Analytical Differential Calculus with Integration

Han Xu ✉
Peking University, China

Zhenjiang Hu ✉
Peking University, China

Abstract

Differential lambda calculus was first introduced by Thomas Ehrhard and Laurent Regnier in 2003. Despite more than 15 years of history, little work has been done on a differential calculus with integration. In this paper, we shall propose a differential calculus with integration from a programming point of view. We show its good correspondence with mathematics, which is manifested by how we construct these reduction rules and how we preserve important mathematical theorems in our calculus. Moreover, we highlight applications of the calculus in incremental computation, automatic differentiation, and computation approximation.

2012 ACM Subject Classification Software and its engineering → General programming languages; Software and its engineering → Functional languages

Keywords and phrases Differential Calculus, Integration, Lambda Calculus, Incremental Computation, Adaptive Computing

Digital Object Identifier 10.4230/LIPIcs.ICALP.2021.23

1 Introduction

Differential calculus has more than 15 years of history in computer science since the pioneer work by Thomas Ehrhard and Laurent Regnier [9]. It is, however, not well-studied from the perspective of programming languages; we would expect the profound connection of differential calculus with important fields such as incremental computation, automatic differentiation and self-adjusting computation just like how mathematical analysis connects with mathematics. We want to understand what is the semantics of the derivative of a program and how we can use these derivatives to write a program. Technically, we wish to have a clear description of derivatives and introduce integration to compute from operational derivatives to the program.

The two main lines of the work are the differential lambda-calculus [9, 8] and the change theory [7, 3]. On one hand, the differential lambda-calculus uses linear substitution to represent the derivative of a term. For example, given a term $x * x$ (i.e., x^2), with the differential lambda-calculus, we may use the term $\frac{\partial x * x}{\partial x} \cdot 1$ to denote its derivative at 1. As there are two alternatives to substitute 1 for x in the term $x * x$, it gives $(1 * x) + (x * 1)$ (i.e., $2x$) as the derivative (where $+$ denotes "choice"). Despite that the differential lambda-calculus provides a concise way to analyze the alternatives of linear substitution on a lambda term, there is a gap between analysis on terms and computation on terms. For instance, let $+'$ denotes our usual addition operator, we will have $\frac{\partial x +' x}{\partial x} \cdot 1 = (1 +' x) + (x +' 1)$, which is far away from the expected $1 +' 1$. Moreover, it offers no method to integrate over a derivative, say $\frac{\partial t}{\partial x} \cdot y$.

On the other hand, the change theory gives a systematic way to define and propagate (transfer) changes. The main idea is to define the change of function f as *Derive* f, satisfying

$$f(x \oplus \Delta x) = f(x) \oplus (\mathit{Derive}\ f)\ x\ \Delta x.$$

where $\oplus$ denotes an updating operation. It reads that the change over the input x by Δx results in the change over the result of $f(x)$ by $(\mathit{Derive}\ f)\ x\ \Delta x$. While the change theory

448th International Colloquium on Automata, Languages, and Programming (ICALP 2021).
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

provides a general way to describe changes, the changes it described are differences (deltas) instead of derivatives. It is worth noting that derivative is not the same as delta. For example, by the change theory, we can deduce that $f(x)$ will be in the form of $x * x + C$ if we know $(Derive\ f)\ x\ \Delta x = 2 * x * \Delta x + \Delta x * \Delta x$, but we cannot deduce this form if we just know that its derivative is $2 * x$, because the change theory has no concept of integration or limitation.

Although a bunch of work has been done on derivatives [9, 8, 7, 3, 20, 17, 22, 10], there is unfortunately, as far as we are aware, little work on integration. It may be natural to ask what a derivative really means if we can not integrate it. If there is only a mapping from a term to its derivative without its corresponding integration, how can we operate on derivatives with a clear understanding of what we actually have done?

In this paper, we aim at a new differential framework, having dual mapping between derivatives and integrations. With this framework, we can manifest the power of this dual mapping by proving, among others, three important theorems, namely the Newton-Leibniz formula, the Chain Rule and the Taylor's theorem.

Our key idea can be illustrated by a simple example. Suppose we have a function f mapping from an n-dimensional space to an m-dimensional space. Then, let x be $(x_1, x_2, ..., x_n)^T$, and $f(x)$ be $(f_1(x), f_2(x), ..., f_m(x))^T$. Mathematically, we can use a matrix A to represent its derivative, which satisfies the equation

$$f(x + \Delta x) - f(x) = A\Delta x + o(\Delta x),\ \ where\ A\ = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix}$$

However, computer programs usually describe computation over data of some structure, rather than just scalar data or matrix. In this paper, we extend the idea and propose a new calculus that enables us to perform differentiation and integration on data structures. Our main contributions are summarized as follows.

- We have made the first attempt of designing a calculus that provides both derivative and integral. It is an extension of the lambda calculus with five new operators including derivatives and integrations. We give clear semantics and typing rules, and prove that it is sound and strongly normalizing. (Section 2)
- We prove three important theorems and highlight their practical application for incremental computation, automatic differentiation, and computation approximation.
  - We prove the Newton-Leibniz formula: $\int_{t_1}^{t_2} \frac{\partial t}{\partial y}|_x dx = t[t_2/y] \ominus t[t_1/y]$. It shows the duality between derivatives and integrations, and can be used for incremental computation. (Section 3)
  - We prove the Chain Rule: $\frac{\partial f(g\ x)}{\partial x}|_{t_1} * t = \frac{\partial f\ y}{\partial y}|_{g\ t_1} * (\frac{\partial g\ z}{\partial z}|_{t_1} * t)$. It says $\forall x, \forall x_0, (f(g(x)))' * x_0 = f'(g(x)) * g'(x) * x_0$, and can be used for incremental computation and automatic differentiation. (Section 4)
  - We prove the Taylor's Theorem: $f\ t = \sum_{k=0}^{\infty} \frac{1}{k!}(f^{(k)}\ t_0) * (t \ominus t_0)^k$. Different from that one of the differential lambda-calculus [Tho03], this Taylor's theorem manifests results of computation instead of analysis on occurrence of terms. It can be used for approximation of a function computation. (Section 5)

The rest of the paper is organized as follows. Section 2 gives the syntax and semantics of the full calculus, and proves soundness and some useful theorems. Sections 3, 4 and 5

Terms	t	$::=$	c	constants of interpretable type
		$\mid$	x	variable
		$\mid$	$\lambda x : T.\, t$	lambda abstraction
		$\mid$	$t\ t$	function application
		$\mid$	$(t_1, t_2, \ldots, t_n) \mid \pi_j\ t$	n-tuple and projection
		$\mid$	$t \oplus t$	addition
		$\mid$	$t \ominus t$	subtraction
		$\mid$	$t * t$	multiplication
		$\mid$	$\frac{\partial t}{\partial x}\|_t$	derivative
		$\mid$	$\int_t^t t\ dx$	integration
		$\mid$	$inl\ t \mid inr\ t$	left/right injection
		$\mid$	$case\ t\ of\ inl\ x_1 \Rightarrow t \mid inr\ x_2 \Rightarrow t$	case analysis
		$\mid$	$fix\ t$	fix point
Types	T	$::=$	B	base type
		$\mid$	$(T_1, T_2, \ldots, T_n)$	product type
		$\mid$	$T \to T$	function type
		$\mid$	$T + T$	sum type
Contexts	Γ	$::=$	$\emptyset$	empty context
		$\mid$	$\Gamma,\ x : T$	variable binding

Figure 1 Calculus Syntax

correspond to the three main theorems in this paper, and we give a proof and an application for each theorem. Section 6 gives more theorems and applications, Section 7 discusses the related works, and Section 8 concludes the paper.

2 Calculus

In this section, we shall give a clear definition of our calculus with both derivatives and integration. We explain important insights in our design, and prove some useful properties and theorems that will be used later.

2.1 Syntax

Our calculus, as defined in Figure 1, is an extension of the simply-typed lambda calculus [21]. Besides the usual constant, variable, lambda abstraction, function application, and tuple, it introduces five new operations: addition $\oplus$, subtraction $\ominus$, multiplication $*$, derivative $\frac{\partial t}{\partial x}|_t$ and integration $\int_t^t t\ dx$. The three binary operations, namely $\oplus$, $\ominus$, and $*$, are generalizations of those from our mathematics. Intuitively, $x \oplus \Delta$ is for updating x with change Δ, $\ominus$ for canceling updates, and $\otimes$ for distributing updates. We build up terms from terms of base types (such as $\mathbb{R}$, $\mathbb{C}$), and on each base type we require these operations satisfy the following properties:

- The addition and multiplication are associative and commutative, i.e., $(a \oplus b) \oplus c = a \oplus (b \oplus c)$, $a \oplus b = b \oplus a$, $(a * b) * c = a * (b * c)$, $a * b = b * a$.
- The addition and the subtraction are cancellable, i.e., $(a \oplus b) \ominus b = a$ and $(a \ominus b) \oplus b = a$.

