridge University Press, 1925–1927.

[7] Wikipedia contributors. Alfred north whitehead — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Alfred_North_Whitehead&oldid=1210197147, 2024. [Online; accessed 1-March-2024].

[8] Wikipedia contributors. Bertrand russell — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Bertrand_Russell&oldid=1211107483, 2024. [Online; accessed 1-March-2024].

[9] Wikipedia contributors. Moses schönfinkel — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Moses_Sch%C3%B6nfinkel&oldid=1182829400, 2023. [Online; accessed 1-March-2024].

[10] Wikipedia contributors. Heinrich behmann — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Heinrich_Behmann&oldid=1174543561, 2023. [Online; accessed 1-March-2024].

[11] Wikipedia contributors. Paul bernays — Wikipedia, the free encyclopedia, 2024. [Online; accessed 1-March-2024].

[12] Wikipedia contributors. David hilbert — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=David_Hilbert&oldid=1209839108, 2024. [Online; accessed 1-March-2024].

[13] Wikipedia contributors. Kleene–rosser paradox — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Kleene%E2%80%93Rosser_paradox&oldid=1136515418, 2023. [Online; accessed 1-March-2024].

[14] Wikipedia contributors. J. barkley rosser — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=J._Barkley_Rosser&oldid=1163420770, 2023. [Online; accessed 1-March-2024].

[15] Wikipedia contributors. Lambda calculus — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Lambda_calculus&oldid=1209818880, 2024. [Online; accessed 1-March-2024].

[16] Wikipedia contributors. Alonzo church — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Alonzo_Church&oldid=1206821694, 2024. [Online; accessed 1-March-2024].

[17] Wikipedia contributors. Currying — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Currying&oldid=1205270387, 2024. [Online; accessed 1-March-2024].