Lambda Calculus and The Church-Rosser Theorem

Ke Wu

December 18, 2019

Contents

1 Introduction

"Entscheidungsproblem", or the "Decision Problem", proposed by David Hilbert and Wilhelm Ackermann in 1928, asks for a general algorithm that, when given a set of logic rules and a logical statement, decided whether the given statement is valid (i.e. provable using the given rules or axioms). Formally introduced by Alonzo Church in the 1930s, Lambda Calculus (λ-calculus) is a formal system invented to address this problem, or as it turned out, to demonstrate that this problem cannot be addressed. Along with the Turing machine model (named after its inventor, Alan Turing), they served as the theoretical foundation to a negative answer to the Decision Problem[1].

The Turing machine model continued to thrive as a model of computation, eventually leading to the invention of modern computers and the birth of Computer Science as a new discipline. However, λ-calculus seemed to remain only a formalism. Fortunately, the potential of λ-calculus was rediscovered around 1960s, when scientists began to notice the connection between λ-calculus and various areas of study, e.g. functional programming languages, and semantics of natural language. Since then, λ-calculus has been serving as an important theoretical foundation to many functional languages and related research[2].

In this paper, I will provide a general introduction to the pure λ-calculus and a proof to the Church-Rosser Theorem, one of the key theorems related to λ-calculus.

2 Syntax and Semantics of λ-calculus

Consider a simple function, $f(x) = x^2 + 2x + 1$. We can easily ask the question: "What is the value of this polynomial when $x = 1$?" To find out the answer, we simply substitute the value 1 for x in the polynomial, and compute the result (in this case, we have $f(1) = 1^2 + 2 \cdot 1 + 1 = 4$). The core of λ-calculus is to capture this idea of substitution, or to put it more formally, the idea of "function application". In λ-calculus, we can express the polynomial as a λ-term, $\lambda x.(x^2 + 2x + 1)$. Intuitively, the notation "λx" in the expression has a very similar function to $f(x)$ in the polynomial: it tells us what the variable is in the body. I will discuss this concept in more detail later.

We can *apply* this expression to an *argument* to get its value. We write 'Fa" to denote the application of F on the argument a. As with the previous example, the argument is 1, so we have:

$$(\lambda x.(x^2 + 2x + 1))1 \longrightarrow 1^2 + 2 \cdot 1 + 1 = 4$$

In fact, this example doesn't quite demonstrate the actual λ-calculus, since "numbers" as we know them do not actually exist in λ-calculus; instead, they are encoded using functions. A more realistic example of λ-calculus would be the following:

$$(\lambda x.xx)y \longrightarrow yy$$

As ridiculous as this example might look, we can see that the idea of *substitution* is really at the heart of λ-calculus; even with bare symbols, the rule of substitution still holds as we imagine it will. This illustrates the central axiom of λ-calculus: β-reduction. Intuitively, this axiom simply dictates that given a λ-term $(\lambda x.M)$, β-reduction will replace every instance of x (the bound variable) in the expression M with the argument N. I will discuss this axiom in more detail later.

2.1 Syntax

In this section, I will formally introduce λ-calculus.

Definition 2.1. The alphabet of λ-calculus is the set $\{\text{“}\lambda\text{”}, \text{“(”}, \text{“)”}, \text{“.”}\} \cup V$, where V is an infinite set of ***variables***: $\{v_0, v_1, \dots\}$. ***λ-terms*** are words over the alphabet, defined inductively as follows:

1. x is a λ-term for any variable $x \in V$.
2. If M and N are λ-terms, then (MN) is a λ-term. **(Application)**
3. If M is a λ-term, and $x \in V$, then $(\lambda x.M)$ is a λ-term. **(Abstraction)**

Using our previous example, the term $(\lambda x.xx)$ is a λ-term constructed using rule (3), while the term $((\lambda x.xx)y)$ is constructed with rule (2).

To avoid any notational ambiguity, I will adopt the following conventions for the rest of the paper:

1. Lowercase letters represent arbitrary variables, and uppercase letters represent arbitrary λ-terms.
2. We can omit parentheses in applications by ***left-associativity***, meaning that when we have more than two terms strung together, e.g. $M_1M_2M_3 \dots M_n$, it's interpreted as $(\dots((M_1M_2)M_3)\dots M_n)$.
3. Abstraction, on the other hand, is ***right-associative***, meaning $\lambda x.\lambda y.\lambda z.xyyz \equiv \lambda x.(\lambda y.(\lambda z.xyyz))$. For functions with multiple arguments, i.e. $\lambda x_1.(\lambda x_2.\dots(\lambda x_n.M))$, we can use the shorthand notation $\lambda x_1 \dots x_n.M$.

2.2 Free vs. Bound variables

As promised earlier, I will unveil the mystery of λx now. In an abstraction term, $\lambda x.M$, the use of the λ symbol is to ***bind*** the variable x in the term M. It is very similar to the function of universal and existential quantifiers, $\forall$ and $\exists$, in first-order logic. Formally, we can define the concepts of free and bound variables as follows:

Definition 2.2. ***FV(M)*** is the set of free variables in M, defined by these inductive rules:

1. $FV(x) = \{x\}$
2. $FV(MN) = FV(M) \cup FV(N)$
3. $FV(\lambda x.M) = FV(M) - \{x\}$

A variable is ***bound*** in M if it is not free. Intuitively, if a variable x appears under the scope of a λx, then it is bound.

Example. *Consider the expression $M = \lambda xy.xyz$. What is $FV(M)$?*

Proof. First, let's rewrite M in the form of explicit abstraction: $M \equiv \lambda x.(\lambda y.xyz)$. By rule (1), we have $FV(x) = \{x\}$, $FV(y) = \{y\}$, $FV(z) = \{z\}$. By rule (2), we have $FV(xyz) = FV(x) \cup FV(y) \cup FV(z) = \{x, y, z\}$. By rule (3), we have $FV(\lambda y.xyz) = FV(xyz) - \{y\} = \{x, z\}$. Similarly, $FV(\lambda x.(\lambda y.xyz)) = FV(\lambda y.xyz) - \{x\} = \{z\}$. Therefore, we have $FV(M) = FV(\lambda x.(\lambda y.xyz)) = \{z\}$. It follows naturally that the bound variables in M are $\{x, y\}$. $\square$

2.3 Substitution

Before introducing the formal theory of λ-calculus, we need to understand how substitution works for λ-terms.