$$\frac{c : T \in \Gamma}{\Gamma \vdash c : T}\ (\textsc{TCon}) \qquad \frac{x : T \in \Gamma}{\Gamma \vdash x : T}\ (\textsc{TVar})$$

$$\frac{\Gamma, x : T_1 \vdash t : T_2}{\Gamma \vdash \lambda x : T_1 . t : T_1 \to T_2}\ (\textsc{TAbs}) \qquad \frac{\Gamma \vdash t_1 : T_1 \to T_2 \quad \Gamma \vdash t_2 : T_1}{\Gamma \vdash t_1\ t_2 : T_2}\ (\textsc{TApp})$$

$$\frac{\forall j \in [1,n], \Gamma \vdash t_j : T_j}{\Gamma \vdash (t_1, t_2, ..., t_n) : (T_1, T_2, ..., T_n)}\ (\textsc{TPair}) \qquad \frac{\forall j \in [1,n], \Gamma \vdash t : (T_1, T_2, ..., T_n)}{\Gamma \vdash \pi_j\ t : T_j}\ (\textsc{TProj})$$

$$\frac{\Gamma \vdash t_1 : T^* \quad \Gamma \vdash t_2 : T^*}{\Gamma \vdash t_1 \oplus t_2 : T^*}\ (\textsc{TAdd}) \qquad \frac{\Gamma \vdash t_1 : T^* \quad \Gamma \vdash t_2 : T^*}{\Gamma \vdash t_1 \ominus t_2 : T^*}\ (\textsc{TSub})$$

$$\frac{\Gamma \vdash t_1 : \frac{\partial T^*}{\partial T} \quad \Gamma \vdash t_2 : T}{\Gamma \vdash t_1 * t_2 : T^*}\ (\textsc{TMul1}) \qquad \frac{\Gamma \vdash t_1 : T \quad \Gamma \vdash t_2 : B}{\Gamma \vdash t_1 * t_2 : T}\ (\textsc{TMul2})$$

$$\frac{\Gamma \vdash t : T_1}{\Gamma \vdash \mathit{inl}\ t : T_1 + T_2}\ (\textsc{TInl}) \qquad \frac{\Gamma \vdash t : T_2}{\Gamma \vdash \mathit{inr}\ t : T_1 + T_2}\ (\textsc{TInr})$$

$$\frac{\Gamma \vdash t : T \to T}{\Gamma \vdash \mathit{fix}\ t : T}\ (\textsc{TFix}) \qquad \frac{\Gamma \vdash t_1 : T_1 \quad \Gamma, x : T_1 \vdash t_2 : T_2}{\Gamma \vdash \frac{\partial t_2}{\partial x}|_{t_1} : \frac{\partial T_2}{\partial T_1}}\ (\textsc{TDer})$$

$$\frac{\Gamma \vdash t_1 : T \quad \Gamma \vdash t_2 : T \quad \Gamma, x : T \vdash t : \frac{\partial T^*}{\partial T}}{\Gamma \vdash \int_{t_1}^{t_2} t\ dx : T^*}\ (\textsc{TInt1})$$

$$\frac{\Gamma \vdash t_1 : B \quad \Gamma \vdash t_2 : B \quad \Gamma, x : T \vdash t : T}{\Gamma \vdash \int_{t_1}^{t_2} t\ dx : T}\ (\textsc{TInt2})$$

$$\frac{\Gamma, x_1 : T_1 \vdash t_1 : T \quad \Gamma, x_2 : T_2 \vdash t_2 : T \quad \Gamma \vdash t : T_1 + T_2}{\Gamma \vdash \mathit{case}\ t\ \mathit{of}\ \mathit{inl}\ x_1 \Rightarrow t_1 \mid \mathit{inr}\ x_2 \Rightarrow t_2 : T}\ (\textsc{TCase})$$

Figure 2 Typing Rules

- The multiplication is distributive over addition, i.e., $a * (b \oplus c) = a * b \oplus a * c$.

▶ **Example 1** (Basic Operations on Real Numbers)**.** For real numbers $r_1, r_2 \in \mathbb{R}$, we have

$$\begin{aligned} r_1 \oplus r_2 &= r_1 + r_2 \\ r_1 \ominus r_2 &= r_1 - r_2 \\ r_1 * r_2 &= r_1\ r_2 \end{aligned}$$

We use $\frac{\partial t_1}{\partial x}|_{t_2}$ to denote derivative of t_1 over x at point t_2, and $\int_{t_1}^{t_2} t\ dx$ to denote integration of t over x from t_1 to t_2.

2.2 Typing

As defined in Figure 1, we have base types (denoted by B), tuple types, function types, and sum type. To make our later typing rules easy to understand, we introduce the following type notations:

$$\begin{array}{llll} \text{Type} & T^* & ::= & B & \text{base type} \\ & & \mid & (T^*, T^*, ..., T^*) & \text{product type} \\ & & \mid & T \to T^* & \text{arrow type} \end{array}$$

T^* means the types that are addable (i.e., updatable through $\oplus$). We view the addition between functions, tuples and base type terms as valid, which will be showed by our reduction rules later. But here, we forbid the addition and subtraction between sum types because we view updates such as $inl\ 0 \oplus inr\ 1$ as invalid. If we want to update the change to a term of sum types anyway, we may do case analysis such as $case\ t\ of\ inl\ x_1 \Rightarrow inl\ (x_1 \oplus ...) \mid inr\ x_2 \Rightarrow (x_2 \oplus ...)$.

Next, we introduce notations for derivatives on types:

$$\frac{\partial T}{\partial \mathsf{B}} = T \quad \frac{\partial T}{\partial (T_1, T_2, ..., T_n)} = (\frac{\partial T}{\partial T_1}, \frac{\partial T}{\partial T_2}, ..., \frac{\partial T}{\partial T_n})$$

The first notation says that with the assumption that differences (subtraction) of values of base types are of base types, the derivative over base types has no effect on the result type. And, the second notation resembles partial differentiation. Note that we do not consider derivatives on functions because even for functions on real numbers, there is no good mathematical definition for them yet. Therefore, we do not have a type notation for $\frac{\partial T}{\partial (T_1 \to T_2)}$. Besides, because we forbid the addition and subtraction between the sum types, we will iew the differentiation of the sum types as invalid, so we do not have notations for $\frac{\partial T}{\partial (T_1 + T_2)}$ either.

Figure 2 shows the typing rules for the calculus. The typing rules for constant, variable, lambda abstraction, function application, tuple, and projection are nothing special. The typing rules for addition and subtraction are natural, but the rest three kinds of rules are more interesting. Rule TMul1 and TMul2 show the typing rule for $t_1 * t_2$. If t_1 is a derivative of T_1 over T_2, and t_2 is of type T_2, then multiplication will produce a term of type T_1. This may be informally understood from our familiar equation $\frac{\triangle Y}{\triangle X} * \triangle X = \triangle Y$. Rule TDer shows introduction of the derivative type through a derivative operation, while Rule TInt1 and TInt2 show cancellation of the derivative type through an integration operation.

2.3 Semantics

We will give a two-stage semantics for the calculus. At the first stage, we assume that all the constants (values and functions) over the base types are *interpretable* in the sense there is a default well-defined interpreter to evaluate them. At the second stage, the important part of this paper, we define a set of reduction rules and use the full reduction strategy to compute their normal form, which enjoys good properties of soundness, confluence, and strong normalization.

More specifically, after the full reduction of a term in our calculus, every subterm (now in a normal form of interpretable types) outside the lambda function body will be interpretable on base types, which will be proved in the appendix. In other words, our calculus helps to reduce a term to a normal form which is interpretable on base types, and leave the remaining evaluations to interpretation on base types. We will not give reduction rules to the operations on base types because we do not want to touch on implementations of primitive functions on base types.

For simplicity, in this paper we will assume that the important properties such as the Newton-Leibniz formula, the Chain Rule, and the Taylor's theorem, are satisfied by all the primitive functions and their closures through addition, subtraction, multiplication, derivative and integration. This assumption may seem too strong, since not all primitive functions on base types meet this assumption. However, it would make sense to start with the primitive functions meeting these requirements to build our system, and extend it later with other primitive functions.

2.4 Interpretable Types and Terms

Here, a term is interpretable means it can be directly interpreted by base type interpreter. We use B to denote the base type, over which its constants are interpretable. To make this clear, we define interpretable types as follows.

▶ **Definition 2** (Interpretable Type). *Let B be base types. A type iB is interpretable if it is generated by the following grammar:*

$$\begin{array}{rcll} iB & ::= & \mathsf{B} & \textit{base type} \\ & | & iB \to iB & \textit{function type} \end{array}$$

Constants of interpretable types can be both values or primitive functions of base types. For example, we can use $\sin(x)$, $\cos(x)$, $\mathit{square}(x)$ as primitive functions in our calculus.

Next, we consider terms that are constructed from constants and variables of interpretable types. These terms are interpretable by a default evaluator under an environment mapping variables to constants. Formally, we define the following interpretable terms.

