A Hitchhiker's guide to Lambda (λ)

Shitohichi Umaya

2000-4-11(Sat)

Contents

Abstract

A long time ago, I read an article about λ calculus. It said this calculus is invented for reconstructing the system of mathematics. I was interested in the idea "reconstructing the mathematical system." What does that mean? There are many λ calculus introduction including Wikipedia. But, some of the procedure is hard to understand for me. Here is a memo about that.

1 Introduction

This article is based on my English version blog [2, 3] between 2008-9-8 and 2009-4-10.

2 λ calculus and Hitchhiker's guide

One day, I wonder about λ calculus, and looked up Wikipedia. There are nice entry about that, but I could not understand some examples of calculation. It took a week to figure out. Maybe this is trivial, but I would like to brag myself about that. There is a common saying, "they brag most who can do least."

I am not familiar with λ calculus. If I encounter something not familiar, I usually lookup some kind of guide book. I went to some city, I have a book called "Chikyu no aruki kata (How to walk on the earth)," or Hitchhiker's guide to the Galaxy. This article could be a tiny version of a hitchhiker's guide to λ calculus.

According to the Hitchhiker's guide to the Galaxy, the Hitchhiker's guide to the Galaxy is the most successful book in the universe. However, it seems no one knows the book is written in which language. If that is the most successful book, it seems all the people living the universe can read. Or, also according to the guide, there is no population in the universe, maybe no one read the book. But then what does it mean the most successful...

The guide could have an artificial intelligence, and all the people might read the guide using a babel fish. (In case you do not know the guide, the babel fish is a translation fish between any kind of species in the universe using some kind of telepathy. But the translation quality is sometimes not so good. There is a famous translation engine after this fish on the web.)

But the main character of the guide, Arthur, can read the guide before he implanted a babel fish. Then, the guide should have some kind of translator. There could be a tiny possibility that there is an earth language (Queen's English) version of the guide, but I am sceptical of that. Because the guide company tried to sell one version of guide to all the parallel universe including the past and the future.

Even there is a translator included, there must be a "native" or "internal" language of the guide. I can not imagine what kind of language is used for the guide, but some part related logic or mathematics could be understandable. I am pretty sure some entries about mathematics or logic in the guide. It is interesting for me that what kind of logical expression in there. It seems it is infinite improbable, but, it may be λ calculus.

3 λ calculus and function

The word lambda calculus itself has "calculus," so it can calculate numbers. But, when I started to learn the lambda calculus, I want to know that what is "calculate" means. I also think about "what is the number?" If I want to teach "what is the number" to small children, I do not know how to teach it. Also I did not recall how to learn that. But, I think I know what numbers are.

I can tell that there are some properties about numbers. First, it does **not** matter how to read it. In the hitchhiker's guide, there was a planet called Earth. The people on the planet have a lot of languages. For example, English, German, Latin, Japanese, and so forth. Interestingly enough, every language seems to have a concept about numbers. There are many representations for numbers, for example, (1, 2, 3, ...), (one, two, three, ...), (ein, zwei, drei, ...), (ichi, ni, san, ...), and so on. But no matter how you read them, there should be a common concept underneath. In the guide, maybe (herring, sandwiches, herring sandwiches, ...) or something like that. I know the definition by negation (it does not matter how to read them) is not a good idea here, but this at least tells you one aspect of numbers. There is a concept "numbers" which is independent from how to read them. Then, what is the substance of numbers? You can not say, it is one, two, three, ... Inside the guide, there must be "something," and they are presented in some representation according to readers. The representation is always chosen as the reader can understand. I called it "native" in one section before. This is the substance of the guide, someone could call it as information. But, it does not matter which language is employed to store such information inside of the guide. We could investigate further the substance of the guide and also substance of numbers.

I think λ calculus is a mathematics that talks about function. Function is a useful idea in mathematics. It's like a vending machine that if you put something in (for example, money), then something comes out (for example, a drink). Many of the cases, people imagine a box which converts from something to another thing. For instance, a converter box which tells you that one Altair dollar is how much Sirius yen. The guide must also have such a box inside.

The function is not necessary to be a box. However, I learn in such way at my school. Moreover, the character of "function" in my native language means "a number box." So, I usually write a box and put a name like f or g. I think this was a standard on earth. Anyway, both teachers and students did not think about this deeply. Especially, if the usual people know what is function, then computer scientists have a problem since they do not know what it is. Nowadays function is a popular idea, so, computer scientists need to invent something fancy.

Most of the computer scientists use the words "λ calculus" as to deceive the people. It is easy to find such people, if you ask them "What's λ calculus?" then they will answer, "fixed-point...," "Combinator blah," "computability..." to deceive you.

λ calculus might be a deep idea and might be useful to think about function. Although I am just a Sunday wannabe mathematician, I could enjoy some superficial results. I could even finish this with "Church-Rosser blah blah" to deceive you. However, as you know I am now writing this article. Such a person usually wants to brag that "I know it! I know it!" So, I will continue this story at least until Marvin will show up. By the way, Marvin is **the** genus robot in the Hitchhiker's guide. He/(or She/It?) has the most depressed mind with the best brain ever made in the universe by Sirius cybernetics corporation. This company first made a brain which has equal ability to a few hundreds bionic brains. But the result shows that a genus is crazy. So be Marvin.

Finally, this section is ended here since I introduced my favorite Marvin.

4 Intermission 1

I have a Japanese version of this page. There was a good question on my last article. "What happen if the vending machine (a function) is broken?"

First of all, we need to define what the broken means. Some of you would say, "Broken is broken, what else?" But, this answer does not make any sense for mathematicians. Are there no common sense in the head of mathematician? Maybe, yes. But, there is a reason.

I am a Sunday mathematician/programmer. Mainly I program a code to solve some of my problems. To tell my computer to solve my problem, I need to interpret my problem to a code which my computer can execute/understand. Many of mathematics formulation is really formulated, which means you do not need to understand what it is, they are just a procedure. Then my computer can execute to solve my problem. I formulate some problem since after that is done, I do not need to think about that. Rest of the problem is solved automatically. This is fun for me. "Computer, search some information. Computer, find the shortest path from my home to the station. Computer, find the cheapest ticket under this condition..." All these things must be translated to a code which my computer can understand. "Broken is broken" is not enough for the computer. Since computers are so stupid so far.

For example, "broken" could mean: 1. when you input something, but nothing comes out, 2. when you input anything, the output is always the same, or 3. when you input something, the output seems totally random. Which does the broken mean for you? None of them? Current usual computer can not guess, we need to tell that.

Also, "broken" has subjective meaning. There are chips which have build-in self destruction mechanism. For example, some kind of decryption chips. These chips contain a secret encryption key. Some malicious people want to read the information. When the chip detects such activity, it breaks itself. Some of the credit card chips and DVD copy detection chips have this function.

When you can not read the information, usually it means "broken." But the designer of these chips designed to do that. If someone can still read the secret information, it is broken for the designer. Therefore, "broken" is subjective meaning. When someone's credit card is stolen, the owner usually does not want to his/her card available anymore. If the chip does not self destructed by the malicious one's attack, then the owner may sue the designer, "The chip was not broken, because it was broken. If it is not broken, it should have broken itself." Human can understand this means, but, it is difficult for current machines.

λ calculus thinks about all kind of functions. Therefore, such "broken" function (whatever it means) must be described. If λ calculus can not describe some of the functions, that is the limit of this calculus. But, a person must define what is the broken means if he/she want to describe it in λ calculus. Mathematician is a such a lazy people, but they are also perfectionist. The art of mathematics is how to reach the "perfect laziness." Therfore, all kinds of functions are of course considered. In the a few thousand years of mathematics history, one era is ended when the people find out the perfectness of mathematics system or limit of mathematics system. But this story is too large here and I would like to concentrate only at the some aspect of the λ calculus.

5 Introduction to λ calculus

Standard mathematics books explain mathematical stuffs as a sequence definition, theorem, proof, definition, theorem, proof, ... This is quite simple and enough abstracted. Therefore, we can also explain λ calculus in the standard way. But Marvin will sure complain that is so depressed. More abstracted theory could be more applicable to many things. It becomes less unnecessary stuffs, then, it becomes simpler and also more beautiful in a sense. The theory is to the point when more abstract.

Japanese sword seek for the beauty in the sword itself, it never decorates with some kind of jewels. Because a sword maker/master thinks the beauty comes from the sword itself. They shamed if they need to cover a sword with non-sword component. We can find many swords, staffs, ... are decorated with gold or some jewels. I can also see some kind of gorgeousness in that, however, I prefer beauty in these kind of simpleness. A French pilot said "A designer knows he has achieved perfection not when there is nothing left to add, but when there is nothing left to take away." I sympathize with this word. Also, I like the story about a ship called Vasa [4, 8], which sank in the sea. The story about a project manager (a Swedish King) requested too many features to his ship. Any software comes from Sirius Cybernetics coop. has too much features and hard to use, because the project managers believe that the customers want to have new features instead of a simple and stable software. (Kode Vicious Pride and Prejudice (The Vasa) CASM Vol.51, No.9)

The beauty of mathematics is in the simpleness and abstraction. I could understand that a mathematician wants to discuss more simple and more abstract entity. But then the mathematics becomes more difficult to the usual people like me. I see here some kind of closeness. If only a few people who knows mathematics well can enjoy the mathematics, that's a bit sad for me.

When I discover some beauty, I sometimes want to keep it to only myself. If I discover some beauty which nobody doesn't know, I feel some privilege. Because I need a lot of effort to find out something beauty in mathematics, it is a kind of reward for me. If I explain them to many people and people also find the beauty, that's nice. On the other hand, I sometimes feel some kind of loneliness. Like it is not only mine anymore. Someone who loves mathematics seeks for more beauty in that, then she/he may search for more abstract form. The result becomes more substance, all the background, history, etc. will be lost. That's also the reward for mathematicians. It is pure substance, however, it is not familiar with me anymore. I do not feel that I understand that anymore.

I also prefer abandoned part of mathematics. Why λ calculus is created? How it is created? What was the first attempt of that? These may be not so important after constructed the λ calculus. "What can λ do?" becomes more important. Usually "what is it?" is not so important in mathematics like operators are more important than some kind of substances like numbers.

Vogons in the guide are extremely officious, their relationship is dry. Only important thing is what others can do for them. "Who" did is no matter at all. For them, even relative is only a relationship which people knows whom. Even if their grandmother ask to help her children, they do nothing unless there is a contract. A family may be worse than bureaucracy. A dry software company only asks employees to implement new features. This is important for business. But, who did it is not so important. This means, it does not matter who did that. It is just a function of the company. However, if a company does not applicate or does not matter the people, the people usually leave such kind of company. So, Vogonism is hardly work in the human society. But it seems Vogonism is working in mathematical theory.