Definition 2.3. We write $M[x \coloneqq N]$ to represent the function substituting all *free* occurrences of x in M with the term N. The precise definition of substitution is defined inductively as follows:

1. $x[x \coloneqq M] = M$
2. $y[x \coloneqq M] = y \ (y \neq x)$
3. $(AB)[x \coloneqq M] = A[x \coloneqq M]B[x \coloneqq M]$
4. $(\lambda x.A)[x \coloneqq M] = \lambda x.A$
5. $(\lambda y.A)[x \coloneqq M] = \lambda y.(A[x \coloneqq M]) \ (y \neq x)$

Rule (1) and (2) are the base cases for substitution, where the term we are dealing with is a single variable. If the variable matches the one we are currently substituting, then we replace the variable with the corresponding expression; if the variable does not match, we do not perform any substitution. Rule (3) dictates that substitution distributes over application. Rule (4) reiterates the fact that we are only substituting all *free* occurrences; if the variable we are substituting is bound by a λ in the term, then we do not perform any substitution. Rule (5) states that if the bound variable is different, then we can perform substitution on the body of the abstraction term.

Example. *Consider the substitution,* $(x(\lambda x.y))[x \coloneqq z]$. *What is the result of this substitution?*

Proof. In this case, we can first apply rule (3), setting $A = x$, $B = \lambda x.y$, $M = z$. By rule (3), we can then consider $A[x \coloneqq M]B[x \coloneqq M]$, i.e. the term becomes $x[x \coloneqq z](\lambda x.y)[x \coloneqq z]$. Rule (1) applies directly to $x[x \coloneqq z]$, from which we get $x[x \coloneqq z] = z$. Since $z \neq x$, rule (4) applies to $(\lambda x.y)[x \coloneqq z]$, from which we get $(\lambda x.y)[x \coloneqq z] = \lambda x.y$. We can combine these two results, and conclude that the final result of the substitution is $z(\lambda x.y)$. □

Now, consider these two λ-terms: $\lambda x.x$ and $\lambda y.y$. The only difference between these two terms is the name of the variable, and very much like mathematical functions, changing the name of the variable should not change the functions themselves. For example, the function $f(x) = 2x + 1$ is exactly the same as $f(y) = 2y + 1$. To capture this natural equivalence through renaming in λ-calculus, we define the concept of α-conversion as follows:

Definition 2.4. (α-conversion) For any $\lambda-$term M, we can perform ***α-conversion*** to get an equivalent $\lambda-$term N by replacing v with v' for all abstraction term $\lambda v.A$ in M, given that v' does not occur in A. We say that terms M and N are ***α-convertible*** if we can perform a series of α-conversion to get from M to N.

Now we have everything we need to finally introduce β-reduction, the central axiom of λ-calculus:

Definition 2.5. (β-reduction) For all λ-terms M, N,

$$(\lambda x.M)N \longrightarrow M[x \coloneqq N]$$

provided that no free variables in N becomes bound after its substitution into M.

Essentially, the extra condition on β-reduction is to make sure that free variables remain free. The importance of this condition can be best illustrated with some examples.

Example. *Consider the λ-term,* $(\lambda x.(\lambda y.x))y$. *What should the substitution result be?*

If we perform the substitution as it is right now, we will have $(\lambda x.(\lambda y.x))y \longrightarrow \lambda y.x[x \coloneqq y]$, and by substitution rule (5) and (1), we have $\lambda y.x[x \coloneqq y] = \lambda y.y$. However, does not satisfy our condition for using β-reduction: we require that no free variables in N becomes bound after the substitution. The variable λ-term y corresponds to N in this case, and we can see that it changes from free to bound after we perform the substitution. The intuition behind why we don't want this to happen is that, although they have the same name, y, the inner one bound by λ is really referring to a *different* variable than the outermost y.

This is where α-conversion comes in rescue. Since it should not matter what name we give to the binding variable, we can simply rename the inner binding expression from $\lambda y.x$ to $\lambda z.x$, i.e. $(\lambda x.(\lambda y.x))y \equiv (\lambda x.(\lambda z.x))y$. Note that since the bound variable did not appear in the body anywhere in the first place, we do not have to perform any further name-changing in the body either. After α-conversion, we can now safely use β-reduction to get the result we would expect: $(\lambda x.(\lambda z.x))y \longrightarrow \lambda z.x[x \coloneqq y] = \lambda z.y$.

2.4 Reduction

However, β-reduction alone is still not quite enough for us to exploit the full potential of λ-calculus. Consider the following example:

Example. *Consider the λ-term, $\lambda x.((\lambda y.y)z)$. Is it true that $\lambda x.((\lambda y.y)z) \longrightarrow \lambda x.z$? Can you prove it?*

Intuitively, we know this statement should be true, since the subterm $(\lambda y.y)z = z$ by our laws of substitution. However, the β-reduction rule as it is now only allows us to perform substitution when we have an application term at the outermost level, i.e. when we have a λ-term M such that $M = (\lambda x.M')N$ for some other λ-term M', N. Since our λ-term is an abstraction term, it does not fit the form required by the rule. This suggests that we are still missing some essential rules.

To solve the problem that arose in the last section, we now define the binary relations $\rightarrow_\beta$ **(single-step β-reduction)** and $\twoheadrightarrow_\beta$ **(multi-step β-reduction)** on λ-terms. Together they will be the complete definition for β-reduction.

Definition 2.6. The binary relations $\rightarrow_\beta$ and $\twoheadrightarrow_\beta$ are defined inductively as follows:

1. (a) $(\lambda x.M)N \rightarrow_\beta M[x \coloneqq N]$
   (b) $M \rightarrow_\beta N \implies ZM \rightarrow_\beta ZN$
   (c) $M \rightarrow_\beta N \implies MZ \rightarrow_\beta NZ$
   (d) $M \rightarrow_\beta N \implies \lambda x.M \rightarrow_\beta \lambda x.N$
2. (a) $M \twoheadrightarrow_\beta M$
   (b) $M \rightarrow_\beta N \implies M \twoheadrightarrow_\beta N$
   (c) $M \twoheadrightarrow_\beta N, N \twoheadrightarrow_\beta L \implies M \twoheadrightarrow_\beta L$

As the name suggests, single-step β-reduction represents one single step of β-reduction, i.e. performing substitution once on the given λ-term. Intuitively, multi-step β-reduction represents a *sequence* of substitutions on the given term.

Now with this more robust definition for β-reduction, we can come back to the example and see how it can be proven in the new system.

Example. *Show that $\lambda x.((\lambda y.y)z) \rightarrow_\beta \lambda x.z$.*

Proof. By rule 1(a), we know that $(\lambda y.y)z \to_\beta y[z := y] = z$. Then, by rule 1(d), let $M = (\lambda y.y)z$, $N = z$, we have $\lambda x.M \to_\beta \lambda x.N$, i.e. $\lambda x.((\lambda y.y)z) \to_\beta \lambda x.z$. □

Definition 2.7.

1. A ***β-redex*** is a term of the form $(\lambda x.M)N$. We call $M[x := N]$ its *contractum*.
2. M ***is a β-normal form (β-nf)*** if it doesn't contain any β-redexes as its subterms.
3. M ***has a β-normal form*** if there exists a β-normal term N and that $M \twoheadrightarrow_\beta N$.