▶ **Definition 3** (Interpretable Terms). *A term is an interpretable if it belongs to it.*

$$\begin{array}{rcll} it & ::= & c & \textit{constants of } iB \\ & | & x & \textit{variable of } iB \\ & | & \lambda x : iB.\, it & \textit{lambda abstraction} \\ & | & it\ it & \textit{function application} \\ & | & it \oplus it & \textit{addition} \\ & | & it \ominus it & \textit{subtraction} \\ & | & it * it & \textit{multiplication} \\ & | & \frac{\partial it}{\partial x}|it & \textit{derivative} \\ & | & \int_{it}^{it} it\ dx & \textit{integration} \end{array}$$

2.4.1 Reduction Rules

Our calculus is an extension of simply-typed lambda calculus. Our lambda abstraction and application are nothing different from the simply-typed lambda calculus, and we have the reduction rule:

$$(\lambda x : T.\, t)t_1 \ \to\ t[t_1/x].$$

We use an n-tuple to model structured data and projection π_j to extract j-th component from a tuple, and we have the following reduction rule:

$$\pi_j(t_1, t_2, ...t_n) \ \to\ t_j.$$

Similarly, we have reduction rules for the case analysis:

$$\mathit{case}\ (\mathit{inl}\ t)\ \mathit{of}\ \mathit{inl}\ x_1 \Rightarrow t_1 \mid \mathit{inr}\ x_2 \Rightarrow t_2 \ \to\ t_1[t/x_1]$$

$$\mathit{case}\ (\mathit{inr}\ t)\ \mathit{of}\ \mathit{inl}\ x_1 \Rightarrow t_1 \mid \mathit{inr}\ x_2 \Rightarrow t_2 \ \to\ t_2[t/x_2]$$

Besides, we introduce fix-point operator to deal with recursion:

$$\mathit{fix}\ f \ \to\ f\ (\mathit{fix}\ f)$$

It is worth noting that tuples, having a good correspondence in mathematics, should be understood as structured data instead of high-dimensional vectors because there are some

$$\frac{t_0 : \mathsf{B}}{\frac{\partial(t_1,t_2,...,t_n)}{\partial x}|_{t_0} \rightarrow (\frac{\partial t_1}{\partial x}|_{t_0}, \frac{\partial t_2}{\partial x}|_{t_0}, ..., \frac{\partial t_n}{\partial x}|_{t_0})} \quad (\text{EAppDer1})$$

$$\frac{t_0 : \mathsf{B}}{\frac{\partial inl/inr\ t}{\partial x}|_{t_0} \rightarrow inl/inr\ \frac{\partial t}{\partial x}|_{t_0}} \quad (\text{EAppDer2})$$

$$\frac{t_0 : \mathsf{B}}{\frac{\partial(\lambda y:T.t)}{\partial x}|_{t_0} \rightarrow \lambda y : T.\frac{\partial t}{\partial x}|_{t_0}} \quad (\text{EAppDer3})$$

$$\frac{\forall i \in [1,n],\ t_{i*} = (t_1, t_2..., t_{i-1}, x_i, t_{i+1}..., t_n)}{\frac{\partial t}{\partial x}|_{(t_1,t_2,...,t_n)} \rightarrow (\frac{\partial t[t_{1*}/x]}{\partial x_1}|_{t_1}, \frac{\partial t[t_{2*}/x]}{\partial x_2}|_{t_2}, ..., \frac{\partial t[t_{n*}/x]}{\partial x_n}|_{t_n})} \quad (\text{EAppDer4})$$

$$\frac{t_1, t_2 : \mathsf{B}}{\int_{t_1}^{t_2}(t_{11}, t_{12}, ...t_{1n})dx \rightarrow (\int_{t_1}^{t_2} t_{11}dx, \int_{t_1}^{t_2} t_{12}dx, ..., \int_{t_1}^{t_2} t_{1n}dx)} \quad (\text{EAppInt1})$$

$$\frac{t_1, t_2 : \mathsf{B}}{\int_{t_1}^{t_2} inl/inr\ t\ dx \rightarrow inl/inr\ \int_{t_1}^{t_2} t\ dx} \quad (\text{EAppInt2})$$

$$\frac{t_1, t_2 : \mathsf{B}}{\int_{t_1}^{t_2} \lambda y : T_2.tdx \rightarrow \lambda y : T_2. \int_{t_1}^{t_2} tdx} \quad (\text{EAppInt3})$$

$$\frac{\forall i \in [1,n], t_{i*} = (t_{21}..., t_{2i-1}, x_i, t_{1i+1}..., t_{1n})}{\int_{(t_{11},t_{12},...t_{1n})}^{(t_{21},t_{22},...,t_{2n})} tdx \rightarrow \int_{t_{11}}^{t_{21}} \pi_1(t[t_{1*}/x])dx_1 \oplus ... \oplus \int_{t_{1n}}^{t_{2n}} \pi_n(t[t_{n*}/x])dx_n} \quad (\text{EAppInt4})$$

Figure 3 Reduction Rules for Derivative and Integration

operations that are different from those in mathematics. As will be seen later, there is difference between our multiplication and matrix multiplication, and derivative and integration on tuples of tuples has no correspondence to mathematical objects.

The core reduction rules in our calculus are summarized in Figure 3, which define three basic cases for both reducing derivative terms and integration terms. For derivative, we use $\frac{\partial t}{\partial x}|_{t_0}$ to denote the derivative of t over x at point t_0, and we have four reduction rules:

- Rule EAppDer1 is to distribute point $t_0 : \mathsf{B}$ into a tuple. This resembles the case in mathematics; if we have a function f defined by $f(x) = (f_1(x), f_2(x), \ldots, f_m(x))^T$, its derivative will be $(\frac{\partial f_1}{\partial x}, \frac{\partial f_2}{\partial x}, \ldots, \frac{\partial f_m}{\partial x})^T$. For example, if we have a function $f : \mathbb{R} \rightarrow (\mathbb{R}, \mathbb{R})$ defined by $f(x) = (x, x * x)$, then its derivative will be $(1, 2 * x)$.
- Rule EAppDer2 is similar to Rule EAppDer1.
- Rule EAppDer3 is to distribute point $t_0 : \mathsf{B}$ into a lambda abstraction. Again this is very natural in mathematics. For example, for function $f(x) = \lambda y : B.x * y$, then we would have its derivative on x as $\lambda y : B.y$.
- Rule EAppDer4 is to deal with partial differentiation, similar to the Jacobian matrix in mathematics (as shown in the introduction). For example, if we have a function that maps a pair (x, y) to $(x*x, x*y \oplus y)$, which may be written as $\lambda z : (\mathsf{B}, \mathsf{B}). (\pi_1 \mathsf{z} * \pi_1 \mathsf{z}, (\pi_1 \mathsf{z} * \pi_2 \mathsf{z} \oplus \pi_2 \mathsf{z}))$ then we would have its derivative $\frac{\partial (f\ z)}{\partial z}|_{(x,y)}$ as $((2 * x, y), (0, x \oplus 1))$.

Similarly, we can define four reduction rules for integration. Rules EAppInt1,EAppInt2 and EAppInt3 are simple. Rule EAppInt4 is worth more explanation. It is designed to establish the Newton-Leibniz formula

$$\int_{t_1}^{t_2} \frac{\partial t}{\partial y}|_x dx = t[t_2/y] \ominus t[t_1/y]$$

$$(t_{11}, ..., t_{1n}) \oplus (t_{21}, ..., t_{2n}) \;\rightarrow\; (t_{11} \oplus t_{21}, ..., t_{1n} \oplus t_{2n}) \quad \text{(EAppAdd1)}$$

$$(\lambda x : T.\, t_1) \oplus (\lambda y : T.\, t_2) \;\rightarrow\; \lambda x : T.\, (t_1 \oplus t_2[y/x]) \quad \text{(EAppAdd2)}$$

$$(t_{11}, ..., t_{1n}) \ominus (t_{21}, ..., t_{2n}) \;\rightarrow\; (t_{11} \ominus t_{21}, ..., t_{1n} \ominus t_{2n}) \quad \text{(EAppSub1)}$$

$$(\lambda x : T.\, t_1) \ominus (\lambda y : T.\, t_2) \;\rightarrow\; \lambda x : T.\, (t_1 \ominus t_2[y/x]) \quad \text{(EAppSub2)}$$

$$\frac{t_0 : \mathsf{B}}{(t_1, t_2, ..., t_n) * t_0 \;\rightarrow\; (t_1 * t_0, t_2 * t_0, ..., t_n * t_0)} \quad \text{(EAppMul1)}$$

$$\frac{t_0 : \mathsf{B}}{(\lambda x : T.t) * t_0 \;\rightarrow\; \lambda x : T.(t * t_0)} \quad \text{(EAppMul2)}$$

$$\frac{t_0 : \mathsf{B}}{(inl/inr\ t) * t_0 \;\rightarrow\; inl/inr\ (t * t_0)} \quad \text{(EAppMul3)}$$

$$\frac{t_1 : (t_{11}, t_{12}, ...t_{1n}), t_2 : (t_{21}, t_{22}, ...t_{2n})}{t_1 * t_2 \;\rightarrow\; (t_{11} * t_{21}) \oplus (t_{12} * t_{22}) \oplus ... \oplus (t_{1n} * t_{2n})} \quad \text{(EAppMul4)}$$

Figure 4 Reduction Rules for Addition, Subtraction and Multiplication

when t_1 and t_2 are tuples:

$$\int_{(t_{11}, t_{12}, ..., t_{1n})}^{(t_{21}, t_{22}, ..., t_{2n})} \frac{\partial t}{\partial y}|_x dx = t[(t_{21}, t_{22}, ..., t_{2n})/y] \ominus t[(t_{11}, t_{12}, ..., t_{1n})/y].$$

So we design the rule to have

$$\int_{t_{1j}}^{t_{2j}} \frac{\partial t[(t_{21}, ..., t_{2(j-1)}, x'_j, t_{1(j+1)}, ..., t_{1n})/y]}{\partial x'_j}|_{x_j} dx_j = \int_{(t_{21}, ..., t_{2(j-1)}, t_{1j}, t_{1(j+1)}, ..., t_{1n})}^{(t_{21}, ..., t_{2(j-1)}, t_{2j}, t_{1(j+1)}, ..., t_{1n})} \frac{\partial t}{\partial y}|_x dx.$$

Notice that under our evaluation rules on derivative, $\pi_j(\frac{\partial t}{\partial x}|_{x=(x_1, x_2, ..., x_n)})$ will be equal to the derivative of t to its j-th parameter x_j, so the integration will lead us to the original t.