The important thing of λ is also what the λ calculus can do. However, this is a Hitchhiker's guide. Let's talk about not-so-important things, like, why λ is created. λ calculus is also made by a person. Therefore, there must be some kind of motivation. Marvin is the most depressed existence in the universe, he (she/it?) has no motivation at all. He is always depressed. He even has no motivation to suicide... But, the designer of Marvin from Sirius Cybernetics coop. did not want to make a depressed robot

(I think). The designers just thought if they can combine a few hundreds of genius brains, such thing should be super-genus. They are enough to smart to design Marvin, however, they are not enough smart to imagine that such super-genius is insane. There is a purpose of λ. Let's talk about the motivation of λ next time.

6 Motivation of λ calculus

According to the Wikipedia [7], λ calculus was introduced by Church and Kleene in the 1930s as part of an investigation into the foundations of mathematics.

By the way, I am an amateur Sunday mathematician, therefore, please do not believe this blog without check by yourself. This is just I think I understand these stuffs. There must be many errors. I try to avoid errors, but, this is not my profession. I am also not confident about my English. Welcome the comments. I could say in cool way, I was inspired by the Wikipedia's page. If you understand the Wikipedia's entry, I don't recommend to waste of time by reading this blog. There are bunch of interesting text around the world. When you read "λ calculus was introduced by Church and Kleene in the 1930s as part of an investigation into the foundations of mathematics.", and if you think "I see, that's the reason of why λ calculus was introduced. That's easy." then, you don't need to read this blog. But, if you think "In 1930? It seems very recently as the history of mathematics. Why is the foundations of mathematics considered in this time? Wait, what is the foundations of mathematics? Didn't Pythagoras, Apollonius, or Euclid study the foundations of mathematics in B.C.?" Then, you may continue to read this text.

The foundations of mathematics means that where the mathematics can start, or what kind of starting point is all right for mathematics system. Mathematics starts with a set of definition. A person should define them. One of the most fundamental mathematical object would be numbers. Also propositions and logic, how to calculate numbers would be fundamental stuffs in mathematics. These seems too obvious. Many mathematicians did not care them until 20th century. Or Euclid did too good job about formulation and we needed not re-consider until 20th century.

But, why such obvious things are important at that time? Mathematicians started to think about formalization at that time. It may differ the mathematics between languages, how can we sure they are exactly the same. For example, number '1' is exactly the same in English and Japanese? Then, they wanted to define numbers. If someone want to define numbers, it should be done without numbers. Otherwise, we can define the numbers as "numbers are numbers." For example, "Love is love," "Consciousness is conscious of consciousness." These may be true, but, we can not use it to think about the foundations of mathematics. Then a question is how can we define numbers without using numbers. This seems paranoia, but, mathematicians are perfectionist.

I found this is an interesting idea. Because these are enough to formalized, abstracted, and simplified. We could make a machine to implement that system. If you want to create a machine to compute something, we need to make it by some kind of materials (water, stone, iron, ...). Zaphod may say with his two heads, "You know, 1 or 2 or 3, something like that. Just make up a machine which can understand them." But I think this is not so easy. Numbers are such an abstracted idea. Abstracted means that there are a lot of applications. "What is the common idea between the earth, humans, a bread, signals, Zaphod's heads, and a church?" I can answer that "they are countable." I've already said, but I have no idea how to teach the numbers to children. That's a highly abstracted idea.

The motivation of λ calculus is to think about the foundations of mathematics. This means that re-thinking all the mathematical system from the numbers and calculus. At that time, mathematicians are interested in more in the sanity of the mathematical system by formalization. They started to doubt "obvious things" like numbers 1, 2, 3,.... However, this formalization is based on enough simple system and we could build a machine which can partially process the system. The reason I am interested in λ calculus is this part "we can build a machine of that." Also this is the reason that λ calculus is the basic theory of the computer and computer languages.

Let's define the natural numbers without using 1, 2, 3, ...

7 Defining natural numbers

7.1 Definition of natural numbers

Peano defined natural numbers Peano [1]. He actually described properties of natural number, not seems to try to define the natural numbers. But these are somehow the same. The following five definitions are called Peano's axiom which defines the natural numbers. If you are not familiar with mathematical notation, it might be hard to get what they said. But, the basics are not so difficult. These are copied from Mathworld [1].

1. Zero is a number
2. If a is a number, the successor of a is a number.
3. Zero is not the successor of a number.
4. Two numbers of which the successors are equal are themselves equal.
5. (induction axiom.) If a set S of numbers contains zero and also the successor of every number in S, then every number is in S.

The first definition said, there is the first number called Zero. Here it said Zero, but it does not matter which number is. It should be a "something." However, you may ask "What is something?" It is really just "something" Why we need to say the first number as Zero? It is fine as One, 42, -1, or x. You can write in Japanese "Rei", or in German, "Null." In short, this is "something." But, one thing I would like to make this clear, this "Zero" is not the number 0. Because we want to define numbers, so we do not know any numbers, even 0. It is just something the first number and we can just call it Zero. Personally, I prefer to write x since it seems more "something."

Definitions are similar to rules of a game. Therefore, we should not think about why this is defined. That is just a starting point of the discussion. It is easy to imagine that this is hard for especially someone who is not familiar with mathematics. This is the same as rules of some game, like soccer game, "A player is not allowed to touch a ball by hand except goalkeepers." If you ask "Why a player can not use his/her hands?" Then one can only answer that "That's a rule of the soccer game." This is also true as the rule of chess, shougi, or go... If there are ten kind of games, there are ten kind of rules. Mathematics is the same, there are many rules of mathematics and each rule makes different mathematics. We can make arbitrary kind of mathematics, only necessary condition is such mathematical system must be consistent. But, most of the arbitrary rules can not make interesting mathematics. "Interesting" is quite subjective word and it seems not so fit to mathematics. But, many can feel that. I sometimes encountered that some people believe that the mathematics is a kind of truth in the universe. But, mathematics is nothing related with how the universe is. That's the physics's area. However, well established mathematics can describe our universe well. I am fascinated this point of mathematics. It is like a game/sport that has a well established rule are so fun and interesting. As there are many kind of games and sports, there are many mathematics and each mathematics may have different rules. Although, many rules can be shared in mathematics. One can make up new rules and can create a new sports. But it is difficult to create a new interesting sports. It is the same in mathematics, you can create own mathematics easily by making up several definitions, but it is very difficult to make an interesting new mathematics.

The second definition means that we can create a next natural number from the current one. This function creates a successor number from the current number, therefore it called "successor" function. This defines "plus one" function. We have already had the first natural number, Zero, then we can make a successor number from Zero. This successor number is "something" of Zero. It is usually called one, but not necessary. This definition just said, it is something different from Zero. We have now:

1. there is the first number,
2. we can make a successor number from a number.

Out of these two definitions/rules, it seems we can make the whole natural numbers, but this is not enough for that.

The third definition said that there is no loop of successor function, i.e., if you repeat the successor function staring with x, the result of them will never return to the number x. This seems natural since x + 1 + 1 ... + 1 != x. There is such mathematics (modulo), however, Peano's natural number does not think about that. For example, there are such system, like 12 + 1 = 1, or 31 + 1 = 1. If you think this is odd, think about your calender. Amazingly, the next day of December 31st (31.12) is January 1st (1.1). If you look at the month part, 12 + 1 = 1 and days part is 31 + 1 = 1. Time has also similar system, next of 23:59 is 00:00, means 59 + 1 = 0. Some might say that is not a calculation, but, there is such calculation in your computer. Someone may think 1 + 1 = 2 is the simplest mathematics. I think it is not so simple.

The fourth definition said that the same number's successors are always the same. In other word, different two numbers's successors are never the same. The calender example also does not follow this rule. 31 + 1 = 1 for January, 28 + 1 = 1 for February, 30 + 1 = 1 for April. 31, 28, 30 are not the same number, but the + 1's result are all 1. This definition tells you that the calender system never happens in the natural number.

I think now Marvin want to complain my explanation since these are so obvious and not need to explain. The last definition said all the natural numbers follows this rule are also the natural number. This is the fundamental of mathematical induction.

Marvin: "By the way, this explanation is quite similar to the explanation from Kouji Shiga's book. You should refer that." That's true. This explanation comes from "A story of growing mathematics." Unfortunately, I do not have this book now. I left them in Japan... I like all the Shiga's book. When I found out that he had an open seminar at Yokohama, I visited his class. It took four hours to go and back the class by train. It was fantastic classes. When I left Japan, I regret that I could not take his class anymore.

Peano's axiom reminds me two things: Lao-tsu and Wittgenstein. The coincidence works again for the guide. The section 42 of Lao-tsu starts with "Tao is created by one. One bears two, two bears three, and three bears everything..." I have no idea about what Wittgenstein try to explain, however, his word: the words are just projection of the world. I think it sounds right. λ calculus projects back this words to a machine. I.e., logic (logos) is projected to the world again. So far, I just describe calculation with words. But no matter what words I use, this can be run on a machine, which belong to the world. I am satisfied when I see the logic really runs on a machine. Then I feel this is not just meaningless words.

Next time, let me describe the natural numbers by λ.

7.2 Church numerals

Last time we were talking about how Peano defined the natural number. Because λ calculus defines the numbers based on that. Mathematical formulation makes the discussion (called proof) more exact. This "exact" is important for mathematician. The formulation causes increasing the exactness, which means, there are no such thing, like "You know about the numbers, just do something like calculation in appropriate way." Because even every single obvious issue should be defined in formulation. As the side effect, we could execute these rules on a machine — we can make a computer! That's the interesting point for me.

There are many ways to how to implement a computer as a machine. Pascaline [5] and Charles Babbage's differential engine [6] are gear based. λ calculus does not suggest direct implementation form, it is symbolic and more abstract. But, it provides a suggestion of implementation.

Before Marvin points out, this explanation is Masahiko Sato and Takafumi Sakurai's "Basic theory of programming." Prof. Sato's class was tough. At least I had totally no idea if I only attended. However, when I asked questions, even though they are extremely stupid questions, he answered the question until I said "thanks I understand." The problem is the class was so tough, therefore, I do not have any questions except "What can I ask?" It is very difficult to figure out what is I don't figure out.

Here, I use a circle as a sign of "This is a number." I use a square as Peano's "Zero," and a successor number is represented by reputation of "Zero". See Figure 1.

As you see, there is no one or two. We have only signs of "number" and "Zero." Our number system is depends on radix 10 system. We could also use Roman number system, Babylonian system, or Chinese characters. These are all for human's convenience, and all of these number representations are not substance. Zero is sufficient to represent the Peano's system. I use Zero as a square, since I would like to stress that "Zero" is also a name for something. It does not matter if it is "hoge," or "Petrosiliuszwackelmann."

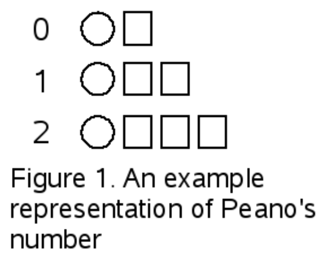

Figure 1: An example representation of Peano's number

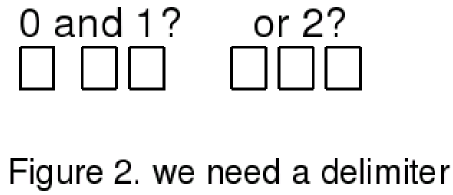

Figure 2: Necessity of delimiter to distinguish the numbers

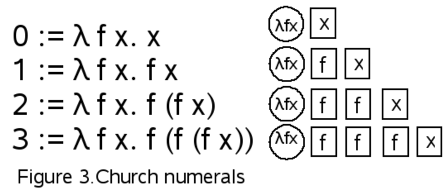

Figure 3: Church numerals

Then I use a square as following the Prof. Sato's book.