For example, the term $(\lambda x.(\lambda y.y))z$ is a β-redex, and its contractum is $\lambda y.y[x := z]$. Since $(\lambda x.(\lambda y.y))z$ is a β-redex, we know that it is not a β-normal form. However, it does have the β-normal form $\lambda y.y$ after we perform the substitution.

3 Induction on Proof Tree

Before getting to the main theorem of this paper, there is one important detour we need to make. In this section, I will introduce the perspective of seeing proofs as mathematical structures, and performing induction on them. This idea will be key to proving the Church-Rosser Theorem.

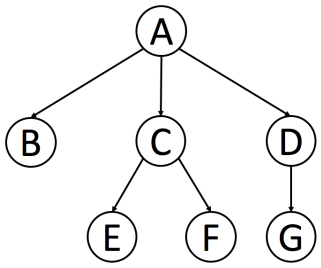

Figure 1: Example Tree

3.1 The World of Trees

Trees. Alveoli of the Earth, home to squirrels and birds, origin of this piece of paper. There are many things we can say about trees, but above all, we don't have enough of them. Unfortunately, those trees (and related environmental concerns) are not the focus of this section. Recall another definition of tree you might have encountered in graph theory:

Definition 3.1. A ***tree*** is a connected acyclic undirected graph. A ***rooted tree*** is a tree where a special node is singled out as the "root". In a rooted tree, the ***parent*** of a node x is the node y connected to x on the path to the root. We also define x as the ***child*** of y. If a node has no child nodes, we call it a ***leaf node***.

Example. *In figure 1, if A is the root node, which nodes are the leaf nodes? Which node(s) have C as their parent? Which node is the child node of D?*

The leaf nodes are B, E, F, and G. And E, F have C as their parent node. Finally, G is the child node of D.

3.2 Induction on Trees

There are a few other key terms about trees that we need to introduce:

Definition 3.2. The ***height*** of a node u is the length of the longest path from u to any leaf. A leaf node has height 0. The height of a tree is defined as the height of its root node.

Example. *What is the height of the node C in figure 1? What about the height of the tree?*

Since the longest path length from C to any leaf node (E or F) is 1, we know C has height 1. Since A is the root node, the height of the tree is 2.

Since the root node always has the biggest height, and that all child nodes will have a lesser height than their parent node, we can in fact perform mathematical induction on the height of the tree. I will demonstrate by the following example.

Example. *Let T be a tree in which every node can have at most 2 child nodes, and the height of T is h. What is the maximum number of nodes (denoted as n) we can have in the tree?*

Claim: There can be at most $2^{h+1} - 1$ nodes.

Proof. I will prove the claim by induction on the height of the tree.

Base case: $h = 0$. This suggests that the root node is a leaf node. Therefore, we know the root node is the only node in the tree, thus $n = 1$. Since $2^{0+1} - 1 = 1$, we can conclude $n \leq 2^{h+1} - 1$. Thus the base case holds.

Induction hypothesis: For $h < k$, we have $n \leq 2^{h+1} - 1$.

Inductive Step: $h = k$. In this case, we want to show that this tree has at most $2^{k+1} - 1$ nodes. We know that the root node has height k, and its child node has height $k - 1$. By the induction hypothesis, we know that each child node of the root can have at most $2^{k-1+1} - 1$ nodes in their corresponding subtree. Therefore, this tree have at most $2(2^k - 1) + 1 = 2^{k+1} - 1$ nodes. Thus we have established the claim.

□

3.3 Proof as Tree

Traditionally, when we think about proofs using inference rules, we tend to think of them as linear or sequential. However, we can also represent proofs as a tree structure.

Definition 3.3. A ***Proof Tree*** is a finite tree where each node is a rule, and the subtree rules have conclusions which exactly match the parent's premises. All leaves of a proof tree must be axioms. A tree with a non-axiom leaf is not a proof.

A conventional way to visualize proof trees is using this form:

$$\frac{A \quad B}{C}$$

where A, B are the ***premises*** and C is the ***conclusion***. This is equivalent to the statement: "A and $B \implies C$". Note that in this representation, premises are separated by white spaces.

Unlike the tree in the previous example, proof trees are oriented like an actual "tree", i.e.: the root node is at the bottom, containing the statement we want to prove as its conclusion. The subtrees are the child nodes positioned above the root node, and they represent subproofs that are needed to reach the final conclusion.

Now that we have defined proofs as trees, we can proceed to establish proof by induction on the height of proof trees in the next section.

3.4 Compatibility of β-reduction

One key property of single-step β-reduction and multi-step β-reduction is ***compatibility***; we will define compatibility formally in this section, and prove that both relations satisfy the criteria. We will make use of this property in our proof of the central theorem.

Definition 3.4. A binary relation R on λ-terms is ***compatible*** if it satisfies the following rules for any λ-terms M, R, N:

1. $M\ R\ N \implies (ZM)\ R\ (ZN)$
2. $M\ R\ N \implies (MZ)\ R\ (NZ)$
3. $M\ R\ N \implies \lambda x.M\ R\ \lambda x.N$

A good example of a compatible relation is ***equality*** of λ-terms; if two terms M and M' are equal, then no matter how many abstractions and applications we perform on them (at the same time), the resulting terms will still be equal.

Lemma 1. *$\to_\beta$ and $\twoheadrightarrow_\beta$ are compatible on λ-terms.*

Proof. (Compatibility of $\to_\beta$) First, we can easily show that $\to_\beta$ is compatible by its definition. Rule 1(b), 1(c), and 1(d) of Definition 2.6 correspond to the three conditions of compatibility directly, so we can safely conclude that $\to_\beta$ is a compatible relation. □

Proof. (Compatibility of $\twoheadrightarrow_\beta$) This proof is less straightforward. Essentially, we'd like to show that the following properties hold for $\twoheadrightarrow_\beta$:

1. $M \twoheadrightarrow_\beta N \implies ZM \twoheadrightarrow_\beta ZN$
2. $M \twoheadrightarrow_\beta N \implies MZ \twoheadrightarrow_\beta NZ$
3. $M \twoheadrightarrow_\beta N \implies \lambda x.M \twoheadrightarrow_\beta \lambda x.N$

Let's focus on the first property. Since it is a rule of inference, we can assume the premise, $M \twoheadrightarrow_\beta N$, is true. Therefore, there must exists some proof tree, T, for $M \twoheadrightarrow_\beta N$. We will proceed to prove this rule by *strong induction on the height of T*.

Base case: The height of T is 0. As we discussed before, this indicates that T is an axiom. Since the only axiom we have for $\twoheadrightarrow_\beta$ is $M \twoheadrightarrow_\beta M$ (the identity rule), we know this is the only possible candidate for T. This suggests that $M = N$, and by compatibility of equality, we know that $ZM = ZN$. Then we can apply the identity rule again, and get $ZM \twoheadrightarrow_\beta ZN$. Thus we have shown the base case holds.