Finally, we discuss the reduction rules for the three new binary operations, as summarized in Figure 4. The addition $\oplus$ is introduced to support the reduction rule of integration. It is also useful in proving the theorem and constructing the formula. We can understand the two reduction rules for addition as the addition of high-dimension vectors and functions respectively. Similarly, we can have two reduction rules for subtraction $\ominus$. The operator $*$ was introduced as a powerful tool for constructing the Chain Rule and the Taylor's theorem. The first two reduction rules can be understood as multiplications of a scalar with a function and a high-dimension vector respectively, while the last one can be understood as the multiplication on matrix. For example, we have

$$((1, 4), (2, 5), (3, 6)) * (7, 8, 9) = (50, 122)$$

which corresponds to the following matrix multiplication.

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \begin{pmatrix} 7 \\ 8 \\ 9 \end{pmatrix} = \begin{pmatrix} 50 \\ 122 \end{pmatrix}$$

It is worth noting that while they are similar, $*$ is much different from the matrix multiplication operation. For example, we cannot write x as an m-dimensional vector (or $m * 1$ matrix) in

Taylor's theorem because no matrix A is well-performed under $A * x * x$, but we can write Taylor's Theorem easily under our framework. In the matrix representation, the number of rows of the first matrix and the number of columns of the second matrix must be equal so that we can perform multiplication on them. This means, we can only write case $m = 1$'s Taylor's theorem in matrices, while our version can perform for any tuples.

2.5 Normal Forms

Normal Form	nf	$::=$	nb	normal form on iB
		$\mid$	$(nf, nf, \ldots, nf)$	tuple
		$\mid$	$\lambda x : T.\, t$	function, t cannot be further reduced
		$\mid$	$inl/inr\ nf$	injection
Normal Forms on iB	nb	$::=$	c	constants on iB
		$\mid$	x	variables on iB
		$\mid$	$nb\ nf$	primitive function application
		$\mid$	$nb \oplus nf \ \mid nf \oplus nb$	addition
		$\mid$	$nb \ominus nf \ \mid nf \ominus nb$	subtraction
		$\mid$	$nb * nb$	multiplication
		$\mid$	$\frac{\partial nb}{\partial x}\vert_{nb}$	derivative
		$\mid$	$\int_{nb}^{nb} nb\ dx$	integration

Figure 5 Normal Forms

In our calculus, base type stands in a very special position, and we may involve many evaluations under the context of some free variables of an interpretable type. So for simplicity, we would use full reduction[1] but allow free variables of interpretable types (i.e., iB) in our normal form. Figure 5 defines our normal form. It basically consists of normal form on interpretable types, the tuple normal form, and the function normal form.

We have an interesting result about normal form of a term of interpretable terms.

▶ **Lemma 4** (Interpretability). All the normal forms of terms of interpretable types are interpretable terms. That is, given a term $t : iB$, if t is in normal form, then t is an interpretable term.

Proof. We prove that a normal form t is interpretable by induction on the form of t.

- Case $\lambda x : T.t$. Because $\lambda x : T.t$ is of type iB, T must be of type iB. Notice that the function body t has a free variable x of type iB. By induction, we know t is an interpretable term, therefore, $\lambda x : T.t$ is interpretable.
- Case $(nf, nf, ..., nf)$. This case is impossible, because it is not of type iB. Using the same technique, we can prove the cases for $inl/inr\ nf$.
- Case c. It is an interpretable term itself.
- Case $\frac{\partial nb_1}{\partial x}|_{nb_2}$. By induction, we have both nb_1 and nb_2 are interpretable terms. By definition of interpretable term it, we have $\frac{\partial it}{\partial x}|_{it}$ is an interpretable term. Thus $\frac{\partial nb_1}{\partial x}|_{nb_2}$

[1] By full reduction, we mean that a term can be reduced wherever any of its subterms can be reduced by a reduction rule.

is an interpretable term. Using the same technique, we can prove the cases for $nb * nb$, $\frac{\partial nb}{\partial x}|_{nb}$, and $\int_{nb}^{nb} nb\ dx$.

- Case $nb\ nf$. nb is of type iB, and $nb\ nf$ is of type iB. Thus we can deduce that nf is of type iB. By induction both nb and nf are interpretable terms. Thus $nb\ nf$ is an interpretable term. Using the same technique, we can prove the cases for $nb \oplus nf$, $nf \oplus nb$, $nb \ominus nf$, and $nf \ominus nb$.

◀

2.6 Soundness

Next, we prove some properties of our calculus. The proof is rather routine with some small variations.

▶ **Lemma 5** (Progress). Suppose t is a well-typed term, then t is either a normal form or there is some t' such that $t \to t'$.

Proof. The full proof is in the appendix. In the proof, we adopt a little variation that we allow free variables of interpretable types because we may need rule induction on the form of t inside the term like $\int_{t_1}^{t_2} tdx$ or $\frac{\partial t}{\partial x}|_{t_3}$ where t_1, t_2 and t_3 are of base types. The rest is the common practice. ◀

For preservation, we start with the preservation under substitution.

▶ **Lemma 6** (Preservation under substitutions). If $\Gamma, x : S \vdash t : T$ and $\Gamma \vdash s : S$, then we have $\Gamma \vdash t[s/x] : T$.

And then we can prove the preservation lemma.

▶ **Lemma 7** (Preservation). If $\Gamma \vdash t : T$ and $t \to t'$, then $\Gamma \vdash t' : T$.

Proof. The full proof is in the appendix. ◀

2.6.1 Confluence and Strong Normalization

▶ **Definition 8** (Reduction Relation ρ). *A term t_1 has the relation ρ with t_2, denoted by $t_1 \rho t_2$, if and only if t_1 is t_2 or there is a reduction rule appiled on t_1 for one step that turns t_1 into t_2. We write ρ^* as ρ's transitive closure.*

▶ **Lemma 9** (Confluence). One term has at most one normal form.

Proof. We adopt the techniques used in [4]. We first define a binary relation $\twoheadrightarrow$. Then we prove the relation $\twoheadrightarrow$ has the diamond property, and reduction relation ρ satisfies that $\rho \subseteq \twoheadrightarrow \subseteq \rho^*$. The full proof is in the appendix. ◀

▶ **Lemma 10** (Strong Normalization). If we remove the term $fix\ t$, then every term is strongly normalizable.

Proof. We adopt the technique used in [12]. We prove it by induction on types and forms of terms. The full proof is in the appendix. ◀

2.6.2 Term Equality

We need to talk a bit more on equality because we do not consider reduction or calculation on primitive functions. This notion of equality has little to do with our evaluation but has a lot to do with the equality of primitive functions. Using this notion of equality, we can compute the result from completely different calculations. This will be used in our later proof of the three theorems.

Since we have proved the confluence property, we know that every term has at most one normal form after reduction. Thus, we can define our equality based on their normal forms; the equality between unnormalizable terms is undefined.

A term t_1 is said to be equal to a term t_2, if and only if for all free variables $x_1, x_2, ..., x_n$ in t_1 and t_2, $t_1[u_1/x_1, ..., u_n/x_n] = t_2[u_1/x_1, ..., u_n/x_n]$, where u_i is a closed and normalizable term (Here normalizable means it has a normal form).

A closed-term $t_1 = t_2$, if their normal forms n_1 and n_2 have the relation that $n_1 = n_2$, where a normal form n_1 is said to be equal to another normal form n_2, if they satisfy one of the following rules:

- (1) n_1 is a of type iB, then n_2 has to be of the same type, and under the base type interpretation, n_1 is equal to n_2;
- (2) n_1 is $(t_1, t_2, ..., t_n)$, then n_2 has to be $(t_1', t_2', ..., t_n')$, and $\forall j \in [1, n], t_j$ is equal to t_j';
- (3) n_1 is $\lambda x : T.t$, then n_2 has to be $\lambda y : T.t'$ (y can be x), and $n_1\ x$ is equal to $n_2\ x$.
- (4) n_1 is $inl\ t_1'$, then n_2 has to be $inl\ t_2'$, and t_1' is equal to t_2'.
- (5) n_1 is $inr\ t_1'$, then n_2 has to be $inr\ t_2'$, and t_1' is equal to t_2'.

▶ **Lemma 11.** The equality is reflexive, transitive and symmetric for normalizable terms.

Proof. Based on the equality of terms of base types, we can prove it by induction. ◀

▶ **Lemma 12.** The equality is consistent, e.g., we can not prove equality between arbitrary two terms.

Proof. Notice that except for the equality introduced by the base type interpreter, other equality inference all preserve the type. So for arbitrary t_1 of type (B, B) and t_2 of type B, we can not prove equality between them. ◀

▶ **Lemma 13.** The equality is consistent, e.g. we can not prove equality between arbitrary two terms.