When you see one square, then I understand you want to call it one. I agree with that. But, we need to represent Zero. The usual number 0 means "There is nothing here" – nothing exists here. If we want to start the number Zero, then I think this is the way. Although, Peano's system can start any number of natural number, one, two, or 42. It does not matter where to start. But still I would agree with Figure 1's Zero looks like one.

Let's start with why we need a circle instead of using only Zero in the next section.

7.3 Church numerals (2)

A circle was a symbol to represent a number. But, there is no such thing (a symbol to represent a number) in Peano's axiom. Peano's axiom only defines a Zero and a successor. We employed a square to represent Zero. But when we tell two numbers to a machine, we can not distinguish two numbers if we have only Zeros. See Figure 2. Therefore, we use a circle as a delimiter. One could say, we can use a space, but we also need to tell a space to a machine, otherwise any machine can not know a space exists. We need something like a number 0. 0 means "there is nothing." If we write down nothing, how we could know something is missing. If we put 0, then we know nothing actively exists. 0 can represent "existence of nothing." This is an excellent invention of human being.

By the way, speaking about space, there is no space character in Japanese. I think also Korean and Chinese do not have space character. Therefore, a processing of Asian text starts with finding words. A space character represents no character, but existence of no character makes so easy to find words. Space character is also a brilliant invention. I remind myself that Laotsu's "usefulness of uselessness."

The Church numerals, one of the representations of numbers of λ calculus, are defined in the same manner of Figure 1:

$$\begin{aligned}
0 &:= \lambda f x.x \\
1 &:= \lambda f x.f x \\
2 &:= \lambda f x.f(f x) \\
3 &:= \lambda f x.f(f(f x))
\end{aligned}$$

It seems these are not numbers, but these satisfy Peano's axiom, therefore, these are numbers. Because no matter what it looks like, Peano's axiom defines what the number should be. If the condition is satisfied, they are numbers. That's the rule. If you closely look this numbers, the number of 'f' is represents the number. Zero has no 'f,' 1 has one 'f'. Two has two 'f's . Three are also the same. This is the definition of Church numerals. As you see in Figure 3, this is almost the same as Figure 1's number. (Almost means both f and x are square. Different things should have different symbols. This is not good, but this is for computation.)

Marvin: What a long story to define just numbers! You spent nine articles to define it. In the Wikipedia's λ calculus page, when the Church numerals are defined, it has already defined all formal λ expressions. You said just one is one square, two is two squares, ... this simple thing took so long. It is OK to explain something simple, but it is so pedantic. You have not

Figure 4. SUCC mark I (Sirius Cybernetics corp.)

Figure 4: SUCC Mark I (Sirius Cybernetics corporation)

even defined how to compute numbers. I have experienced five universes time. But this is too slow for me. So depressed.

But Marvin, this is the way I understand the Church number. I am an amateur mathematician, so I could not understand if the definitions of Peano's axiom is only presented. It is so abstracted and dry for me.

Marvin: Doesn't matter. I am tired. You defined numbers. Now what? Roman number is better than this for computing. How can you write 2008? My co-processor will be depressed.

OK. We have now numbers, let's compute it.

8 SUCC mark I

Blaise Pascal made a calculator for helping his father's job (tax calculation). It is painful to calculate a huge amount of accounting. I am not good at calculation. So, I thought if I have a computer, I do not need to compute anything myself. That's one of the motivation I took a computer science course at my university. Some people totally misunderstand that a computer scientist is good at arithmetic. No. If someone is good at arithmetic, why does she/he need to learn that? If a human can fly faster than sound, maybe we do not need to use a plane. If people can communicate without speaking between thousand km away, why we need a telephone? I can not do arithmetic, therefore I learned computer science.

Figure 4 is a computer SUCC mark I by Sirius Cybernetics corp. This computer gets one Church number as an input, and outputs another Church number. This computer does not understand what the number is, but just execute one procedure. It is a kind of vending machine. A vending machine can calculate changes, this SUCC mark I also can compute something.

SUCC mark I runs computation as the following.

1. Figure 5. Initial state. Given an input Church number on the 'input' board. Here, Church number 1 is the input as an example. (Do you remember that the Church number 1 is one circle and one square?)

2. Figure 6. The input is copied on the output. SUCC mark I scan the input and choose the same tile from left to right.

3. Figure 7. The calculation result. After a copy, SUCC mark 1 puts one more square tile at the end of the output.

As you see, when we input the Church number 0, then the machine outputs Church number 1. The same procedure can generate 2 from 1, 3 from 2, and so forth. Finally, we have a computer! and please shut up, Marvin.

Marvin: ....

It seems ridiculously simple, but this is the basic of our computer. By the way, SUCC represents Successor. This is a successor number generator. As we have already seen, this function and Zero generates all natural numbers. We could add more procedures.

9 Pop1

9.1 Instructions of Pop1

We thought about a machine which generates the next number of the input last time. Here, we have seen a procedure of "increment one." We coined this procedure as "increment one," like this has something meaning. Or for human, this procedure has a meaning, "increment one." But, the machine SUCC 1 just moves some symbols around. SUCC 1 did not understand the numbers. Here is too much to think about what is "meaning" or what is "understanding." There is a theory called "Society of Mind" by Marvin Minsky. The book said a complex combination of simple functionalities creates intelligence, or you can not distinguish such complex thing from an intelligent thing. But SUCC 1 is such a simple machine and I could safely say it has no intelligence. (By the way, are there any relationship between Marvin Minsky and the robot Marvin?) Although human being prefers that a machine performs something meaningful, it does

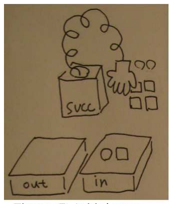

Figure 5. Initial state.
The input has Church
number 1.

Figure 5: SUCC Mark I. Initial state

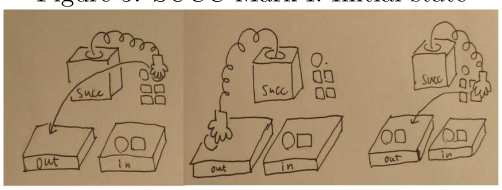

Figure 6. Scan the input and replicate the
'symbols' (circle and square) on the output

Figure 6: SUCC Mark I. Scan the input and replicate the 'symbols' (circle and square) on the output

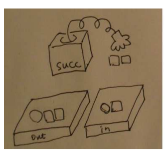

Figure 7. The calculation
result of SUCC mark I

Figure 7: SUCC Mark I. The result

not matter for the machine (or a machine can not matter anything so far). Performing a procedure becomes a computation. Intriguing, it is not necessary to understand what the computation means to perform computations.

SciFi novel usually describes this as a problem. A machine just executes its instructions without any understanding. We can make any dangerous machine (from Cordwainer Smith, The Instrumentality of Mankind's Menschenjager to a movie Terminator, and so on) Only moral can stop this. My favorite is "Variant 2" by Philip K. Dick, but it is too much here, so back to λ calculus.

Figure 8 shows the machine Pop1. Pop1 can execute three instructions.

1. Copy input to output: Copy the contents of input table to the output table. (Figure 9)

2. Delete head and tail: Delete head and tail on the output table (Figure 10)

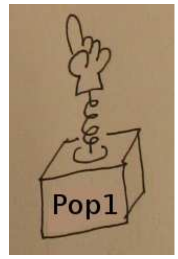

Figure 8: Pop1 (Sirius Cybernetics corporation)

3. Add head and tail: Add a head and tail to the number on the output table (Figure 11)

These instructions are always valid if the input is a Church number. If the input is not a Church number, Pop1 casts an error, usually it gets angry and throws the numbers to you. Sirius Cybernetics Coop.'s manual states that "Please put a helmet on your head before running the program of Pop1."

Let's write a Pop1's program using these Pop1 instructions.

9.2 Pop1 Program

Pop1's program is fixed. We can not change the program. The Cybernetics coorp. considers their software is top secret matter. Even the people working for the company can not see it. Actually developers of the software can not read that. But, if you run the program, Pop1 displays what instruction is currently executed. So we can just write them down (Figure 12). (The references (a), (b), (c) are Figure 9, 10, 11, respectively.) You may find that Pop1 has only three instructions.

Marvin: It's a waste to have three instructions in such a machine.

Of course, we could construct a computer with one single instruction, which is Turing equivalent to any computers we have. But, human needs understandings, Marvin. Please do not forget that. (There are several variations of one instruction set machine. My favorite one instruction computer is in the book, Computer Architecture: A Quantitative Approach.)

If you never wrote a computer program, you might wonder this is a real program or not. This is a real program. All the program stuffs are composed of small program units (instructions), even the program of the weddings, TV program, Olympic game program, ... are the same way.

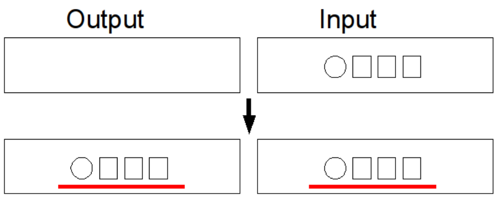

Figure 9: Pop1 procedure (a): copy input to output

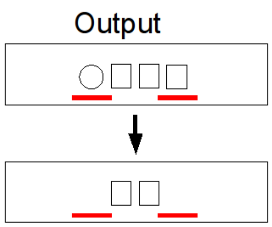

Figure 10: Pop1 procedure (b): Remove head and tail on the output

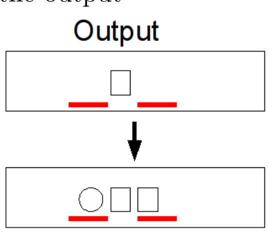

Figure 11: Pop1 procedure (c): Add head and tail on the output

Figure 13 shows all the procedures.

Now, what does this program? It does not matter the machine does something without meaning "for human being," but I can show something meaning example here. Pop1 gets two inputs A and B, and then outputs one result C. Here A, B, and C are called variables. I will write this as

- Pop1(A, B) -> C

Set A is 1, B is 2, then the result is

- Pop1(1, 2) -> 3

Pop1 gets input 1 and 2, then the result is 3. Let's see more.

- Pop1(0, 0) -> 0
- Pop1(0, 1) -> 1
- Pop1(0, 2) -> 2
- ...

1. (a) Copy
2. (b) Delete head and tail
3. (a) Copy
4. (b) Delete head
5. (c) Add head

Figure 12: Pop1 program

- Pop1(1, 0) -> 1
- Pop1(1, 1) -> 2
- Pop1(1, 2) -> 3
- ...
- Pop1(2, 0) -> 2
- Pop1(2, 1) -> 3
- Pop1(2, 2) -> 4
- ...