Induction Hypothesis: For proof tree with height less than h, $M \twoheadrightarrow_\beta N \implies ZM \twoheadrightarrow_\beta ZN$.

Inductive Step: We want to show that when we have a proof tree with height h for $M \twoheadrightarrow_\beta N$, we can prove $ZM \twoheadrightarrow_\beta ZN$. Since we have more than one rules that will allow us to prove the premise $M \twoheadrightarrow_\beta N$, we will perform a case analysis on all the possible ways to generate the proof for $M \twoheadrightarrow_\beta N$ (the rules are listed in Def 2.6).

1. $M \to_\beta N \implies M \twoheadrightarrow_\beta N$: If this rule is used to generate the proof for $M \twoheadrightarrow_\beta N$, we know that there must exist some proof for the premise, $M \to_\beta N$. Since we've shown $\to_\beta$ is a compatible relation, we know that $M \to_\beta N \implies ZM \to_\beta ZN$. We can then apply rule 2(b) again to get $ZM \twoheadrightarrow_\beta ZN$.
2. $M \twoheadrightarrow_\beta L, L \twoheadrightarrow_\beta N \implies M \twoheadrightarrow_\beta N$: If this rule is used to generate the proof for $M \twoheadrightarrow_\beta N$, we know that there must exist some proofs for the premises, $M \twoheadrightarrow_\beta L$ and $L \twoheadrightarrow_\beta N$. Since both proofs need to be subtrees of our current tree, we know they must have height

less than h. By the induction hypothesis, we know that $ZM \twoheadrightarrow_\beta ZL$ and $ZL \twoheadrightarrow_\beta ZN$. Then, we can apply rule 2(c) to get $ZM \twoheadrightarrow_\beta ZN$.

Since these are the only rules we could have used to generate a proof for $M \twoheadrightarrow_\beta N$, and that we've shown in both cases, we can then produce a proof for $ZM \twoheadrightarrow_\beta ZN$, we can safely conclude that $M \twoheadrightarrow_\beta N \implies ZM \twoheadrightarrow_\beta ZN$.

The proofs of property 2 and 3 are very similar, so I will omit it here for brevity. □

4 The Church-Rosser Theorem

So far we've been dealing with rather simple λ expressions; none of the examples made use of the multi-step β-reduction definition because they never required more than one step of β-reduction. One natural question to ask is: How does our system behave when given a more complicated expression?

Example. *Consider the expression* $(\lambda x.(\lambda y.y)x)z$. *What does it reduce to?*

Ok. Houston, we have a problem. Since there are two β-redexes in this expression, we have more than one way to reduce it: we can either reduce the inner application first (i.e. $y[y \coloneqq x]$), or the outer application (i.e. $(\lambda y.y)x[x \coloneqq z]$). Our system does not include any rule that dictates the order of reduction, so either option is valid. However, you might also notice that no matter which expression we choose to reduce first, once we perform one more reduction, we will end up with the same expression, namely z. It looks like no matter what order of reduction we choose, we will eventually get the same result.

"But is this always true? Does the order of reduction not matter for all expressions?"

I'm glad you asked! It turns out that in Untyped λ-calculus, the order of reduction really does not matter in that all orders can lead to the same conclusion eventually. This is the core of the Church-Rosser Theorem. I will state the theorem formally next, and present Takahashi's proof[3] of this theorem, which is based on the idea of parallel reduction.

Church-Rosser Theorem. *If* $M \twoheadrightarrow_\beta N_1$, $M \twoheadrightarrow_\beta N_2$, *then there exists a term* N_3 *such that* $N_1 \twoheadrightarrow_\beta N_3$ *and* $N_2 \twoheadrightarrow_\beta N_3$.

The Church-Rosser Theorem has many important implications, but the two listed below are the most widely recognized/researched.

1. It establishes the potential for **concurrency** in reduction. This is especially useful in Computer Science; parallelization can greatly improve performance of program.
2. It allows different **reduction strategies** for different expressions. Sometimes one strategy might cause the program to get stuck in an infinite loop; the flexibility offered by the Church-Rosser Theorem allows programs to potentially avoid infinite loops.

The proof of this theorem can be broken down into roughly three parts: first, we want to introduce (yet another!) binary relation on λ-terms, ***parallel reduction***, and establish certain properties that parallel reduction has with respect to single-step β-reduction and multi-step β-reduction. Then, we want to discuss how we will relate a key property of parallel reduction to the Church-Rosser Theorem. Finally, we will demonstrate the proof of that key property, and thus proving the Church-Rosser Theorem.

4.1 Parallel β-Reduction

Definition 4.1. We inductively define $\rightrightarrows_\beta$, the parallel β-reduction, as follows:

1. $x \rightrightarrows_\beta x$
2. $M \rightrightarrows_\beta M' \implies \lambda x.M \rightrightarrows_\beta \lambda x.M'$
3. $M \rightrightarrows_\beta M', N \rightrightarrows_\beta N' \implies MN \rightrightarrows_\beta M'N'$
4. $M \rightrightarrows_\beta M', N \rightrightarrows_\beta N' \implies (\lambda x.M)N \rightrightarrows_\beta M'[x := N']$

The first three rules dictate that in this system, we have $M \rightrightarrows_\beta M$ holds for all λ-term M. The fourth rule is more subtle; when used in tandem with the third rule, it essentially allows us to contract some β-redexes (possibly overlapping) in M simultaneously to obtain M'.

Now, we will relate $\rightrightarrows_\beta$ with $\to_\beta$ and $\twoheadrightarrow_\beta$ with the following lemmas:

Lemma 2. *$M \to_\beta M' \implies M \rightrightarrows_\beta M'$.*

Proof. We will prove this lemma by strong induction on the height of the proof tree of $M \to_\beta M'$.

Base case: The proof tree has height 0. Since the only axiom we have for $\to_\beta$ is Def 2.6 1(a), which is essentially β-reduction, we know that M is a β-redex, and M' is its contractum, i.e. $M \equiv (\lambda x.A)B$, and $M' \equiv A[x := B]$. Since we know $\rightrightarrows_\beta$ includes identity on λ-terms, we can establish the relations $A \rightrightarrows_\beta A$ and $B \rightrightarrows_\beta B$. Then by rule (4), we know that $(\lambda x.A)B \rightrightarrows_\beta x[A := B]$, which is equivalent to $M \rightrightarrows_\beta M'$. Thus we have shown that the base case holds.

Induction Hypothesis: For all proof trees with height less than h, $M \to_\beta M \implies M \rightrightarrows_\beta M'$.

Inductive Step: We want to show that when we have a proof tree with height h for $M \to_\beta M'$, we can prove $M \rightrightarrows_\beta M'$. Since we have more than one rules that will allow us to prove the premise $M \to_\beta M'$, we will perform a case analysis on all the possible ways to generate the proof for $M \to_\beta M'$ (the rules are listed in Def 2.6; I changed the name of the expressions for clarity's sake).