Proof. Notice that except for the equality introduced by base type interpreter, other equality inference all preserve the type. So for arbitrary t_1 of type (B, B) and t_2 of type B, we can not prove equality between them. ◀

Next we give some lemmas that will be used later in our proof. It is relatively unimportant to the mainline of our calculus, so we put their proofs in the appendix.

▶ **Lemma 14.** $t_1 \rho^* t_1', t_2 \rho^* t_2', t_1[t_2/x] \rho^* t_1'[t_2'/x]$

▶ **Lemma 15.** $t_1 = t_1', t_2 = t_2'$, then $t_1 \oplus t_2 = t_1' \oplus t_2'$

▶ **Lemma 16.** Let s be a subterm of t. For any s', if $s' = s$, then $t[s'/s] = t$.

▶ **Lemma 17.** $t_1 * (t_2 \oplus t_3) = (t_1 * t_2) \oplus (t_1 * t_3)$

▶ **Lemma 18.** $(t_1 \ominus t_2) \oplus (t_2 \ominus t_3) = t_1 \ominus t_3$

3 Newton-Leibniz's Formula

The first important theorem we will give is the Newton-Leibniz's formula, which ensures the dualities between derivatives and integration. This theorem lays a solid basis for our calculus. Before giving and proving the theorem, as a warmup, let us take a look at a simple calculation example related to derivative and integration.

▶ **Example 19** (Calculation with Derivatives and Integrations). Consider a function f on real numbers, usually defined in mathematics as $f(x,y) = (x+y, x*y, y)$. In our calculus, it is defined as follows.

$$\begin{array}{lcl} f & :: & (\mathbb{R},\mathbb{R}) \to (\mathbb{R},\mathbb{R},\mathbb{R}) \\ f & = & \lambda x : (\mathbb{R},\mathbb{R}).\,(\pi_1(x) \oplus \pi_2(x), \pi_1(x) * \pi_2(x), \pi_2(x)) \end{array}$$

The following shows the calculation of how $\int_{(0,0)}^{(2,3)} \frac{\partial (f\ y)}{\partial y}|_x dx$ comes equal with $f\ y[(2,3)/y] \ominus f\ y[(0,0)/y]$

$$\begin{array}{cl} & \int_{(0,0)}^{(2,3)} \frac{\partial (f\ y)}{\partial y}|_x dx \\ = & \quad \{\text{ Rule EAppInt3 }\} \\ & \int_0^2 \pi_1(\frac{\partial (f\ y)}{\partial y}|_{(x_1,0)})dx_1 \oplus \int_0^3 \pi_2(\frac{\partial (f\ y)}{\partial y}|_{(2,x_2)})dx_2 \\ = & \quad \{\text{ Rule EAppDer3 }\} \\ & \int_0^2 \pi_1(\frac{\partial f(x_1',0)}{\partial x_1'}|_{x_1}, \frac{\partial f(x_1,x_2')}{\partial x_2'}|_{x_2})dx_1 \oplus \int_0^3 \pi_2(\frac{\partial f(x_1',x_2)}{\partial x_1'}|_{x_1}, \frac{\partial f(2,x_2')}{\partial x_2'}|_{x_2})dx_2 \\ = & \quad \{\text{ Projection }\} \\ & \int_0^2 \frac{\partial f(x_1',0)}{\partial x_1'}|_{x_1}dx_1 \oplus \int_0^3 \frac{\partial f(2,x_2')}{\partial x_2'}|_{x_2}dx_2 \\ = & \quad \{\text{ Function Application }\} \\ & \int_0^2 \frac{\partial (x_1'\oplus 0, x_1' * 0, 0)}{\partial x_1'}|_{x_1}dx_1 \oplus \int_0^3 \frac{\partial (2\oplus x_2', 2*x_2', x_2')}{\partial x_2'}|_{x_2}dx_2 \\ = & \quad \{\text{ Rule EAppInt1 }\} \\ & \int_0^2 (\frac{\partial x_1'\oplus 0}{\partial x_1'}|_{x_1}, \frac{\partial x_1' * 0}{\partial x_1'}|_{x_1}, \frac{\partial 0}{\partial x_1'}|_{x_1})dx_1 \oplus \int_0^3 (\frac{\partial 2\oplus x_2'}{\partial x_2'}|_{x_2}, \frac{\partial 2*x_2'}{\partial x_2'}|_{x_2}, \frac{\partial x_2'}{\partial x_2'}|_{x_2})dx_2 \\ = & \quad \{\text{ Lemma 16 }\} \\ & \int_0^2 (1,0,0)dx_1 \oplus \int_0^3 (1,2,1)dx_2 \\ = & \quad \{\text{ Rule EAppInt1 }\} \\ & (2,0,0) \oplus (3,6,3) \\ = & \quad \{\text{ Rule EAppAdd1 }\} \\ & (5,6,3) \end{array}$$

For readability, we substitute the subterm $(\frac{\partial x_1'\oplus 0}{\partial x_1'}|_{x_1}, \frac{\partial x_1' * 0}{\partial x_1'}|_{x_1}, \frac{\partial 0}{\partial x_1'}|_{x_1})$ and $(\frac{\partial 2\oplus x_2'}{\partial x_2'}|_{x_2}, \frac{\partial 2*x_2'}{\partial x_2'}|_{x_2}, \frac{\partial x_2'}{\partial x_2'}|_{x_2})$ for the same subterm $(1,0,0)$ and $(1,2,1)$ during the calculation, though our calculus does not actually perform computation like this. These substitutions are safe to perform (Lemma 16), and give a better demonstration on how the Newton-Leibniz theorem works. And Here we have $\int_{(0,0)}^{(2,3)} \frac{\partial (f\ y)}{\partial y}|_x dx = (5,6,3) = f(2,3) \ominus f(0,0) = f\ y[(2,3)/y] \ominus f\ y[(0,0)/y]$.

◀

▶ **Theorem 20** (Newton-Leibniz). *Let t contain no free occurrence of x, and both $\int_{t_1}^{t_2} \frac{\partial t}{\partial y}|_x dx$ and $t[t_2/y] \ominus t[t_1/y]$ are well-typed and normalizable. Then we have*

$$\int_{t_1}^{t_2} \frac{\partial t}{\partial y}|_x dx = t[t_2/y] \ominus t[t_1/y].$$

Proof. If t_1, t_2 or t is not closed, then we need to prove $\forall u_1, ..., u_n$, we have

$$(\int_{t_1}^{t_2} \frac{\partial t}{\partial y}|_x dx)[u_1/x_1, ..., u_n/x_n] = (t[t_2/y] \ominus t[t_1/y])[u_1/x_1, ..., u_n/x_n].$$

By freezing $u_1, ..., u_n$, we can apply the substitution $[u_1/x_1, ..., u_n/x_n]$ to make every term closed. So, for simplicity, we will assume t, t_1 and t_2 to be closed.

We prove this by induction on types.

- Case: t_1,t_2 and t are of base types. By the confluence lemma, we know there exists the normal form t', t_1' and t_2' of the term t, t_1 and t_2. Also, we know $\int_{t_1}^{t_2} \frac{\partial t}{\partial y}|_x dx = \int_{t_1'}^{t_2'} \frac{\partial t'}{\partial y}|_x dx$ and $t[t_2/y] \ominus t[t_1/y] = t'[t_2'/y] \ominus t'[t_1'/y]$. Since on base types we have $\int_{t_1'}^{t_2'} \frac{\partial t'}{\partial y}|_x dx = t'[t_2'/y] \ominus t'[t_1'/y]$, we have $\int_{t_1}^{t_2} \frac{\partial t}{\partial y}|_x dx = t[t_2/y] \ominus t[t_1/y]$.
- Case: t_1,t_2 are of base types, t is of type $(T_1, T_2, ..., T_n)$. By the confluence lemmas, there exist a normal form $(t_{11}', t_{12}', ..., t_{1n}')$ for t. Using Rules (EAppInt1) and (EAppDer1), we know

$$\begin{array}{rcl} \int_{t_1}^{t_2} \frac{\partial t}{\partial y}|_x dx & = & \int_{t_1}^{t_2} \frac{\partial (t_{11}', t_{12}', ..., t_{1n}')}{\partial y}|_x dx \\ & = & \int_{t_1}^{t_2} (\frac{\partial t_{11}'}{\partial y}|_x, \frac{\partial t_{12}'}{\partial y}|_x, ..., \frac{\partial t_{1n}'}{\partial y}|_x) dx \\ & = & (\int_{t_1}^{t_2} \frac{\partial t_{11}'}{\partial y}|_x dx, \int_{t_1}^{t_2} \frac{\partial t_{12}'}{\partial y}|_x dx, ..., \int_{t_1}^{t_2} \frac{\partial t_{1n}'}{\partial y}|_x dx) \end{array}$$

  On the other hand, we have

$$\begin{array}{l} t[t_2/y] \ominus t[t_1/y] \\ \quad = (t_{11}'[t_2/y], t_{12}'[t_2/y], ..., t_{1n}'[t_2/y]) \ominus (t_{11}'[t_1/y], t_{12}'[t_1/y], ..., t_{1n}'[t_1/y]) \\ \quad = (t_{11}'[t_2/y] \ominus t_{11}'[t_1/y], t_{12}'[t_2/y] \ominus t_{12}'[t_1/y], ..., t_{1n}'[t_2/y] \ominus t_{1n}'[t_1/y]) \end{array}$$