Can you see what is Pop1? This is "Plus operation." Now you see why the machine's name is Pop1 (Plus OPeration 1). The name is for human. It actually does not matter if it is Lambda1, Machine1, Hokuspokus1, or whatever. Here I followed the book from Sato and Sakurai.

Let's rewrite this as in the usual notations.

- Pop1(0, 0) -> 0: $0 + 0 = 0$
- Pop1(0, 1) -> 1: $0 + 1 = 1$
- Pop1(0, 2) -> 2: $0 + 2 = 2$
- ...
- Pop1(1, 0) -> 1: $1 + 0 = 1$
- Pop1(1, 1) -> 2: $1 + 1 = 2$
- Pop1(1, 2) -> 3: $1 + 2 = 3$
- ...
- Pop1(2, 0) -> 2: $2 + 0 = 2$
- Pop1(2, 1) -> 3: $2 + 1 = 3$
- Pop1(2, 2) -> 4: $2 + 2 = 4$
- ...

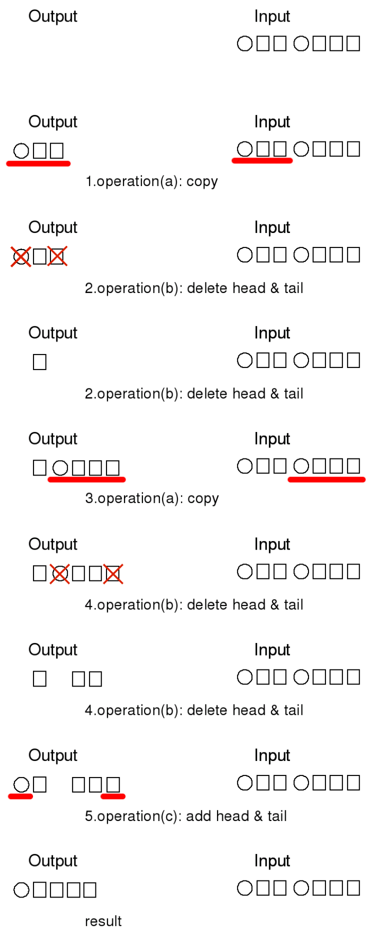

Figure 13: Pop1 program illustrated

Finally, we have a computer, a machine that can compute. At the old time, a computer is a person's job, typically a woman. Although this computer Pop1 has ability to get angry if you mistake the inputs, it does not understand what the number is. It just follows a procedure. However, it "looks like" it can compute. This "looks like" is important. I believe nowadays computer (2009) still does not understand what the number is. Computation is a machine procedure. It looks like computing, and the result is correct.

Appearance and contents, name and meaning are different things. But "formalization" is an idea if all the appearances are the same, let the contents are also the same. This is not a shallow idea. If there are perfect copies, you can not distinguish. Then they are the same. For example, you can buy a software, but each software is a copy. We expect all the software have the same functionality. This is important from the industrial point of view. Of course, I am not ready to accept the exact copy of myself – a copy of human being. But, if I buy two computers of the same model, I expect their functions are the same. If I buy the same DVD title, I want to have the same. Also I expect the quality should not matter if I bought it from a shop in Berlin or in Frankfurt.

On the other hand, I prefer a hand written mail from my friends since each of them is different. I look for the eternal truth in mathematics, but I love also a limited life. It looks like contradiction. I wonder how is the opinion of Marvin, who experienced the universe five times.

9.3 The idea of next level

So far, we saw machines that can compute numbers like SUCC1 or Pop1. Each machine executes own procedures. A procedure is a sequence of instructions, for example, what does the machine do on an input, what does the machine do to create an output. Here we intentionally limited that the machine can only execute one instruction at a time for simplicity, means there is no parallel processing. The examples of instruction are: copy the input number on the input table to the output table, remove head and tail from the number on the output table.

We show the possibility of calculation by a machine which executes a procedure. The concrete examples are SUCC1 and Pop1. We can construct arbitrary machines in the same way. But, do we need to design machines for each problem? To compute addition, do we need Pop1? To compute subtraction, do we need Subtraction1? Or, can we design a little bit more general machine, that one machine can do addition, subtraction, multiplication, and division?

When a mathematician reached this point, s/he can not help thinking: "How many instruction is sufficient to solve all the mathematical problem?" "How can we consider 'all' mathematical problem?" "What can we compute and can not compute in this way?"

We have already talk about a motivation of the λ calculus. The motivation of λ calculus is to think about the foundations of mathematics. λ calculus re-think the mathematics by a formal way. This lead us that calculation can be composed of procedures and that are composed of instructions. Each instruction is performed by a machine. Therefore, calculation can be done by a formal way. It is possible! We saw the possibility. Then people start to ask the next level questions, "how to do that?" Many of the science or mathematics develed in the similar patterns.

It's time to go to the next level of λ calculus.

9.4 Abstraction: Infinite in finite

Here I would like to remind you about **abstraction**. Abstraction derives a common idea that is not associated with any specific instance from many ideas associated with instances. Because this blog is oriented to "getting a feeling of understanding," I need to explain the motivation, "why we should need to mention about **abstraction**?" We introduced the idea, abstraction, because concrete instances are not sufficient. If someone can understand about what is the addition, s/he can add a pair of numbers out from infinite combination of numbers. Because any memory devices have a limit in size, there is a limit to add the numbers in finite combination. It is a strong idea that understanding we can add any number.

We could design a machine that seems compute numbers by memorize the answers. This is a different approach to make machine to compute. We could teach 1 + 1 equals 2, 1 + 2 equals 3. If you ask this machine, what is 1 + 2? The machine can answer 3. But, you asked to the machine, what is $2 + 3$? The machine can not answer the question since the machine has not taught that answer. This is why remembering each instance is not sufficient.

We could abstract 1, 2, 3, ... as a concept of "numbers." We could design a machine which process "numbers." This means a machine can treat infinite kind of number instances.

Here infinity means, we can add "any number" which is quite normal. It should not happens that while $1 + 10$ is possible, $1 + 128$ is not possible. Sometimes, I encounter a person who think that "nothing can do on infinity." However, you can chose one number from infinite numbers and you can add it with one. It might be possible your choice is too large and you can not say that number in your life time. However, you know you can add any two numbers.

I personally think that the most important subject in the high school mathematics is differentiation and integration. It has an idea, "Infinite can live in finite." You can see this through the concept called convergence. For example, there are infinite number of numbers between zero to one. We can project and create one to one mapping from an infinite plane to hemi-sphere with radius one except one point. Pythagoras thought infinite can not live in finite, Zenon made a famous paradox to indicate Pythagoras's mistake.

We could think about infinite numbers by an abstraction of number itself. This is a strong idea. In λ calculus, we abstract functions with three definitions. All three definitions are about functions. Thanks to abstraction, we can handle any functions in λ calculus — infinite type of functions.

Human has an ability to abstract each concrete numbers, like, 1, 2, 3, ... to "(general) numbers." And we can understand that numbers can include all possible numbers. This is outstanding ability of human. It is enough interesting for me that any people have this counting ability and ability of speak except in case of diseases. But a machine seems not have such ability (yet).

Human can easily perform this abstraction. For example, human can have a concept of "chairs." It is still a difficult problem to search pictures of chairs by a machine. Human can see a catalog book of IKEA (a famous furniture shop now in 2009) and can recognize chairs. How can we use it is also easy to be understood. A word "chair" includes infinite type of chairs. But, a human can recognize a new design chair of this year which never existed before.

Similarly, λ calculus does not treat each concrete function. It concerns all kind of functions. We can never predicted what kind of problem we need to solve in the future. However, if we have a theory for any kind of functions and if this concept "function" is enough abstracted, we can use this theory to solve the problem arisen in the future. Mathematics is hard to be obsolete. For example, I doubt the idea of "addition" would be obsolete later in thousand years. The reason we think about this λ calculus or abstraction is we expect the theory last long. I like a knowledge which last long since I do not need to learn all the time. Many study/science aimed this way, but it is not so easy.

Mathematics sticks at infinity, definition, or something mathematics stuffs since we want to expect that what correct today is also correct tomorrow. It is interesting for me that mathematics is the basics of almost all sciences and technology. A study, which does not change once it is established, drives all the world technology change. But if you think a bit more, this is natural. The advancement of technology is based on yesterday's advancement. If this base is changed every day, we could have never advance. The technology itself can change every day. But the base of technology can not be unstable. The basis of the technology is so stable, therefore, we can put some change based on that. This is also the reason that most of primary education includes mathematics.

10 λ, functions, and name

10.1 function = λ

My first problem about λ calculus is that there is only one function, λ. Actually, λ means function, therefore, it is natural to call function as "function" – λ. This sounded strange for me. But λ calculus's function is a bit different from the conventional function when the λ calculus is created. I assume they needed a name to distinguish them.

Anyway, function is λ. I don't know that why the symbol "λ" is selected to represent functions. There are several hypotheses, but we don't exploit it here. One hypothesis is the λ calculus is first started in logic. The letter 'L' in Greek is λ. But actually I don't know, so it is nice way to write that "We don't exploit it here."

My second trouble was why "calculus?" λ calculus seems more like Algebra. In symbolic logic, these are called calculus. However, I am an amateur mathematician, I do not know about this also.

10.2 λ and name

In mathematics, we use function names as f, g, and so force. When we have more than one function, these names f and g are useful to distinguish them. I assume the name f comes from English "function" in English (an old language on a plant called Earth.) But for any function we made, we wrote it as $f :=$ ... except very special functions, for example, $f(x) := x$, $f(x) := x^2$, $f(x) := \sin(x)$, and so on.

It does not matter to write $f :=$ ... or $g :=$ .... The substance part is this ... part, f is just like a tag of a parcel. If we can get a parcel, the tag does not matter.

Every planet has city halls if there are governments. Every planet has banks if there exists money. The people living these planets must wait in a long queue to get the service. This is usually defined by a law. If these services violate the law, i.e., they did not make the people wait, they will be arrested. Surely your planet would be the same. This law seems ubiquitous in our universe. Most of the city hall in these planets issues a number on a card as describing a waiting order. A function name is just like such a number on the card. There is not so much meaning.

I prefer to be called by my name when I am waiting for my turn. But I will be fine, if this call-by-number system makes less waiting time than the call-by-personal-name system. The important thing is this "number" indicates what or who. If "10" indicates me, I am "10." I have no feeling to prefer number 10. I do not say "Oh, the number 10? I do not accept that number." The important thing is "mapping" from the number to me. Therefore, I could accept any number like 10, 42, or 156.

Although I repeated a name is a second matter, mapping from name to substance is essential. I could say a name is for mapping. If you can say several things by one name, it is a great first step of abstraction.

But, λ calculus avoids names. If we do have less names, we can concentrate the substance. We use the name "λ" instead of "function". "λ" seems no meaning, dry, and it seems not a name. On the other hand, a name is important for human understanding process.

Marvin: "What a contradiction! A name is important, but does not matter. I'm depressed."