1. $A \to_\beta B \implies CA \to_\beta CB$: If this rule is used to generate the proof for $M \to_\beta M'$, we know that 1) $M \equiv UV$ and $M' \equiv UW$ for some λ-terms U, V, W, and 2) there must exists some proof for the premise, $V \to_\beta W$. Since we know that the proof tree for $V \to_\beta W$ is a subtree of the current proof tree, it must have a lesser height. By identity, we know that $U \rightrightarrows_\beta U$. By the induction hypothesis, we have $V \rightrightarrows_\beta W$. Thus we can apply rule (3) and get $UV \rightrightarrows_\beta UW$, which is equivalent to $M \rightrightarrows_\beta M'$.
2. $A \to_\beta B \implies AC \to_\beta BC$: This proof is very similar to the one above, so I'll omit it here for brevity.
3. $A \to_\beta B \implies \lambda x.A \to_\beta \lambda x.B$: If this rule is used to generate the proof for $M \to_\beta M'$, we know that 1) $M \equiv \lambda x.U$ and $M' \equiv \lambda x.V$ for some λ-terms U, V, and 2) there must exists some proof for the premise, $U \to_\beta V$. Since we know that the proof tree for $U \to_\beta V$ is a subtree of the current proof tree, it must have a lesser height. By the induction hypothesis, we have $U \rightrightarrows_\beta V$. Thus we can apply rule (2) and get $\lambda x.U \rightrightarrows_\beta \lambda x.V$, which is equivalent to $M \rightrightarrows_\beta M'$.

Since these are the only rules we could have used to generate a proof for $M \to_\beta M'$, and that we've shown in all cases, we can then produce a proof for $M \rightrightarrows_\beta M'$, we can safely conclude that $M \to_\beta M' \implies M \rightrightarrows_\beta M'$. □

Lemma 3. $M \rightrightarrows_\beta M' \implies M \twoheadrightarrow_\beta M'$.

Proof. We will prove this lemma by strong induction on the height of the proof tree of $M \rightrightarrows_\beta M'$.

Base case: The proof tree has height 0. Since the only axiom we have for $\rightrightarrows_\beta$ is rule (1), which is essentially variable identity, we know that $M \equiv x$ and $M = M'$. By Def 2.6 2(a), we have $M \twoheadrightarrow_\beta M'$. Thus we have shown that the base case holds.

Induction Hypothesis: For all proof trees with height less than h, $M \rightrightarrows_\beta M' \implies M \twoheadrightarrow_\beta M'$.

Inductive Step: We want to show that when we have a proof tree with height h for $M \rightrightarrows_\beta M'$, we can prove $M \twoheadrightarrow_\beta M'$. Since we have more than one rules that will allow us to prove the premise $M \rightrightarrows_\beta M'$, we will perform a case analysis on all the possible ways to generate the proof for $M \rightrightarrows_\beta M'$.

1. $A \rightrightarrows_\beta B \implies \lambda x.A \rightrightarrows_\beta \lambda x.B$: If this rule is used to generate the proof for $M \rightrightarrows_\beta M'$, we know that 1) $M \equiv \lambda x.U$ and $M' \equiv \lambda x.V$ for some λ-terms U, V, and 2) there must exists some proof for the premise, $U \rightrightarrows_\beta V$. Since we know that the proof tree for $U \rightrightarrows_\beta V$ is a subtree of the current proof tree, it must have a lesser height. By the induction hypothesis, we have $U \twoheadrightarrow_\beta V$. Since we know $\twoheadrightarrow_\beta$ is compatible, we can conclude that $\lambda x.U \twoheadrightarrow_\beta \lambda x.V$, which is equivalent to $M \twoheadrightarrow_\beta M'$.
2. $A \rightrightarrows_\beta A', B \rightrightarrows_\beta B' \implies AB \rightrightarrows_\beta A'B'$: If this rule is used to generate the proof for $M \rightrightarrows_\beta M'$, we know that 1) $M \equiv UV$ and $M' \equiv U'V'$ for some λ-terms U, V, U', V', and 2) there must exists some proofs for the premises, $U \rightrightarrows_\beta U'$ and $V \rightrightarrows_\beta V'$. Since we know that the proof tree for $U \rightrightarrows_\beta U'$ and $V \rightrightarrows_\beta V'$ are subtrees of the current proof tree, they must have lesser heights. By the induction hypothesis, we have $U \twoheadrightarrow_\beta U'$ and $V \twoheadrightarrow_\beta V'$. Since $\twoheadrightarrow_\beta$ is compatible, we can first use the fact $U \twoheadrightarrow_\beta U'$ to get $UV \twoheadrightarrow_\beta U'V$. Similarly, we can use $V \twoheadrightarrow_\beta V'$ to get $U'V \twoheadrightarrow_\beta U'V'$. Then by transitivity of $\twoheadrightarrow_\beta$ (as defined in Def 2.6 2(c)), we can conclude that $UV \twoheadrightarrow_\beta U'V'$, which is equivalent to $M \twoheadrightarrow_\beta M'$.
3. $A \rightrightarrows_\beta A', B \rightrightarrows_\beta B' \implies (\lambda x.A)B \rightrightarrows_\beta A'[x \coloneqq B']$: If this rule is used to generate the proof for $M \rightrightarrows_\beta M'$, we know that 1) $M \equiv (\lambda x.U)V$ and $M' \equiv U'[x \coloneqq V']$ for some λ-terms U, V, U', V', and 2) there must exists some proofs for the premises, $U \rightrightarrows_\beta U'$ and $V \rightrightarrows_\beta V'$. Since we know that the proof tree for $U \rightrightarrows_\beta U'$ and $V \rightrightarrows_\beta V'$ are subtrees of the current proof tree, they must have lesser heights. By the induction hypothesis, we have $U \twoheadrightarrow_\beta U'$ and $V \twoheadrightarrow_\beta V'$. Since $\twoheadrightarrow_\beta$ is compatible, we can first use the fact $U \twoheadrightarrow_\beta U'$ to get $\lambda x.U \twoheadrightarrow_\beta \lambda x.U'$. Then, we can apply compatibility again to get $(\lambda x.U)V \twoheadrightarrow_\beta (\lambda x.U')V$. Since $V \twoheadrightarrow_\beta V'$, we can also deduce that $(\lambda x.U')V \twoheadrightarrow_\beta (\lambda x.U')V'$. Then, by transitivity of $\twoheadrightarrow_\beta$ (as defined in Def 2.6 2(c)), we know that $(\lambda x.U)V \twoheadrightarrow_\beta (\lambda x.U')V'$. Then axiomatically, we have $(\lambda x.U')V' \to_\beta U'[x \coloneqq V']$. By Def 2.6 2(b), we can turn the $\to_\beta$ to a $\twoheadrightarrow_\beta$, i.e. $(\lambda x.U')V' \twoheadrightarrow_\beta U'[x \coloneqq V']$. Finally, by transitivity of $\twoheadrightarrow_\beta$ again, we have $(\lambda x.U)V \twoheadrightarrow_\beta U'[x \coloneqq V']$, which is equivalent to $M \twoheadrightarrow_\beta M'$.