  By induction, we have $\forall j \in [1, n], \int_{t_1}^{t_2} \frac{\partial t_{1j}'}{\partial y}|_x dx = t_{1j}'[t_2/y] \ominus t_{1j}'[t_1/y]$, so we prove the case.
- Case: t_1,t_2 are of base types, t is of type $A \to B$. By Lemma 16, we can use $\lambda z : A.t\ z$ (for simiplicity, we use $\lambda z : A.t'$ where $t' = t\ z$) to substitute for t, where z is a fresh variable. Now, we have for any u,

$$\begin{array}{rcl} (\int_{t_1}^{t_2} \frac{\partial t}{\partial y}|_x dx)\ u & = & (\int_{t_1}^{t_2} \frac{\partial \lambda z:A.t'}{\partial y}|_x dx)\ u \\ & = & \lambda z : A.(\int_{t_1}^{t_2} \frac{\partial t'}{\partial y}|_x dx)\ u \\ & = & \int_{t_1}^{t_2} \frac{\partial t'[u/z]}{\partial y}|_x dx \end{array}$$

  and on the other hand, since z is free in t_1 and t_2, we have

$$\begin{array}{rcl} (t[t_2/y] \ominus t[t_1/y])\ u & = & ((\lambda z : A.t')[t_1/y] \ominus (\lambda z : A.t')[t_2/y])\ u \\ & = & \lambda z : A.(t'[t_2/y] \ominus t'[t_1/y])\ u \\ & = & (t'[t_2/y] \ominus t'[t_1/y])[u/z] \\ & = & (t'[u/z])[t_2/y] \ominus (t'[u/z])[t_1/y] \end{array}$$

  By induction (on B), we know $\int_{t_1}^{t_2} \frac{\partial t'[u/z]}{\partial y}|_x dx = (t'[u/z])[t_2/y] \ominus (t'[u/z])[t_1/y]$, thus we prove the case.
- Case: t_1,t_2 are of base types, t is of type $T_1 + T_2$. This case is impossible because the righthand term is not well-typed.

- Case: t_1,t_2 are of type $(T_1, T_2, ..., T_n)$, t is of any type T. By using the confluence lemma, we know there exist the normal forms $(t'_{11}, t'_{12}, ..., t'_{1n})$ and $(t'_{21}, t'_{22}, ..., t'_{2n})$ for t_1 and t_2 respectively.
  Applying Rules (EAppDer3) and (EAppInt3), we have

$$\begin{array}{rcl}\int_{t_1}^{t_2} \frac{\partial t}{\partial y}|_x dx & = & \int_{(t'_{11},t'_{12},...,t'_{1n})}^{(t'_{21},t'_{22},...,t'_{2n})} \frac{\partial t}{\partial y}|_x dx \\ & = & \int_{t'_{11}}^{t'_{21}} \pi_1(\frac{\partial t}{\partial y}|_x[(x_1, t'_{12}, ...t'_{1n})/x])dx_1 \oplus \cdots \oplus \\ & & \quad \int_{t'_{1n}}^{t'_{2n}} \pi_n(\frac{\partial t}{\partial y}|_x[(t'_{21}, t'_{22}, ..., x_n)/x])dx_n\end{array}$$

  Notice that there is no occurrence of x in t, so we have

$$\begin{array}{l}\int_{t'_{1j}}^{t'_{2j}} \pi_j(\frac{\partial t}{\partial y}|_x[(t'_{21}, t'_{22}, ..., t'_{2(j-1)}, x_j, t'_{1(j+1)}, ..., t'_{1n})/x])dx_j \\ \quad = \int_{t'_{1j}}^{t'_{2j}} \pi_j(\frac{\partial t}{\partial y}|_{(t'_{21},t'_{22},...,t'_{2(j-1)},x_j,t'_{1(j+1)},...,t'_{1n})})dx_j \\ \quad = \int_{t'_{1j}}^{t'_{2j}} \pi_j(\frac{\partial t[t_{1*}/y]}{\partial x_1}|_{t'_{21}}, \frac{\partial t[t_{2*}/y]}{\partial x_2}|_{t'_{22}}, ..., \frac{\partial t[t_{(j-1)*}/y]}{\partial x_{j-1}}|_{t'_{2(j-1)}}, \\ \qquad\qquad \frac{\partial t[t_{j*}/y]}{\partial x'_j}|_{x_j}, \frac{\partial t[t_{(j+1)*}/y]}{\partial x_{j+1}}|_{t'_{1(j+1)}}, ..., \frac{\partial t[t_{n*}/y]}{\partial x_n}|_{t'_{1n}})dx_j \\ \quad = \int_{t'_{1j}}^{t'_{2j}} \frac{\partial t[(t'_{21},t'_{22},...,t'_{2(j-1)},x'_j,t'_{1(j+1)},...,t'_{1n})/y]}{\partial x'_j}|_{x_j} dx_j\end{array}$$

  By induction (on the case where t_1, t_2 are of type T_j, t is of type T), we have

$$\begin{array}{l}\int_{t'_{1j}}^{t'_{2j}} \frac{\partial t[(t'_{21},t'_{22},...,t'_{2(j-1)},x'_j,t'_{1(j+1)},...,t'_{1n})/y]}{\partial x'_j}|_{x_j} dx_j \\ \quad = (t[(t'_{21}, t'_{22}, ..., t'_{2(j-1)}, x'_j, t'_{1(j+1)}, ..., t'_{1n})/y])[t'_{2j}/x'_j] \ominus \\ \qquad\qquad (t[(t'_{21}, t'_{22}, ..., t'_{2(j-1)}, x'_j, t'_{1(j+1)}, ..., t'_{1n})/y])[t'_{1j}/x'_j] \\ \quad = (t[(t'_{21}, t'_{22}, ..., t'_{2(j-1)}, t'_{2j}, t'_{1(j+1)}, ..., t'_{1n})/y]) \ominus \\ \qquad\qquad (t[(t'_{21}, t'_{22}, ..., t'_{2(j-1)}, t'_{1j}, t'_{1(j+1)}, ..., t'_{1n})/y])\end{array}$$

  Note that the last equation holds because x'_j is a fresh variable and t has no occurrence of x'_j.
  Now we have the following calculation.

$$\begin{array}{cl} & \int_{t_1}^{t_2} \frac{\partial t}{\partial y}|_x dx \\ = & \quad \{ \text{ all the above } \} \\ & ((t[(t'_{21}, t'_{12}, ..., t'_{1n})/y]) \ominus (t[(t'_{11}, t'_{12}, ..., t'_{1n})/y])) \oplus \\ & \qquad ((t[(t'_{21}, t'_{22}, ..., t'_{1n})/y]) \ominus (t[(t'_{21}, t'_{12}, ..., t'_{1n})/y])) \oplus \cdots \oplus \\ & \qquad\quad ((t[(t'_{21}, t'_{22}, ..., t'_{2n})/y]) \ominus (t[(t'_{21}, t'_{22}, ..., t'_{1n})/y])) \\ = & \quad \{ \text{ Lemma 18 } \} \\ & (t[(t'_{21}, t'_{22}, ..., t'_{2n})/y]) \ominus (t[(t'_{11}, t'_{12}, ..., t'_{1n})/y]) \\ = & \quad \{ \text{ Lemma 51 } \} \\ & t[t_2/y] \ominus t[t_1/y]\end{array}$$

◀

Application: Incremental Computation

A direct application is incrementalization [18, 7, 11]. Given a function $f(x)$, if the input x is changed by Δ, then we can obtain its incremental version of $f(x)$, $f'(x, \Delta)$,

$$f(x \oplus \Delta) = f(x) \oplus f'(x, \Delta)$$

where f' satisfies that

$$f'(x, \Delta) = \int_x^{x \oplus \Delta} \frac{\partial f(x)}{\partial x}|_x \; dx.$$

▶ **Example 21** (Averaging a Pair of Real numbers)**.** As a simple example, consider the average of a pair of real numbers

$$\begin{array}{lcl} average & :: & (\mathbb{R}, \mathbb{R}) \to \mathbb{R} \\ average & = & \lambda x.(\pi_1(x) + \pi_2(x))/2 \end{array}$$

Suppose that we want to get an incremental computation of *average* at $x = (x_1, x_2)$ when the first element x_1 is changed to $x_1 + d$ while the second component x_2 is kept the same. The incremental computation is defined by

$$inc(x, d) = average(x, (d, 0)) = \int_x^{x \oplus (d,0)} \frac{\partial average(x)}{\partial x}|_x \; dx = \frac{d}{2}$$

which is efficient.

4 The Chain Rule

The Chain Rule is another important theorem of the relation between function composition and derivatives. This Chain Rule in our calculus has many important applications in automatic differentiation and incremental computation. We first give an example to get some taste, before we give and prove the theorem.