I mean, mapping is the substance, which is depends on name. Therefore, a name is important "for human," but the name itself can be anything like 10. In this sense, it does not matter. Isn't it clear?

A computer called a "computer" in English, "Keisanki" in Japanese, "Rechner" in German. Even the real substance hardware is the same, it is called by different names. "1" is "one" in English, "Eins" in German, and "ichi" in Japanese. "A rose by any other name would smell as sweet." "One by any other name should share the same concept as 1." Human usually understand a substance by its name. In this sense, the name is important. Some can think this is a philosophical problem. But a computer language called scheme (a lisp) clearly distinguish the substance and its name. The substance is λ, and we can call it directory, or we can bind the name to a λ (mapping!), then call it by the bind name. This is well defined. A machine can perform it. So, I think it is not a problem like "what is life?" Of course, it is difficult to make a map with a good name. Maybe think about a good name is a philosophical problem. (Here I use a stereotype as philosophy == difficult).

11 The first step of λ calculus

The function itself is more important than the name of function. First we need to recognize this is a function or not. In the λ calculus, The symbol λ is used as a marker of a function. Since the name of function is not so important compare to its substance, we should be able to represent the substance of the function without the name. If we need a name, we could make a connection between the name and the substance of the function. (It is called binding.)

Once we used a vending machine as an analogy of a function in this article. Because we can put something into a function, then we can get something out from the function. If you put some money to a vending machine, then you can get some goods from it. I think the simplest vending machine is that if I put something in, then I can get it out without any change. Such function gets a "something" and outputs "something (the same thing I put)." If you input $\circ$, then your output is $\circ$. Let's write (define) such function as $\lambda\ \circ.\circ$. Also we define the first $\circ$ is an input and the second $\circ$ is its output. Here I use a symbol $\circ$, but I choose this arbitrary. I want to say this is something not fixed. This "something" is essential. If I can say this something as a tangible instance, the idea here will be lost. This is a part of abstraction.

Here a vending machine can sell something (or anything). If you limit this machine's ability to only an instance, for example, bottles of water. This machine is just an ordinary machine. It can only handle bottles of water. We want to have more powerful system, therefore we should keep this as "something."

Why do we write the function like that? I think it is quite natural to ask "why" here unless you are a mathematician. Mathematicians know this is the same to a rule of game. Therefore, they understand if you said "it's a rule (It's called a definition)." However, this has some convenience idea behind for mathematicians.

The first symbol "λ" tells you that this is a function. This is a marker or an identifier of a function. We are talking about functions, so, we need to distinguish that this is a function or not. It is possible that we can actually write this marker with "K", or "I will write a function now". In this way, "$\lambda \circ .\circ$" is re-written as "I will write a function now $\circ.\circ$". We also define that the before of "." represents an input and the after the "." is its output. This define is also an artificial rule. For example, we could write the output at the first place and the second one is an input. Or, we can make what is the input/output clear, we can add "this is an input" and "this is the output." In this manner, "$\lambda\ \circ.\circ$" becomes "I will write a function now this is an input $\circ$. this is the output $\circ$."

Because it is cumbersome to write "I will write a function" every time, let's back to the λ, then, "λ this is an input $\circ$. this is the output $\circ$." I am serious to say that "because it is cumbersome." Here if we agree the first one is an input and the last one is the output, then, "$\lambda \circ .\circ$" is sufficient and no misunderstanding. "$\lambda \circ .\circ$" said this is a function, the input is $\circ$ and the output $\circ$.

One thing I am not sure is why λ calculus uses λ as an indicator? But for mathematicians, this does not matter. If we write a function as a "function" literally, or write "1" as "1," it is hard to answer why a function is called function. Why a fall called fall. Maybe this is philosophically question, but I have no work for this. By the way, Mark Twain seems have an idea about the word fall.

If the input is $\circ$ and its output is $\diamond$, we could write this functions as "$\lambda \circ .\diamond$." However, it is ambiguous the relationship between $\circ$ and $\diamond$. The function "$\lambda\circ.\circ$" is rather simple. Let's back to the analogy of vending machine. If you input 10 Euro to the vending machine "$\lambda \circ .\circ$," you will get exact the same 10 Euro. If you input 5 Euro, you will get exact the same 5 Euro. It doesn't matter how much you input to the machine, the output of the machine is always the same amount of your input.

What if we input $\oslash$ to "$\lambda \circ .\diamond$?" In the lambda calculus, if you input a symbol $\oslash$, the output is $\diamond$. This means, "$\lambda \circ .\diamond$", and "$\lambda \oslash .\diamond$" are identical. "$\lambda \circ .\circ$" means if you input "something" then you get "exactly the same something." We don't input $\circ$ only, but something else we can input. It's a bit hard. If you know the idea "variable," we can say "$\circ$" is a variable.

Let's get back to this "$\circ$". "$\circ$" represents "something" here.

Marvin: Again, "something"... I am tired.

You may ask "what is something?" I understand. But the answer is still "something" Or "something of something". For example, if I limit the subject to money, I could say this something is "something of money." If you put "some(thing) of money", exact the same "some(thing) of money" will be out. The function "$\lambda \circ .\circ$" does it. Equation 1 shows the structure of λ expression. The output amount never larger or smaller than the input. Therefore, if you put 100 Altair dollar, the output is 100 Altair dollar. If you put 200 Altair dollar, the output is 200 Altair dollar. If you put 'some' Altair dollar, the output is exact same 'some' Altair dollar. This is the meaning of "something". I can not say more exact since it is abstracted. Some schools teach this "something" as 'x', so some people feel easy to understand as if you input x, then the output is x.

$$\underbrace{\lambda}_{\text{Function}} \underbrace{x}_{\text{Input}} . \underbrace{x}_{\text{Output}} \tag{1}$$

It does not matter if we replace "x" with "$\circ$" or "$\diamond$." The same function we can write as "$\lambda \circ .\circ$," "$\lambda \diamond .\diamond$," "$\lambda x.x$," or "$\lambda y.y$." Conventionally, people wrote this as "$\lambda x.x$." This is the meaning of "Something." If you ask mathematicians, "What is something here?" then they will answer, "something is something." They are not fooling you, that is the best answer they have.

Let's back to the analogy of vending machine again. A simple machine can only accept 100 Altair dollar and can issue 100 Altair dollar ticket for Sirius. This vending machine is simple because it fixes the destination and the price. There were such vending machine on earth is described by Heron of Alexandria, A.D. 10-70.

However, if the machine can accept other destinations and other prices, that would be more versatile. For example, it can also sell 150 Altair dollar ticket for Orion. Not only Altair dollar, but if it accepts a galactic credit card and you can get the ticket for the restaurant at the end of the universe. Or a concert ticket in Kreuzberg. I think you would agree the "something" is more general, this is more useful. The first example of "something" was just 100 Altair dollar. Then it becomes any price of Altair dollar, then becomes credit card. The output started with a ticket for Sirius, then Orion, restaurant and concert. We would like to think about all kind of "something" here, that's the idea of function.

Now I hope you know what the meaning of "something" here. Marvin seems have a comment.

Marvin: "One of my designer developed a machine, called a general exchanger. The difference between an usual exchanger and a general exchanger is the general exchanger accepts anything, and outputs something which has the equal value to the input. (A normal exchanger only exchange money, like 100 Alter dollar to 10000 Sirius yen.) When you put your software, which, by the way, you took more than three months to develop it, the output was an old bread with a cup of cold tea."

I: "That was broken, wasn't it?"

M: "Yes, it was. I can not accept there was a cup of tea."

I: "..."

12 λ calculus: a two times function

Let's write down a function which outputs two times of the input number. The input is 'x,' then the output is '$2x$.' Therefore, we could write it as

$$\lambda x.2x \tag{2}$$

As you see in Equation 1, this is a function and its input is 'x' and the output is '$2x$.' Then this becomes a function which multiples the input with two.

However, this is not exact. Here I cheated you a bit. We are now thinking about the functions. Do we know a function 'multiplication?' We should start with something fundamental, then we would like to develop it. But here we already use an undefined function, 'multiplication.' Let's think again, do we know about numbers? If we did not define numbers like "2," we can not use it. One of my motivation to learn λ calculus was to build a machine which can compute the numbers. Functions like addition, multiplication, subtraction might be easy for human being, but how can a machine know them? We should define each of them, addition, multiplication, and so on.

A function $(\lambda x.2x)$ is $f(x) := 2x$ by the conventional notation. $g(x) := 2x$ also represents the same function. Input is x, then two times of input will be outputted. The names $f(x)$ or $g(x)$ are just an identifier as in the numbers which you might get in your townhall. (By the way, ":=" means "define" here.) Names are usually important for understanding. But if you ask the name is really substance or not, the answer is no. There are no difference between f or g in this example. In the notation of λ expression, we can read $\lambda x.2x$ as "the function which input is x and the output is $2x$." In this way, we do not need to write a name like f or g.

We call a function as Lambda in Lambda calculus all the time. I think that is related with that "the name is not the substance of a function." To concentrate this idea, every function is called λ. It does not matter the function is called as a, b, f, or g. In this way, I could imagine that there are people who are seriously thinking about functions. These people have an idea, they do not want to be bothered by names, but they would like to study what the function is.

13 λ calculus: applying a function

In λ calculus, a function has only one argument. An argument is a parameter of function, or an input of function. We can talk about more than one argument case later.

When the argument value is determined – this means when the input is determined –, put the value to the right side of the λ expression (= function), and apply the function to the value. In the case of vending machine, someone just put the money into it. The machine waits the money. Once some money is in, it computes the output. Many of the functions are similar; They wait an argument. When an argument is determined, the computation starts.

The word "apply" seems a big word. I think this "apply" is not so far from "assign" the value. However, we can assign a non-value, or we can assign another function, so it is better to use the word "apply."

Let's see an example of applying a function.

$$f(x) := 2x \quad (3)$$
$$\lambda x.2x \quad (4)$$

These two functions are the same. Equation 3 is a conventional notation and Equation 4 is a λ expression. Let's assign x to 3, or apply the lambda expression to 3.

$$\begin{aligned} f(3) &:= 2x \\ &= 2 * 3 \\ &= 6 \\ (\lambda x.2x)3 &= (\lambda 3.23) \\ &= 2 * 3 \\ &= 6 \end{aligned}$$

We got the same result. Both functions are a function which multiply the input by two. We inputed three, then we got six.

In lambda calculus, a function can take only one argument. Then how can we handle a two argument function? For example, $f(x, y) := x - y$. This case, we can make a function which take a function with one argument, and the function returns a new function. This is such a function.

$$\lambda y.(\lambda x.x - y)$$

First, let's see the $(\lambda x.x - y)$ part, this is a one argument function, the input is x, and the output is $x - y$. We can apply this to $x = 3$, the result is

$$(\lambda x.x - y)3 = 3 - y$$

But this is a function with an argument y.

$$\lambda y.3 - y$$

Assume $y = 1$,

$$\begin{aligned} (\lambda y.3 - y)1 &= 3 - 1 \\ &= 2 \end{aligned}$$

Good. We have computed $3-1$. As you see, one apply determines one argument. We can repeat this as many as we wish. In this way, we can handle any number of arguments.