Since these are the only rules we could have used to generate a proof for $M \rightrightarrows_\beta M'$, and that we've shown in all cases, we can then produce a proof for $M \twoheadrightarrow_\beta M'$, we can safely conclude that $M \rightrightarrows_\beta M' \implies M \twoheadrightarrow_\beta M'$. □

Lemma 4. $M \rightrightarrows_\beta M', N \rightrightarrows_\beta N' \implies M[y \coloneqq N] \rightrightarrows_\beta M'[y \coloneqq N']$.

Proof. We will prove this lemma by strong induction on the height of the proof tree of $M \rightrightarrows_\beta M'$.

Base case: The proof tree has height 0. The only axiom we have for $\rightrightarrows_\beta$ is rule (1), which is essentially variable identity, and thus we know that $M = M'$. We have two cases to consider here:

1. $M \equiv x$ and $x \neq y$. Since the variable does not match the one we are substituting, the rules of substitution dictate that no substitution occurs, i.e. $M[y \coloneqq N] \equiv x[y \coloneqq N] = x$. Now, consider $M'[y \coloneqq N']$; since $M' = M$, we know the result of this substitution will be x as well. By variable identity of $\rightrightarrows_\beta$, we have $x \rightrightarrows_\beta x$, thus $M[y \coloneqq N] \rightrightarrows_\beta M'[y \coloneqq N']$.
2. $M \equiv y$. In this case, the variable matches the one we are currently substituting. By the rules of substitution, we have $M[y \coloneqq N] \equiv y[y \coloneqq N] = N$. Similarly, we can conclude that $M'[y \coloneqq N'] \equiv y[y \coloneqq N'] = N'$. Since we know $N \rightrightarrows_\beta N'$ as part of the premises, we know that $M[y \coloneqq N] \rightrightarrows_\beta M'[y \coloneqq N']$.

Induction Hypothesis: For all proof trees with height less than h, $M \rightrightarrows_\beta M', N \rightrightarrows_\beta N' \implies M[y \coloneqq N] \rightrightarrows_\beta M'[y \coloneqq N']$.

Inductive Step: We want to show that when we have a proof tree with height h for $M \rightrightarrows_\beta M'$, we can prove $M[y \coloneqq N] \rightrightarrows_\beta M'[y \coloneqq N']$. Since we have more than one rule that will allow us to prove the premise $M \rightrightarrows_\beta M'$, we will perform a case analysis on all the possible ways to generate the proof for $M \rightrightarrows_\beta M'$.

1. $A \rightrightarrows_\beta B \implies \lambda x.A \rightrightarrows_\beta \lambda x.B$: If this rule is used to generate the proof for $M \rightrightarrows_\beta M'$, we have two cases here we need to consider:
   (a) $M \equiv \lambda x.U$ and $M' \equiv \lambda x.V$ for some λ-terms U, V, and $x \neq y$. In this case, the substitution $M[y \coloneqq N] \equiv (\lambda x.U)[y \coloneqq N] = \lambda x.(U[y \coloneqq N])$. Similarly, we have $M'[y \coloneqq N'] \equiv (\lambda x.V)[y \coloneqq N'] = \lambda x.(V[y \coloneqq N'])$. We know there must exist some proof for the premise, $U \rightrightarrows_\beta V$, and it will be a subtree of the current proof tree, which means it must have a lesser height. By the induction hypothesis, we have $U[y \coloneqq N] \rightrightarrows_\beta V[y \coloneqq N']$. By rule (2) of Def 4.1, we can conclude $\lambda x.(U[y \coloneqq N]) \rightrightarrows_\beta \lambda x.(V[y \coloneqq N'])$, which is equivalent to $M[y \coloneqq N] \rightrightarrows_\beta M'[y \coloneqq N']$.
   (b) $M \equiv \lambda y.U$ and $M' \equiv \lambda y.V$ for some λ-terms U, V. In this case, since the inner term is "protected" by another λ-binding of the same variable, no substitution takes places here (see rule (4) of Def 2.3), i.e. $M[y \coloneqq N] \equiv (\lambda y.U)[y \coloneqq N] = \lambda y.U$. Similarly, we have $M'[y \coloneqq N'] \equiv (\lambda y.V)[y \coloneqq N'] = \lambda y.V$. Since these are exactly the original terms M, M', and we know that $M \rightrightarrows_\beta M'$, we have shown that $M[y \coloneqq N] \rightrightarrows_\beta M'[y \coloneqq N']$.
2. $A \rightrightarrows_\beta A', B \rightrightarrows_\beta B' \implies AB \rightrightarrows_\beta A'B'$: If this rule is used to generate the proof for $M \rightrightarrows_\beta M'$, we know that 1) $M \equiv UV$ and $M' \equiv U'V'$ for some λ-terms U, V, U', V', and 2) there must exists some proofs for the premises, $U \rightrightarrows_\beta U'$ and $V \rightrightarrows_\beta V'$. In this case, the substitution will yield $M[y \coloneqq N] \equiv UV[y \coloneqq N] = U[y \coloneqq N]V[y \coloneqq N]$. Simialarly, we have $M'[y \coloneqq N'] \equiv U'V'[y \coloneqq N'] = U'[y \coloneqq N']V'[y \coloneqq N']$. Since we know that the proof tree for $U \rightrightarrows_\beta U'$ and $V \rightrightarrows_\beta V'$ are subtrees of the current proof tree, by the induction hypothesis, we have $U[y \coloneqq N] \rightrightarrows_\beta U'[y \coloneqq N']$ and $V[y \coloneqq N] \rightrightarrows_\beta V'[y \coloneqq N']$. By Def 4.1(3), we have $U[y \coloneqq N]V[y \coloneqq N] \rightrightarrows_\beta U'[y \coloneqq N']V'[y \coloneqq N']$, which is equivalent to $M[y \coloneqq N] \rightrightarrows_\beta M'[y \coloneqq N']$.
3. $A \rightrightarrows_\beta A', B \rightrightarrows_\beta B' \implies (\lambda x.A)B \rightrightarrows_\beta A'[x \coloneqq B']$: We need a lemma to support the proof of this case.

   Substitution Lemma: If $x \neq y$ and $x \notin FV(L)$, then:

$$M[x \coloneqq N][y \coloneqq L] \equiv M[y \coloneqq L][x \coloneqq N[y \coloneqq L]]$$

.

Proof. I will prove this lemma by induction on the structure of M.