▶ **Example 22** (chain rule)**.** Consider two functions f and g on real numbers, usually defined in mathematics as $f(x, y) = (x + y, x * y, y)$ and $g(x, y) = (x + y, y)$. In our calculus, they are defined as follows.

$$\begin{array}{lcl} f & :: & (\mathbb{R}, \mathbb{R}) \to (\mathbb{R}, \mathbb{R}, \mathbb{R}) \\ f & = & \lambda x : (\mathbb{R}, \mathbb{R}).(\pi_1(x) \oplus \pi_2(x), \pi_1(x) * \pi_2(x), \pi_2(x)) \\ g & :: & (\mathbb{R}, \mathbb{R}) \to (\mathbb{R}, \mathbb{R}) \\ g & = & \lambda x : (\mathbb{R}, \mathbb{R}).(\pi_1(x) \oplus \pi_2(x), \pi_2(x)) \end{array}$$

We demostrate that for any $r_1, r_2, r_3, r_4 \in \mathbb{R}$, we have

$$\frac{\partial f(g\ x)}{\partial x}|_{(r_3, r_4)} * (r_1, r_2) = \frac{\partial f\ x}{\partial x}|_{g\ (r_3,r_4)} * (\frac{\partial g\ x}{\partial x}|_{(r_3,r_4)} * (r_1, r_2))$$

by the following calculation. First, for the LHS, we have:

$$\begin{array}{ll} & \frac{\partial f(g\ x)}{\partial x}|_{(r_3,r_4)} * (r_1, r_2) \\ = & \quad \{ \text{ Rule EAppDer3 } \} \\ & \frac{\partial f(g\ (x_1,r_4))}{\partial x_1}|_{r_3} * r_1 \oplus \frac{\partial f(g\ (r_3,x_2))}{\partial x_2}|_{r_4} * r_2 \\ = & \quad \{ \text{ Application } \} \\ & \frac{\partial (x_1 \oplus r_4 \oplus r_4, (x_1 \oplus r_4) * r_4, r_4)}{\partial x_1}|_{r_3} * r_1 \oplus \frac{\partial (r_3 \oplus x_2 \oplus x_2, (r_3 \oplus x_2) * x_2, x_2)}{\partial x_2}|_{r_4} * r_2 \\ = & \quad \{ \text{ Lemma 16 } \} \\ & (1, r_4, 0) * r_1 \oplus (2, r_3 \oplus 2 * r_4, 1) * r_2 \\ = & \quad \{ \text{ Rule EAppMul1 and Rule EAppAdd1 } \} \\ & (r_1 \oplus (2 * r_2), r_4 * r_1 \oplus (r_3 \oplus (2 * r_4)) * r_2, r_2) \end{array}$$

Now, for the RHS, we calculate with the following two steps.

$$
\begin{array}{ll}
& \frac{\partial g\ x}{\partial x}|_{(r_3,r_4)} * (r_1, r_2) \\
= & \quad \{ \text{ Rule EAppDer3 } \} \\
& \frac{\partial g\ (x_1,r_4)}{\partial x_1}|_{r_3} * r_1 \oplus \frac{\partial g\ (r_3,x_2)}{\partial x_2}|_{r_4} * r_2 \\
= & \quad \{ \text{ Application } \} \\
& \frac{\partial (x_1 \oplus r_4, r_4)}{\partial x_1}|_{r_3} * r_1 \oplus \frac{\partial (r_3 \oplus x_2, x_2)}{\partial x_2}|_{r_4} * r_2 \\
= & \quad \{ \text{ Lemma 16 } \} \\
& (1,0) * r_1 \oplus (1,1) * r_2 \\
= & \quad \{ \text{ Rule EAppMul1 and Rule EAppAdd1 } \} \\
& (r_1 \oplus r_2, r_2)
\end{array}
$$

$$
\begin{array}{ll}
& \frac{\partial f\ x}{\partial x}|_{g\ (r_3,r_4)} * (r_1 \oplus r_2, r_2) \\
= & \quad \{ \text{ Application and Rule EAppDer3 } \} \\
& (\frac{\partial f\ (x_1,r_4)}{\partial x_1}|_{r_3 \oplus r_4}, \frac{\partial f\ (r_3 \oplus r_4, x_2)}{\partial x_2}|_{r_4}) * (r_1 \oplus r_2, r_2) \\
= & \quad \{ \text{ Application } \} \\
& (\frac{\partial (x_1 \oplus r_4, x_1 * r_4, r_4)}{\partial x_1}|_{r_3 \oplus r_4}, \frac{\partial (r_3 \oplus r_4 \oplus x_2, (r_3 \oplus r_4) * x_2, x_2)}{\partial x_2}|_{r_4}) * (r_1 \oplus r_2, r_2) \\
= & \quad \{ \text{ Lemma 16 } \} \\
& ((1, r_4, 0), (1, (r_3 \oplus r_4), 1)) * (r_1 \oplus r_2, r_2) \\
= & \quad \{ \text{ Rule EAppMul4, Rule EAppAdd1 and Lemma 16 } \} \\
& (r_1 \oplus (2 * r_2), r_4 * r_1 \oplus (r_3 \oplus (2 * r_4)) * r_2, r_2)
\end{array}
$$

∎

▶ **Theorem 23** (Chain Rule). Let $f : T_1 \to T$, $g : T_2 \to T_1$. If both $\frac{\partial f(g\ x)}{\partial x}|_{t_1} * t$ and $\frac{\partial f\ y}{\partial y}|_{(g\ t_1)} * (\frac{\partial g\ z}{\partial z}|_{t_1} * t)$ are well-typed and normalizable. Then for any $t, t_1 : T_2$, we have

$$\frac{\partial f(g\ x)}{\partial x}|_{t_1} * t = \frac{\partial f\ y}{\partial y}|_{(g\ t_1)} * (\frac{\partial g\ z}{\partial z}|_{t_1} * t)$$

Proof. Like in the proof of Theorem 20, for simplicity, we assume that f, g, t and t_1 are closed. Furthermore, we assume that t and t_1 are in normal form. We will prove this by induction on types.

- Case T, T_2 are base types, and T_1 is any type. To be well-typed, T_1 must contain no $\to$ or $+$ type. So for simplicity, we suppose T_1 to be $(B, B, B, ..., B)$ of n-tuples, but the technique below can be applied to any T_1 type (such as tuples of tuples) that makes the term well-typed.
  First we notice that

$$
\begin{array}{rcl}
g\ z & = & (\pi_1(g\ z), \pi_2(g\ z), ..., \pi_n(g\ z)) \\
& = & ((\lambda b' : B.\pi_1(g\ b'))\ z, (\lambda b' : B.\pi_2(g\ b'))\ z, ..., (\lambda b' : B.\pi_n(g\ b'))\ z)
\end{array}
$$

  and for any j, we notice that $\pi_j(g\ b')$ has only one free variable of base type, so it can be reduced to a normal form, say E_j, of base type. Let g_j be $\lambda b' : B.E_j$, then we have $g\ z = (g_1\ z, g_2\ z, ..., g_n\ z)$.
  Next, we deal with the term f:

$$
\begin{array}{rcl}
f & = & \lambda a : T_1.\, (f\ a) \\
& = & \lambda a : T_1.\, ((\lambda y_1 : B.\, \lambda y_2 : B., ...\lambda y_n : B.\, (f\ (y_1, y_2, ..., y_n)))\ \pi_1(a)\ \pi_2(a) ...\ \pi_n(a))
\end{array}
$$

and we know that $(f\ (y_1, y_2, ..., y_n))$ only contains base type free variables, so it can be reduced to a base type normal form, say N, so we have

$$f = \lambda a : T.\,((\lambda y_1 : B.\lambda y_2 : B., ...\lambda y_n : B.\,N)\ \pi_1(a)\ \pi_2(a)...\ \pi_n(a)).$$

Now, we can calculate as follows:

$$\begin{aligned}
&\tfrac{\partial f(g\ x)}{\partial x}|_{t_1} * t\\
&= \tfrac{\partial(\lambda a:T.(\lambda y_1:B.\lambda y_2:B.,...\lambda y_n:B.N)\ \pi_1(a)\ \pi_2(a)...\ \pi_n(a))\ (g_1\ x, g_2\ x, ..., g_n\ x)}{\partial x}|_{t_1} * t\\
&= \tfrac{\partial(\lambda y_1:B.\lambda y_2:B.,...\lambda y_n:B.N)\ (g_1\ x)\ (g_2\ x)...\ (g_n\ x)}{\partial x}|_{t_1} * t\\
&= \tfrac{\partial N[(g_1\ x)/y_1, (g_2\ x)/y_2, ...(g_n\ x)/y_n]}{\partial x}|_{t_1} * t
\end{aligned}$$