The first function $(\lambda x.x-y)$'s result is a function. This might puzzled some people, but this is a powerful tool. But in λ calculus, these function which returns a function is also a function.

Sirius Cybernetics corporation sells a vending machine, which sells other vending machines. Sirius Cybernetics corp.'s advertisement is "General purpose vending machine gensym3141! This machine makes you the top manager of your Konzern. You can sell anything." Of course Marvin said, "That's not possible. Even an earthmen knows it is not possible. How depressed." A vending machine positively can sell a computer or a car. Then nothing wrong if a vending machine sells a vending machine. If a function gets a function as an input and its output is a function, still it is a function. Because a function is getting an input and putting an output. That is a λ.

14 A broken vending machine

A few months ago, there was a question, what if the vending machine is broken? Since I use a vending machine as an analogy of a function, we could also think about a broken function.

First of all, what is "broken" means?

1. If you put anything to the machine, nothing comes out.

2. If you put anything to the machine, the output is always the same.

3. If you put anything to the machine, the output is always unexpected.

If a vending machine behaves one of them, we could say it is "broken." But, a word "broken" is still ambiguous. If the machine always behave one of them, such machine might just fulfill its specification. I would like to say, if the machine could not fulfill the specification, then I define the machine is broken. If we agree with this definition, we can only say a machine is broken or not by looking up the specification. λ expression is enough powerful to express these specifications.

1. Nothing comes out: First we define or interpret the meaning of "nothing comes out." If a vending machine is an exchanger of Altair dollar, "nothing comes out" means 0 Altair dollar comes out. Then, we could write it as $\lambda x.0$. In the same way, if nothing comes out means 0 of something comes out, we could write all of these kind of things.

2. Always the same output: The output is always the same is easy. Let's write the output is y, the same thing every time comes out. This is $\lambda x.y$. This means for any input x, the output is always y.

3. An unexpected output: Again first we need to make this "unexpected" means clear. For example, who expects the output? A person expects the output, I presume. A human being cannot expect the output. But, it might be complicated for a human being. Marvin might be able to expect the output. We can use a function which Marvin is involved. For example, Marvin could figure out the output is a Mersennely twisted. Then we can write a function. The output of a pseudo random number generator seems unexpected, but actually it is just that a human can not recognize it. However, Turing proposed a hardware random number generator, which uses a radioisotope observer machine. In this case, Marvin also has a problem to expect the output (maybe).

15 A vending machine gensym3141

A vending machine "gensym3141" is a product of Sirius Cybernetics corporation. This machine's sales point is that it can generate infinite kind of vending machines. This machine can only provide vending machines, but, you can buy anything – a cap of coffee, a car, or a computer – from gensym3141. If you want to have a cup of coffee, then you first buy a vending machine which sells a cup of coffee.

Here, you must notice that there is a vending machine which gensym3141 can not provide. Some people easily misunderstand that a vending machine can provide infinite kind of vending machine, that can provide everything. If you are a one of them, Marvin will laugh at you. But since laughing at you does not help you, let's think about such machine a bit.

The vending machine gensym3141 has a keypad, you can input a number 1, 2, 3, ... This means you can input infinite kind of numbers to gensym3141. For example, the number of a vending machine of Vogon poetry book is 157079632679489661923. But, the number of vending machine of the fishbowl made by dolphin is -111111. gensym3141 does not have '$-$'

button on the keypad. Therefore, you can input infinite kind of number to the gensym3141, yet you can not get the vending machine of the fish-bowl made by dolphin. Or, it is OK if you can understand infinite does not mean everything.

Of course Sirius Cybernetics corporation marketing people are trained to convince the customers like “You can buy infinite kind of vending machine from gensym3141. Infinite kind! You can buy everything from this one single machine!” Also the universe is huge. There are so many stupid customers who believe these words. Therefore, the universe is filled with gensym3141.

By the way, there are many famous vending machines that you can not buy from gensym3141. First of all, you can not by gensym3141. If someone think about such thing, Sirius Cybernetics's mind control machine, which is integrated in the gensym3141, removes your memory. It is highly recommend not to think about that near the gensym3141. One guy tried to buy a vending machine which sells anti-mind-control vending machines from gensym3141. However, when gensym3141 recognizes the intension of a customer, the mind control machine inside gensym3141 is also activated and the customer becomes a loyal employee of the Sirius Cybernetics corporation.

This is one of the patents of Sirius Cybernetics corporation. You must buy gensym3141 directory from Sirius Cybernetics corporation. We can also not get a vending machine which sells the patents. The details are unknown. There is no record of such sacrifice example.

The publisher of the hitchhiker's guide has a monopoly right of selling the guide. The publisher also has a monopoly right of a monopoly right of selling a vending machine of the guide. Therefore, gensym3141 refuses to sell a vending machine of the guide and also refuses to sell a vending machine of monopoly right of selling the guide. Sirius Cybernetics corporation and the publisher of the guide have a conflict that which has the right to sell a vending machine of a vending machine of monopoly right of selling the guide. This lawsuit took a long time and yet seems no end. Even this lawsuit is finished, it is obvious to see there is the next lawsuit.

It seems we have enough about gensym3141, let's back to the lambda's story.

16 λ version Church number

We have already made Church numbers by boxes. It is about how can we define the numbers for a machine. I wanted to talk about computation. For that, I needed numbers. Without numbers, it is hard to talk about computation. Marvin complains the story line was not natural, but the author's writing skill level was apparently not enough to make it natural.

As we have already introduced Church numbers, I will show you them again.

$$
\begin{aligned}
0 &:= \lambda f x . x \\
1 &:= \lambda f x . f x \\
2 &:= \lambda f x . f(f x) \\
3 &:= \lambda f x . f(f(f x))
\end{aligned}
$$

Please remember, the number of 'f's is corresponds to each number. For example, the Church number 0 has two inputs, f and x, but the output is only one, x, means no f. The Church number 1 has output fx, which contains one f.

$$
\begin{aligned}
0 &:= \underbrace{\lambda}_{\text{Function}} \quad \underbrace{f x}_{\text{Inputs}} . \underbrace{x}_{\text{Output}} \\
1 &:= \underbrace{\lambda}_{\text{Function}} \quad \underbrace{f x}_{\text{Inputs}} . \underbrace{f x}_{\text{Output}}
\end{aligned}
$$

This is like Chinese characters. The Chinese character of 1 is “' 一”, 2 is “' 二”, 3 is “三”. If 4, 5, 6, are in the same way, that is Church numbers. But, the ancient Chinese people had a wisdom and they decided not to use the Church number until a computer will be invented. Since it is not so practical without a computer. Roman numbering system is also similar to the Church number especially when the number is large. By the way, Chinese character's 0 is ' 零.' I do not know when the number 0 was recognized in China. But in Edo era in Japan, 0 is expressed as Tada, means free. When you want to buy something by free, which means 0. We can find this in classic Rakugo, “Kohome.”

17 λ version SUCC 1

An important part of Peano's axiom is the successor function. The successor function is a function to make a successor number of the input number. We described a machine called

"SUCC1" as an implementation of the successor function.

When we define numbers as the answer of what is the numbers, we could define numbers by enumerating all numbers. But, mathematicians are lazy, or must be lazy, and it is quite difficult to enumerate all the numbers which is infinite. Mathematicians' answer is that: 1. create the first number, 2. create a function which generates the next number. Then each mathematician applies them to generate an arbitrary number. They do not use concrete examples 1,2,3, ..., but, they abstract the property of numbers and use the property to define the numbers.

A generator of successor number is a function, therefore it is a λ. If we have the first number and this λ, we can generate all the numbers. The vending machine gensym3141 can provide a vending machine (vending machine No.6931471805), which input is a vending machine and the output is a successor vending machine. The vending machine No.1 is "herring vending machine." The vending machine No.2 is "sandwich vending machine." The vending machine No.3 is "herring sandwich vending machine." If vending machine No.6931471805 gets vending machine No.1, the output is vending machine No.2. In the same way, if it gets vending machine No.2, the output is vending machine No.3. Then what is the output of vending machine No.6931471805, of course it is vending machine No.6931471806. This is just one of a vending machine, however, an intelligent life form usually feel something special on such machine. Therefore there is a name, SUCC. The λ of this SUCC is

$$\text{SUCC} \quad := \quad \lambda nfx.f(nfx).$$

If you read the Wikipedia's λ calculus page, you might try to apply this to numbers. If you can easily get the next number, you do not need to read this article anymore. I failed to do that. I spent for a week to figure it out. When I figured it out, I decided to write this article. This article could be just a supplement of Wikipedia's lambda calculus page.

I forget to explain one rule that function application is "left-associative."

$$fxy = (fx)y$$

The function is processed from left to right. 1 - 2 - 3 is not a λ expression, but it is a left-associative example, means (1 - 2) - 3. Enclosed by enclosed parentheses '()' part is calculated first. Therefore,

$$\begin{aligned} 1 - 2 - 3 &= (1 - 2) - 3 \\ &= -1 - 3 \\ &= -4 \end{aligned}$$

If it is right-associative,

$$\begin{aligned} 1 - 2 - 3 &= 1 - (2 - 3) \\ &= 1 - (-1) \\ &= 2. \end{aligned}$$

We got the different answers. If we write this function as a λ expression,

$$= \quad (\lambda x.\lambda y.\lambda z.x - y - z) \ 1 \ 2 \ 3$$

Because $1 - 2 - 3$ is $f(x, y, z) = x - y - z$ where $x = 1, y = 2, z = 3$. This concludes

$$\begin{array}{lll} (\lambda \underline{x}.\lambda y.\lambda z.x - y - z) \ \underline{1} \ 2 \ 3 & \cdots & \text{apply x to 1} \\ = (\lambda \underline{y}.\lambda z.1 - y - z) \ \underline{2} \ 3 & \cdots & \text{apply y to 2} \\ = (\lambda z.1 - 2 - z) \ 3 & & \\ = (\lambda \underline{z}. - 1 - z) \ \underline{3} & \cdots & \text{apply z to 3} \\ = -1 - 3 & & \\ = -4 & & \end{array}$$

$x = 1, y = 2, z = 3$ if this is left-associative and $x = 3, y = 2, z = 1$ if this is right-associative. We use left-associative in λ calculus unless it is explicitly mentioned.

18 λ version SUCC

18.1 Function SUCC

Last time, we told about a rule, function application is left-associative. First of all, why we need to think about something associative? Because it is not so useful when the answers are different even the expressions are the same. If one computes $1-2-3$ as $(1-2)-3$ $(= -4)$ and the other computes it as $1 - (2 - 3)$ $(= 2)$, the values of the expression are different. Someone might think useful if a calculator gives different answers every time, however, I assume many people could agree such calculator is not so useful. We need to define associativity if the input expressions are the same, then the answer are also the same.

This is the reason why we need to think about associativity. The next question is why we use left-associative? Actually this is just a definition and it does not matter which one we take, as long as it is defined one of them. Someone defined it as left long time ago. This is a definition and it does not have meaning. We could have right-associative system. I assume we use left-associative because many of European languages write the letters from left to right, and nowadays mathematical notation is based on European mathematics.