Base case: $M \equiv x$. There are three cases to consider:
(a) $M = x$: In this case, both sides equal $N[y \coloneqq L]$.
(b) $M = y$: In this case, both sides equal L. Since x is not free in L, we know that the substitution, $L[x \coloneqq N[y \coloneqq L]] = L$.
(c) $M = z, z \neq x, y$: In this case, no substitution happens. Both sides equal z.

Induction hypothesis: For any M' that is a sub-expression of M, we have $M'[x \coloneqq N][y \coloneqq L] \equiv M'[y \coloneqq L][x \coloneqq N[y \coloneqq L]]$.

Inductive Step:
(a) $M \equiv \lambda z.M'$: By induction hypothesis, we have

$$\begin{aligned}(\lambda x.M')[x \coloneqq N][y \coloneqq L] &= \lambda x.M'[x \coloneqq N][y \coloneqq L] \\ &= \lambda x.M'[y \coloneqq L][x \coloneqq N[y \coloneqq L]] \\ &= (\lambda x.M')[y \coloneqq L][x \coloneqq N[y \coloneqq L]] \\ &= M[y \coloneqq L][x \coloneqq N[y \coloneqq L]]\end{aligned}$$

(b) $M \equiv M_1 M_2$: By induction hypothsis, we have

$$\begin{aligned}(M_1M_2)[x \coloneqq N][y \coloneqq L] &= (M_1[x \coloneqq N][y \coloneqq L])(M_2[x \coloneqq N][y \coloneqq L]) \\ &= (M_1[y \coloneqq L][x \coloneqq N[y \coloneqq L]])(M_2[y \coloneqq L][x \coloneqq N[y \coloneqq L]]) \\ &= (M_1M_2)[y \coloneqq L][x \coloneqq N[y \coloneqq L]] \\ &= M[y \coloneqq L][x \coloneqq N[y \coloneqq L]]\end{aligned}$$

Since these cases cover all structures of M, we've proven the substitution lemma. □
Now we can establish the proof for this case. If the rule, $A \rightrightarrows_\beta A', B \rightrightarrows_\beta B' \implies (\lambda x.A)B \rightrightarrows_\beta A'[x \coloneqq B']$, is used to generate the proof for $M \rightrightarrows_\beta M'$, there are again two cases we need to consider:

(a) $M \equiv (\lambda x.U)V$ and $M' \equiv U'[x \coloneqq V']$ for some λ-terms U, V, U', V', $x \neq y$. In this case, the substitution on M will give us $M[y \coloneqq N] \equiv ((\lambda x.U)V)[y \coloneqq N] = (\lambda x.U)[y \coloneqq N]V[y \coloneqq N] = (\lambda x.(U[y \coloneqq N]))V[y \coloneqq N]$. Note that by the definition of substitution, no free variable in N can become bound after the substitution. Therefore, we know that $x \notin FV(N)$.

Since there must exists some proofs for the premises, $U \rightrightarrows_\beta U'$ and $V \rightrightarrows_\beta V'$, by induction hypothesis, we have $U[y \coloneqq N] \rightrightarrows_\beta U'[y \coloneqq N']$ and $V[y \coloneqq N] \rightrightarrows_\beta V'[y \coloneqq N']$. Therefore, by def 4.1(4), we can conclude $(\lambda x.(U[y \coloneqq N]))V[y \coloneqq N] \rightrightarrows_\beta U'[y \coloneqq N'][x \coloneqq V'[y \coloneqq N']]$. By the substitution lemma, we have $U'[y \coloneqq N][x \coloneqq V'[y \coloneqq N]] \equiv U'[x \coloneqq V'][y \coloneqq N']$. Therefore, we can conclude that $(\lambda x.(U[y \coloneqq N]))V[y \coloneqq N] \rightrightarrows_\beta U'[x \coloneqq V'][y \coloneqq N']$. Since we also have $M'[y \coloneqq N'] \equiv U'[x \coloneqq V'][y \coloneqq N']$, we have established $M[y \coloneqq N] \rightrightarrows_\beta M'[y \coloneqq N']$.

(b) $M \equiv (\lambda x.U)V$ and $M' \equiv U'[x \coloneqq V']$ for some λ-terms U, V, U', V', $x = y$: Since x is bound in M and all free instances of x be have been substituted in M', we know that $M[y \coloneqq N] = M$ and $M'[y \coloneqq N'] = M'$. Since we already have $M \rightrightarrows_\beta M'$, we can conclude that $M[y \coloneqq N] \rightrightarrows_\beta M'[y \coloneqq N']$.

Since these are the only rules we could have used to generate a proof for $M \rightrightarrows_\beta M'$, and that we've shown in all cases, we can then produce a proof for $M[y \coloneqq N] \rightrightarrows_\beta M'[y \coloneqq N']$, we can safely conclude that $M \rightrightarrows_\beta M', N \rightrightarrows_\beta N' \implies M[y \coloneqq N] \rightrightarrows_\beta M'[y \coloneqq N']$. □

4.2 All Roads Lead to Church-Rosser

We managed to make the connection between $\to_\beta$, $\twoheadrightarrow_\beta$, and $\rightrightarrows_\beta$. What's next? It turns out that we can "translate" the Church-Rosser Theorem into a property we want to show for $\rightrightarrows_\beta$, and then we can prove that property holds by establishing a more general claim for $\rightrightarrows_\beta$.

Since we know from the last section that $M \rightrightarrows_\beta M' \implies M \twoheadrightarrow_\beta M'$, if we can show that:

$$M \rightrightarrows_\beta N_1, M \rightrightarrows_\beta N_2 \implies N_1 \rightrightarrows_\beta N_3, N_2 \rightrightarrows_\beta N_3 \text{ for some } \lambda\text{-term } N_3$$

We will have established the theorem in $\twoheadrightarrow_\beta$ as well. In fact, there is a strong statement that we can prove in $\rightrightarrows_\beta$ that will give us the above property as a consequence. This statement depends on a definition of M^*, formally defined below:

Definition 4.2. We inductively define M^* as follows:

1. $x^* \equiv x$
2. $(\lambda x.M)^* \equiv \lambda x.M^*$
3. $(M_1 M_2)^* \equiv M_1^* M_2^*$ if $M_1 M_2$ is not a β-redex
4. $((\lambda x.M_1)M_2)^* \equiv M_1^*[x \coloneqq M_2^*]$

Intuitively, M^* is obtained by contracting all the β-redexes in M simultaneously.

Lemma 5. $M \rightrightarrows_\beta N \implies N \rightrightarrows_\beta M^*$

As we can see, this is a stronger statement than the one we purposed earlier, because M^* here only depends on M and not on N. This means for any λ-terms originating from the same term, we can construct some M^* that they can all reduce to.

Proof. Now, we will prove Lemma 5 by strong induction on the height of the proof tree for $M \rightrightarrows_\beta N$.