18.2 Left- and right-associative

A definition, like function application is left-associative, is alike a rule of a sport. One of the rules of basketball is a player with the ball must dribble a ball when the player moves. Why is it? Because it is a rule of the game. If someone dribbles a ball in baseball, it doesn't make any sense. If a soccer player uses hands except the goal-keeper, it is a foul. Why is it so? That is out of question. It is the rules of the game. Mathematics also has the similar aspect. "Why is it left-associative?" "Because it is a definition. (it means that's the rule.)"

An unfortunate starts when someone teaches this mathematical rule is the unique and unalterable truth. If one believe $1 + 1 = 2$ is the unique and unalterable truth, then s/he could not understand mathematics at some point. For example, there is a mathematics $1 + 1 = 1$. This mathematics is used in nowadays computer. Nowadays computer widely uses binary system, and binary system has only '0' and '1.' Then $1 + 1 = 2$ could not be possible since 2 does not exist in one digits computation. Another unfortunate could be someone thinks now $1 + 1 = 2$ is a mistake. This is not a mistake. Each game has each rule or truth like baseball has its own and baseball has one as well. $1 + 1 = 2$ is useful, many cases it is considered truth, but, it is done after it has been defined. Namely, if the rules of a game are defined, that rules become the unique and unalterable truth/rule/definition.

Both in sports and mathematics, we are always interested in a similar aspects. "Does the rule/definition make an interesting game?" I think we do not so much care about the rule or the definition itself. I am much interested in the game, not the rules. If we define an operation of addition, does it lead something interesting? If we think about a system which only uses addition and multiplication, what will it be? By the way, the system of linear algebra consists of addition and one constant multiplication.

When I think about other people, I usually found more interesting about "What this person did?" than "Who is this person? (weight, hight, outlook, name, ...)" I also find that "what is the function of the definition?" is more intriguing than the definition itself. I happen to have a question "Why we need this definition?" for many times. Then I hated mathematics at the moment. But, I think I should not have this question.

"But Marvin, no mathematics teacher taught me such things. Do you know why?"

"Because it is obvious."

If someone understand well, then they did not realized it. Even they did not cast a question. However, I always feel no question is not a good sign.

Let's see the SUCC function again.

$$\text{SUCC} \quad := \quad \lambda\, n\, f\, x.f\ (n\ f\ x)$$

This is a short form of

$$\text{SUCC} \quad := \quad ((\lambda n.\lambda f).\lambda x).f\ (n\ f\ x).$$

The left side of "." indicates function, we could remove two redundant λs out of three λs. More precisely, this is related with "curring." "Curring" coined after Haskell Curry, but it seems Curry is not the inventor (Moses Schoenfinkel and Gottlob Frege). It is the technique of transforming a multiple arguments function to a combination of single argument functions. If you fix all arguments except the first one, you get a function of the non-first arguments in this technique. This means we only need to think about single argument function. Why do we think about curring? Because single argument function is simpler than multiple argument function. If the effects are the same, (lazy) mathematician likes the simpler one.

18.3 SUCC – Apply numbers

Let's compute the SUCC function one by one. First time, I was confused and I tried to use '0' or '1' as a number. But here, a number is a Church number, then 0 is $(\lambda fx.xl)$. You can see I am a totally amateur. Please keep the left-associative in mind,

$$\begin{aligned} \text{SUCC } 0 \quad &:= \quad (\lambda nfx.f(nfx))(\lambda fx.x) \\ &= \quad (\lambda fx.f(\underline{(\lambda fx.x)fx}). \end{aligned}$$

Because of left-associative 'n' is $(\lambda fx.x)$. Here the underlined part,

$$(\lambda fx.x)fx$$

You see that any 'f' will vanish? For example, one argument function

$$(\lambda f.3)$$

means,

$$f(x) := 3.$$

Therefore, this function is always 3 for any 'x.' Such variable 'x' is called free variable (or say, the variable does not bound). It is like Marvin in the Heart of Gold. Anything does not matter for Marvin. Zaphod always forgets him. But this free variable is necessary as Marvin in Hitchhiker's guide to the Galaxy. The vending machine gensym3141 does not bound any variables until you put your credit card. Without credit card, all the variables are free variable for gensym3141. λ function has such a function.

In $(\lambda fx.x)fx$, first, f is applied to f, however, this is a free variable. Then this f is vanished. (This λ expression has only 'x' after the '.'.) If you are confused two fs in this λ expression, we can rewrite it to $(\lambda gx.x)fx$. This is the same. When the function applies to f, g does not bound, then f will vanish.

Then, this will be

$$(\lambda fx.x)fx$$
$$(\lambda x.x)x.$$

The last line, λ expression applies to x, it becomes x. Again, there are three xs here. To make this clear, we could change the variable name to y.

$$(\lambda y.y)x$$

This is totally same as $(\lambda x.x)x$. If we write this in conventional way,

$$\begin{aligned} f(x) &:= x \\ f(y) &:= y. \end{aligned}$$

These two are substantially the same.

Marvin is "マービン" in Japanese. It is "Marvin" in English. But Marvin itself does never change according to which language do you use. Please note that the variable name can be changed does not mean $(\lambda x.x)$ can be $(\lambda x.y)$. The corresponding "what to what" must be kept. This is the line and if you over that line, anyone can treat Vogon correctly.

$$(\lambda fx.x)fx$$

is

$$\begin{aligned} (\lambda \underline{f}x.x)\underline{f}x &\quad \cdots \quad \text{remove } f \text{ since } f \text{ is unbound} \\ (\lambda \underline{x}.x)\underline{x} &\quad \cdots \quad \text{Input } x\text{, output } x \end{aligned}$$

This

$$(\lambda x.x)x$$

is

$$x,$$

Therefore,

$$(\lambda fx.f((\lambda fx.x)fx))$$

is

$$(\lambda fx.f(x)).$$

Although parentheses '()' represent priority of the expression, there is no meaning in this case. Then lazy mathematicians will write it as

$$(\lambda fx.fx)$$

Almost, but not yet. We can remove the outmost parentheses also.

$$\lambda fx.fx$$

Now, this is Church number 1. Therefore, SUCC 0 is 1.

Finally, we have computed SUCC 0 in λ expression way.

19 SUCC 1

Now we are ready to calculate SUCC 1.

$$\begin{aligned} &\text{SUCC } 1 \\ &:= (\lambda \underline{n}fx.f(\underline{n}fx))\underline{(\lambda fx.fx)} \\ &= (\lambda fx.f((\lambda \underline{f}x.\underline{f}x)\underline{f}x)) \\ &= (\lambda fx.f((\lambda \underline{x}.f\underline{x})\underline{x})) \\ &= (\lambda fx.f(fx)) \\ &= \lambda fx.f(fx) \\ &= 2 \end{aligned}$$

SUCC 1 is 2. The underline characters are processed at each line. In case if $((\lambda fx.fx)fx))$ is not clear, we could change the variable name more unique to distinguish. $((\lambda fx.fx)fx)$ is the same to $((\lambda gh.gh)fx)$. Because the meaning of the function does not change if the binding variable name is changed. (more detail, see Wikipedia α-conversion) For example, $f(x) = 2x$ is the same to $f(g) = 2g$. If you draw the graph of this function, the only difference is the axis name is $\{x, f(x)\}$ or $\{g, f(g)\}$.

$$\begin{aligned} ((\lambda gh.gh)fx) &\quad \cdots \quad g \text{ is replaced with } f \\ ((\lambda h.fh)x) &\quad \cdots \quad h \text{ is replaced with } x \\ (fx) & \end{aligned}$$

I hope this is not ambiguous anymore.

20 ADD

Addition is defined by the following.

$$\text{PLUS} := \lambda mnfx.mf(nfx)$$

Let's calculate 1 + 2.

1 and 2 are

$$\begin{aligned} 1 &:= \lambda fx.fx, \\ 2 &:= \lambda fx.f(fx) \end{aligned}$$

respectively.

$$\begin{aligned} & (\lambda \underline{m}nfx.\underline{m}f(nfx))\underline{(\lambda fx.fx)}(\lambda fx.f(fx)) \\ = \; & (\lambda nfx.(\lambda \underline{f}x.\underline{f}x)f(nfx))\underline{(\lambda fx.f(fx))} \\ = \; & (\lambda nfx.(\lambda \underline{x}.fx)\underline{(nfx)})(\lambda fx.f(fx)) \\ = \; & (\lambda \underline{n}fx.f(\underline{n}fx))\underline{(\lambda fx.f(fx))} \\ = \; & (\lambda fx.f((\lambda \underline{f}x.\underline{f}(fx))\underline{f}x)) \\ = \; & (\lambda fx.f((\lambda \underline{x}.f(f\underline{x}))\underline{x})) \\ = \; & (\lambda fx.f(f(fx))) \\ = \; & \lambda fx.f(f(fx)) \\ = \; & 3 \end{aligned}$$

Therefore 1 + 2 = PLUS 1 2 = 3. It seems a magic. But the principle is the same as the Pop1. Church number represents numbers by the number of fs. Therefore, addition is basically concatenate the numbers. If $1 = f$ and $2 = ff$, $1 + 2 = f + ff = fff$. In the same way, for example $3 + 4 = fff + ffff = fffffff = 7$.

21 MULT

Multiplication is defined as the following.

$$\text{MULT} := \lambda mnf.m(nf)$$

Since

$$\begin{aligned} 2 &:= \lambda fx.f(fx) \\ 3 &:= \lambda fx.f(f(fx)) \end{aligned}$$

$$\begin{aligned} & \text{MULT } 2\ 3 \\ := \; & (\lambda \underline{m}nf.\underline{m}(nf))\underline{(\lambda fx.f(fx))} \\ & (\lambda fx.f(f(fx))) \\ = \; & (\lambda nf.(\lambda \underline{f}x.\underline{f}(\underline{f}x))\underline{(nf)})(\lambda fx.f(f(fx))) \end{aligned}$$

You can see that (nf) is copied as many as fs.

$$= \; (\lambda \underline{n}f.(\lambda x.(\underline{n}f)((\underline{n}f)x)))\underline{(\lambda fx.f(f(fx)))}$$

(nf) is copied the first argument (=2) times. The second argument (=3) is replaced with n of (nf).

$$\begin{aligned} = \; & (\lambda f.(\lambda x.((\lambda \underline{f}x.\underline{f}(\underline{f}(\underline{f}x)))\underline{f}) \\ & (((\lambda \underline{f}x.\underline{f}(\underline{f}(\underline{f}x)))\underline{f})x))) \\ = \; & (\lambda f.(\lambda x.(\lambda \underline{x}.f(f(f\underline{x}))) \\ & \underline{(((\lambda x.f(f(fx))))x)))} \\ = \; & (\lambda f.(\lambda x.f(f(f(((\lambda \underline{x}.f(f(f\underline{x}))))\underline{x})x)))) \\ = \; & (\lambda f.(\lambda \underline{x}.f(f(f(f(f(f\underline{x}))))\underline{x})))) \\ = \; & (\lambda f.f(f(f(f(f(fx)))))) \\ = \; & 6 \end{aligned}$$

There is another multiplication definition.

$$\text{MULT} := \lambda mn.m(\text{PLUS}n)0$$

I thought this 0 is just number 0 instead of Church number 0, when I saw this in Wikipedia. This is not easy for beginners.

$$\begin{aligned} & \text{MULT } 2\ 3 \\ := \; & (\lambda \underline{m}n.\underline{m}(\text{PLUS}n)\underline{(\lambda fx.x)}) \\ & (\lambda fx.f(fx))(\lambda fx.f(f(fx))) \\ = \; & (\lambda n.(\lambda \underline{f}x.\underline{f}(\underline{f}x))\underline{(\text{PLUS}n)} \\ & (\lambda fx.x))(\lambda fx.f(f(fx))) \end{aligned}$$

You can find the similar pattern as the first definition, (PLUS n) is copied k-times (k = the first argument number).

$$\begin{aligned}
&= (\lambda \underline{n}.(\lambda x.(\mathrm{PLUS}\underline{n})((\mathrm{PLUS}\underline{n})x)) \\
&\quad \underline{(\lambda fx.x))}(\lambda fx.f(f(fx))) \\
&= (\lambda n.(\mathrm{PLUS}n)((\mathrm{PLUS}n) \\
&\quad (\lambda fx.x)))(\lambda fx.f(f(fx))) \\
&= (\mathrm{PLUS}(\lambda fx.f(f(fx)))) \\
&\quad ((\mathrm{PLUS}(\lambda fx.f(f(fx))))(\lambda fx.x))
\end{aligned}$$

The underline part, PLUS 3 0 is 3 (= 3 + 0). Therefore,

$$= (\mathrm{PLUS}(\lambda fx.f(f(fx))))(\lambda fx.f(f(fx)))$$

This is 3 + 3, then,

$$= 6.$$

22 PRED

We have addition and multiplication. Next I would like to have subtraction. But, we do not have minus numbers of Church numbers. Because we construct the numbers by number of fs. For simplicity, we do not think about minus numbers here. If you want to know more, you can look up Wikipedia.

We prepare PRED before we go to subtraction. The PRED (predecessor) function generates one Church number before the input number. Here we will not think about before the number 0.

The definition of PRED is

$$\mathrm{PRED} := \lambda nfx.n(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u).$$

Let's compute PRED 2.

$$\begin{aligned}
&\mathrm{PRED}\ 2 \\
&:= (\lambda \underline{n}fx.\underline{n}(\lambda gh.h(gf))\underline{(\lambda u.x)} \\
&\quad (\lambda u.u))(\lambda fx.f(fx)) \\
&= (\lambda fx.(\lambda \underline{f}x.\underline{f}(\underline{f}x))\underline{(\lambda gh.h(gf))} \\
&\quad (\lambda u.x)(\lambda u.u)) \\
&= (\lambda fx.(\lambda x.(\lambda gh.h(gf))((\lambda \underline{g}h.h(\underline{g}f))\underline{x})) \\
&\quad (\lambda u.x)(\lambda u.u)) \\
&= (\lambda f\underline{x}.(\lambda x.(\lambda gh.h(gf))(\lambda h.h(xf))) \\
&\quad \underline{(\lambda u.x)}(\lambda u.u)) \\
&= (\lambda fx.(\lambda \underline{g}h.h(\underline{g}f))\underline{(\lambda h.h((\lambda u.x)f))}(\lambda u.u)) \\
&= (\lambda fx.(\lambda h.h((\lambda h.h((\lambda \underline{u}.x)f))\underline{f}))(\lambda u.u))
\end{aligned}$$

Please notice, one f is vanished here. Number of fs represent Church number, therefore, remove one f is minus 1.

$$\begin{aligned}
&= (\lambda fx.(\lambda h.h((\lambda \underline{h}.\underline{h}x)\underline{f}))(\lambda u.u)) \\
&= (\lambda fx.(\lambda h.h((fx)))(\lambda u.u)) \\
&= (\lambda fx.((\lambda u.u)((fx)))) \\
&= (\lambda fx.(fx)) \\
&= (\lambda fx.fx) \\
&= 1
\end{aligned}$$

It is curious to compute PRED 0. What is the predecessor of 0?

$$\begin{aligned}
&\mathrm{PRED}\ 0 \\
&:= (\lambda nfx.n(\lambda gh.h(gf)) \\
&\quad (\lambda u.x)(\lambda u.u))(\lambda fx.x) \\
&= (\lambda fx.(\lambda fx.x)(\lambda gh.h(gf)) \\
&\quad (\lambda u.x)(\lambda u.u)) \\
&= (\lambda fx.(\lambda x.x)(\lambda u.x)(\lambda u.u)) \\
&= (\lambda fx.(\lambda u.x)(\lambda u.u)) \\
&= (\lambda fx.x) \\
&= 0
\end{aligned}$$

It returns 0. This is a nice property of this PRED definition. Because this guarantees the output is always a Church number. If the result of calculation is not defined, it is not so nice. You can find more details of this matter in the field of "group theory." For example, if a vending machine seller machine outputs non-vending machine, it is not nice. Because such machine is no longer a vending machine seller machine.

PRED returns 0 if the input is 0, otherwise it returns one predecessor number.

I am fascinated that Church number itself is part of the program. A number is a program here. Of course Church number is a function. Therefore, it is natural in a sense.

If we apply $\lambda n.n(\lambda gh.h(gf))$ to 0,

$$\begin{aligned}
&\lambda n.n(\lambda gh.h(gf)))(\lambda fx.x) \\
&\quad = \underline{(\lambda fx.x)}(\lambda gh.h(gf)) \\
&\quad = \underline{(\lambda x.x)}
\end{aligned}$$

The output is 0. (Note, this is only a part of PRED function and not a whole expression.) The basic mechanism here is that when this part of the function apples to 0, then the first team is ignored. Since Church number 0 $(\lambda fx.x)$ does not have binding of f.

23 SUB

These series of the examples might be not so fun if you don't try to apply them by yourself. If you do not follow them step by step, these are just a bunch of equations. Therefore, I recommend to try it.

We will define the subtraction using PRED function.

$$\text{SUB} := \lambda mn.n\text{PRED}m$$

Let's compute SUB 3 2 (= 3 - 2).

$$\begin{aligned}
&\text{SUB } 3\ 2 \\
&= (\lambda mn.n\text{PRED}m)(\lambda fx.f(f(fx))) \\
&\quad (\lambda fx.f(fx)) \\
&= (\lambda n.n\text{PRED}(\lambda fx.f(f(fx)))) \\
&\quad (\lambda fx.f(fx)) \\
&= (\lambda fx.f(fx))\text{PRED}(\lambda fx.f(f(fx))) \\
&= (\lambda x.\text{PRED}(\text{PRED}x))(\lambda fx.f(f(fx)))
\end{aligned}$$

You see that the PRED is duplicated as the second argument (= 2). This is the trick. If the second argument (subtrahend) is ten, then PRED is repeated ten times.

$$= \text{PRED}(\text{PRED}(\lambda fx.f(f(fx))))$$

This is PRED (PRED 3).

$$\begin{aligned}
&= \text{PRED2} \\
&= 1
\end{aligned}$$

Therefore, 3 - 2 = 1. We could do subtraction.

24 Conclusion of lambda

Recently I saw this advertisement (Figure 14), "For the finance specialists: Let's start from the simple thing." I assume the simple thing means, $1 + 1 = 2$. I try to explain how to teach $1 + 1$ to a machine in this blog. I took more than eight months and yet not quite complete. (By the way, this is a tobacco advertisement.)

For human beings, this seems simple. But once you want to teach what 1+1 means to a machine, you must know more about it. For example, we discussed what is the numbers, and

Figure 14: An advertisement at Mehringdamm U-Bahn station

we represent it as Church numbers since a machine does not know what the meaning of '1' or '2"s sign. Someone may think this is paranoia since this is so natural.

I believe "natural" does not mean simple. It is just familiar to us. It is not simple at all for me. Some of you might feel it is natural to spend time with your family or your lover. But it is just you are familiar with that, it is not simple thing. It is important for me to see back into the natural things.

I would like to conclude this Hitchhiker's guide to λ calculus at the moment.

One day, I searched lambda calculus in Wikipedia. It said, "it can be verified that PLUS 2 3 and 5 are equivalent lambda expressions" about the PLUS function. However, I did not understand how it works. I needed a large help of my friends. I do not want to forget about this. This was the motivation of writing this blog.

We talked about what is λ calculus, why some people care about that, and several concrete examples. I hope this blog could help the people like me.

But this is not everything about the λ calculus. λ calculus is deep, I am hardly just open the door of this area. I still do not understand combinators. If I could understand it, I would like to continue this blog. I learned that writing is learning or teaching is learning. Also I learned that I could write an article only if I really like it.

I try to keep this article more understandable in informal way. I did neither mention about formal λ expression construction method, nor conversion procedure (α-conversion, β-conversion, and η-conversion). If you wan to know further, Wikipedia [9] would be a good starting point.

25 Summary

$$
\begin{aligned}
0 &:= \lambda fx.x \\
1 &:= \lambda fx.fx \\
2 &:= \lambda fx.f(fx) \\
3 &:= \lambda fx.f(f(fx)) \\
\mathrm{SUCC} &:= \lambda nfx.f(nfx) \\
\mathrm{PLUS} &:= \lambda mnfx.nf(mfx) \\
\mathrm{MULT} &:= \lambda mnf.m(nf) \\
\mathrm{MULT} &:= \lambda mn.m(\mathrm{PLUS}n)0 \\
\mathrm{PRED} &:= \lambda nfx.n(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u) \\
\mathrm{SUB} &:= \lambda mn.n\mathrm{PRED}m
\end{aligned}
$$

Acknowledgments

Thanks to Uchida to give me the cue to write this blog. Thanks to my friends Hoedicke and Rehders to help to understand the examples. Thanks to Maeda who gave me an implementation of Y combinator. This blog is dedicated to my friend Tateoka. I wanted to show him this blog, but unfortunately it is not possible anymore.

References

[1] Mathworld. http://mathworld.wolfram.com/search/?query=peano+theorem&x=0&y=0.

[2] Shitohichi Umaya. http://ameblo.jp/renderrender/.

[3] Shitohichi Umaya. http://shitohichiumaya.blogspot.com/.

[4] Kode Vicious. Pride and Prejudice (The Vasa). *CASM*, 51(9):25–26, 2008.

[5] Wikipedia. http://en.wikipedia.org/wiki/Blaise_Pascal.

[6] Wikipedia. http://en.wikipedia.org/wiki/Difference_engine.

[7] Wikipedia. http://en.wikipedia.org/wiki/Lambda_calculus.

[8] Wikipedia. http://en.wikipedia.org/wiki/Vasa_(ship).

[9] Wikipedia. http://ja.wikipedia.org/wiki/%E3%83%A9%E3%83%A0%E3%83%80%E8%A8%88%E7%AE%97.