Base case: The height of the proof tree is 0. Since the only axiom we have in the system is variable identity, we know $M \equiv x$ and $M = N$ for some variable x. By rule 1 of Def 4.2, we know that $M^* \equiv x^* \equiv x$. Then by identity, we have $x \rightrightarrows_\beta x$, which is equivalent to $N \rightrightarrows_\beta M^*$. Thus we have shown the base case holds.

Induction Hypothesis: For all proof tree with height less than h, we have $M \rightrightarrows_\beta N \implies N \rightrightarrows_\beta M^*$.

Inductive Hypothesis: We want to show that when we have a proof tree with height h for $M \rightrightarrows_\beta N$, we can prove $N \rightrightarrows_\beta M^*$. Since we have more than one rule that will allow us to prove the premise $M \rightrightarrows_\beta N$, we will perform a case analysis on all the possible ways to generate the proof for $M \rightrightarrows_\beta N$.

1. $A \rightrightarrows_\beta B \implies \lambda x.A \rightrightarrows_\beta \lambda x.B$: If this rule is used to generate the proof for $M \rightrightarrows_\beta N$, we know that 1) $M \equiv \lambda x.U$ and $N \equiv \lambda x.V$ for some λ-terms U, V, and 2) there must exists some proof for the premise, $U \rightrightarrows_\beta V$. Since we know that the proof tree for $U \rightrightarrows_\beta V$ is a subtree of the current proof tree, it must have a lesser height. By the induction hypothesis, we have $V \rightrightarrows_\beta U^*$. By rule (2) of Def 4.1, we can conclude that $\lambda x.V \rightrightarrows_\beta \lambda x.U^*$. By the definition of M^*, we know that $(\lambda x.U)^* \equiv \lambda x.U^*$. Therefore, we know that $\lambda x.V \rightrightarrows_\beta \lambda x.U^*$ is equivalent to $N \rightrightarrows_\beta M^*$.

2. $A \rightrightarrows_\beta A', B \rightrightarrows_\beta B' \implies AB \rightrightarrows_\beta A'B'$ and AB is not a β-redex: If this rule is used to generate the proof for $M \rightrightarrows_\beta N$, we know that 1) $M \equiv UV$ and $N \equiv U'V'$ for some λ-terms U, V, U', V', and 2) there must exists some proofs for the premises, $U \rightrightarrows_\beta U'$ and $V \rightrightarrows_\beta V'$. Since we know that the proof tree for $U \rightrightarrows_\beta U'$ and $V \rightrightarrows_\beta V'$ are subtrees of the current proof tree, they must have lesser heights. By the induction hypothesis, we have $U' \rightrightarrows_\beta U^*$ and $V' \rightrightarrows_\beta V^*$. By rule (3) of Def 4.1, we know that $U'V' \rightrightarrows_\beta U^*V^*$, and by the definition of M^*, we know that $(UV)^* \equiv U^*V^*$ (we can safely apply this rule because M is not a β-redex), we can conclude that $N \rightrightarrows_\beta M^*$.
3. $A \rightrightarrows_\beta A', B \rightrightarrows_\beta B' \implies AB \rightrightarrows_\beta A'B'$ and AB is a β-redex: If this rule is used to generate the proof for $M \rightrightarrows_\beta N$, we know that 1) $M \equiv (\lambda x.U)V$ and $N \equiv (\lambda x.U')V'$ for some λ-terms U, V, U', V', and 2) there must exists some proofs for the premises, $\lambda x.U \rightrightarrows_\beta \lambda x.U'$ and $V \rightrightarrows_\beta V'$. Since the only rule that will generate the proof $\lambda x.U \rightrightarrows_\beta \lambda x.U'$ has $U \rightrightarrows_\beta U'$ as a premise, we know there must exist some proof for that statement as well. Since we know that the proof tree for $U \rightrightarrows_\beta U'$ and $V \rightrightarrows_\beta V'$ are subtrees of the current proof tree, they must have lesser heights. By the induction hypothesis, we have $U' \rightrightarrows_\beta U^*$ and $V' \rightrightarrows_\beta V^*$. Then by rule (4) of Def 4.1, we have $(\lambda x.U')V' \rightrightarrows_\beta U^*[x \coloneqq V^*]$. By the definition of M^*, we have $((\lambda x.U)V)^* \equiv U^*[x \coloneqq V^*]$. Thus we have shown that $N \rightrightarrows_\beta M^*$.
4. $A \rightrightarrows_\beta A', B \rightrightarrows_\beta B' \implies (\lambda x.A)B \rightrightarrows_\beta A'[x \coloneqq B']$: If this rule is used to generate the proof for $M \rightrightarrows_\beta N$, we know that 1) $M \equiv (\lambda x.U)V$ and $N \equiv U'[x \coloneqq V']$ for some λ-terms U, V, U', V', and 2) there must exists some proofs for the premises, $U \rightrightarrows_\beta U'$ and $V \rightrightarrows_\beta V'$. Since we know that the proof tree for $U \rightrightarrows_\beta U'$ and $V \rightrightarrows_\beta V'$ are subtrees of the current proof tree, they must have lesser heights. By the induction hypothesis, we have $U' \rightrightarrows_\beta U^*$ and $V' \rightrightarrows_\beta V^*$. By Lemma 4, we have $U'[x \coloneqq V'] \rightrightarrows_\beta U^*[x \coloneqq V^*]$. By the definition of M^*, we have $((\lambda x.U)V)^* \equiv U^*[x \coloneqq V^*]$. Therefore, we have shown that $N \rightrightarrows_\beta M^*$.

Since these are the only rules we could have used to generate a proof for $M \rightrightarrows_\beta N$, and that we've shown in all cases, we can then produce a proof for $N \rightrightarrows_\beta M^*$, we can safely conclude that $M \rightrightarrows_\beta N \implies N \rightrightarrows_\beta M^*$. □

This is the complete proof of Lemma 5, thus finishing the proof for the desired property in $\rightrightarrows_\beta$, and consequently the Church-Rosser Theorem in $\twoheadrightarrow_\beta$.

5 References

[1] Henk(Hendrik) Barendregt and E. Barendsen. *Introduction to lambda calculus*. 2000. URL: http://www.cse.chalmers.se/research/group/logic/TypesSS05/Extra/geuvers.pdf (visited on 05/10/2019).

[2] Jesse Alama and Johannes Korbmacher. "The Lambda Calculus". In: *The Stanford Encyclopedia of Philosophy*. Edited by Edward N. Zalta. Spring 2019. Metaphysics Research Lab, Stanford University, 2019.

[3] Masako Takahashi. "Parallel reductions in λ-calculus". In: *Journal of Symbolic Computation* 7.2 (1989), pages 113–123. ISSN: 0747-7171. DOI: https://doi.org/10.1016/S0747-7171(89)80045-8. URL: http://www.sciencedirect.com/science/article/pii/S0747717189800458.