$$\begin{aligned}
&\tfrac{\partial f\ y}{\partial y}|_{(g\ t_1)} * (\tfrac{\partial g\ z}{\partial z}|_{t_1} * t)\\
&= \tfrac{\partial f\ y}{\partial y}|_{(g_1\ t_1, g_2\ t_1, ..., g_n\ t_1)} * (\tfrac{\partial(g_1\ z, g_2\ z, ..., g_n\ z)}{\partial z}|_{t_1} * t)\\
&= \tfrac{\partial f\ y}{\partial y}|_{(g_1\ t_1, g_2\ t_1, ..., g_n\ t_1)} * (\tfrac{\partial g_1\ z}{\partial z}|_{t_1} * t, \tfrac{\partial g_2\ z}{\partial z}|_{t_1} * t, ..., \tfrac{\partial g_n\ z}{\partial z}|_{t_1} * t)\\
&= \tfrac{\partial(\lambda y_1:B.\lambda y_2:B.,...\lambda y_n:B.N)\ \pi_1(y)\ \pi_2(y)...\ \pi_n(y)}{\partial y}|_{(g_1\ t_1, g_2\ t_1, ..., g_n\ t_1)} *\\
&\qquad (\tfrac{\partial g_1\ z}{\partial z}|_{t_1} * t, \tfrac{\partial g_2\ z}{\partial z}|_{t_1} * t, ..., \tfrac{\partial g_n\ z}{\partial z}|_{t_1} * t)\\
&= (\tfrac{\partial N[y_1'/y_1, g_2\ t_1/y_2, ..., g_n\ t_1/y_n]}{\partial y_1'}|_{g_1\ t_1}, ..., \tfrac{\partial N[g_1\ t_1/y_1, g_2\ t_1/y_2, ..., y_n'/y_n]}{\partial y_n'}|_{g_n\ t_1}) *\\
&\qquad (\tfrac{\partial g_1\ z}{\partial z}|_{t_1} * t, \tfrac{\partial g_2\ z}{\partial z}|_{t_1} * t, ..., \tfrac{\partial g_n\ z}{\partial z}|_{t_1} * t)\\
&= (\tfrac{\partial N[y_1'/y_1, g_2\ t_1/y_2, ..., g_n\ t_1/y_n]}{\partial y_1'}|_{g_1\ t_1} * (\tfrac{\partial g_1\ z}{\partial z}|_{t_1} * t)) \oplus ... \oplus\\
&\qquad (\tfrac{\partial N[g_1\ t_1/y_1, g_2\ t_1/y_2, ..., y_n'/y_n]}{\partial y_n'}|_{g_n\ t_1} * (\tfrac{\partial g_n\ z}{\partial z}|_{t_1} * t))
\end{aligned}$$

Notice that by the base type interpretation, $f(g_1(x), g_2(x), ..., g_n(x)) = f_1'(g_1(x), g_2(x), ..., g_n(x)) * g_1'(x) + f_2'(g_1(x), g_2(x), ..., g_n(x)) * g_2'(x) + ... + f_n'(g_1(x), g_2(x), ..., g_n(x)) * g_n'(x)$ where f_j' means the derivative of f to its j-th parameter, so we get the following and prove the case.

$$\begin{aligned}
&\tfrac{\partial N[(g_1\ x)/y_1, (g_2\ x)/y_2, ...(g_n\ x)/y_n]}{\partial x}|_{t_1} * t\\
&= (\tfrac{\partial N[y_1'/y_1, g_2\ t_1/y_2, ..., g_n\ t_1/y_n]}{\partial y_1'}|_{g_1\ t_1} * (\tfrac{\partial g_1\ z}{\partial z}|_{t_1} * t)) \oplus ... \oplus\\
&\qquad (\tfrac{\partial N[g_1\ t_1/y_1, g_2\ t_1/y_2, ..., y_n'/y_n]}{\partial y_n'}|_{g_n\ t_1} * (\tfrac{\partial g_n\ z}{\partial z}|_{t_1} * t))
\end{aligned}$$

- Case T_2 is base type, T_1 is any type, T is $A \to B$. We prove that for any u of type A, we have $(\frac{\partial f(g\ x)}{\partial x}|_{t_1} * t)\ u = (\frac{\partial f\ y}{\partial y}|_{(g\ t_1)} * (\frac{\partial g\ z}{\partial z}|_{t_1} * t))\ u$.
  First, let $f' = \lambda x : T_1.(f\ x)\ u$, $g' = g$, then by induction we have

$$\frac{\partial f'(g'\ x)}{\partial x}|_{t_1} * t = \frac{\partial f'\ y}{\partial y}|_{(g'\ t_1)} * (\frac{\partial g'\ z}{\partial z}|_{t_1} * t)$$

that is, we have

$$(\frac{\partial f\ (g\ x)\ u}{\partial x}|_{t_1} * t) = (\frac{\partial f\ y\ u}{\partial y}|_{(g\ t_1)} * (\frac{\partial g\ z}{\partial z}|_{t_1} * t))$$

Then, we prove $(\frac{\partial f\ (g\ x)\ u}{\partial x}|_{t_1} * t) = (\frac{\partial f\ (g\ x)}{\partial x}|_{t_1} * t)\ u$ by the following calculation.

$$\begin{aligned}
(\tfrac{\partial f\ (g\ x)}{\partial x}|_{t_1} * t)\ u &= (\tfrac{\partial \lambda a:A.(f\ (g\ x))\ a}{\partial x}|_{t_1} * t)\ u\\
&= (\lambda a : A.(\tfrac{\partial (f\ (g\ x))\ a}{\partial x}|_{t_1} * t))\ u\\
&= (\tfrac{\partial f\ (g\ x)\ u}{\partial x}|_{t_1} * t)
\end{aligned}$$

Next, we prove $(\frac{\partial f\ y}{\partial y}|_{(g\ t_1)} * (\frac{\partial g\ z}{\partial z}|_{t_1} * t))\ u = \frac{\partial f\ y\ u}{\partial y}|_{(g\ t_1)} * (\frac{\partial g\ z}{\partial z}|_{t_1} * t)$. For simplicity, we assume T_1 to be $(B, B, B, ..., B)$ of n-tuples (the technique below can be applied to any T_1 type which makes the term well-typed).

On one hand, by substituting $(g_1\ z, g_2\ z, ..., g_n\ z)$ for $g\ z$, we have

$$\begin{aligned}
&(\frac{\partial f\ y}{\partial y}|_{(g\ t_1)} * (\frac{\partial g\ z}{\partial z}|_{t_1} * t))\ u \\
&\quad = \frac{\partial f\ y}{\partial y}|_{(g_1\ t_1, g_2\ t_1, ..., g_n\ t_1)} * (\frac{\partial (g_1\ z, g_2\ z, ..., g_n\ z)}{\partial z}|_{t_1} * t)\ u \\
&\quad = (\frac{\partial f(y_1, g_2\ t_1, ..., g_n\ t_1)}{\partial y_1'}|_{g_1\ t_1} * (\frac{\partial g_1\ z}{\partial z}|_{t_1} * t) \oplus ... \oplus \\
&\qquad\qquad \frac{\partial f(g_1\ t_1, g_2\ t_1, ..., y_n)}{\partial y_n'}|_{g_n\ t_1} * (\frac{\partial g_n\ z}{\partial z}|_{t_1} * t))\ u
\end{aligned}$$

Since

$$\begin{aligned}
&f(g_1\ t_1, g_2\ t_1, ..., g_{j-1}\ t_1, y_j, g_{j+1}\ t_1, ..., g_n\ t_1) \\
&\quad = \lambda a : A.f\ (g_1\ t_1, g_2\ t_1, ..., g_{j-1}\ t_1, y_j, g_{j+1}\ t_1, ..., g_n\ t_1)\ a
\end{aligned}$$

which will be denoted as $\lambda a : A.t_j^*$, we continue the calculation as follows.

$$\begin{aligned}
&(\frac{\partial f\ y}{\partial y}|_{(g\ t_1)} * (\frac{\partial g\ z}{\partial z}|_{t_1} * t))\ u \\
&\quad = (\frac{\partial \lambda a{:}A.t_1^*}{\partial y_1}|_{g_1\ t_1} * (\frac{\partial g_1\ z}{\partial z}|_{t_1} * t) \oplus ... \oplus \frac{\partial \lambda a{:}A.t_n^*}{\partial y_n}|_{g_n\ t_1} * (\frac{\partial g_n\ z}{\partial z}|_{t_1} * t))\ u \\
&\quad = \lambda a : A.(\frac{\partial t_1^*}{\partial y_1}|_{g_1\ t_1} * (\frac{\partial g_1\ z}{\partial z}|_{t_1} * t)) \oplus ... \oplus (\frac{\partial t_n^*}{\partial y_n}|_{g_n\ t_1} * (\frac{\partial g_n\ z}{\partial z}|_{t_1} * t))\ u \\
&\quad = \frac{\partial t_1^*[u/a]}{\partial y_1}|_{g_1\ t_1} * (\frac{\partial g_1\ z}{\partial z}|_{t_1} * t) \oplus ... \oplus \frac{\partial t_n^*[u/a]}{\partial y_n}|_{g_n\ t_1} * (\frac{\partial g_n\ z}{\partial z}|_{t_1} * t)
\end{aligned}$$

On the other hand, we have

$$\begin{aligned}
&(\frac{\partial f\ y\ u}{\partial y}|_{(g\ t_1)} * (\frac{\partial g\ z}{\partial z}|_{t_1} * t)) \\
&\quad = \frac{\partial f\ y\ u}{\partial y}|_{(g_1\ t_1, g_2\ t_1, ..., g_n\ t_1)} * (\frac{\partial (g_1\ z, g_2\ z, ..., g_n\ z)}{\partial z}|_{t_1} * t) \\
&\quad = \frac{\partial f\ (y_1, g_2\ t_1, ..., g_n\ t_1)\ u}{\partial y_1}|_{g_1\ t_1} * (\frac{\partial g_1\ z}{\partial z}|_{t_1} * t) \oplus ... \oplus \\
&\qquad\qquad \frac{\partial f\ (g_1\ t_1, g_2\ t_1, ..., y_n)\ u}{\partial y_n}|_{g_n\ t_1} * (\frac{\partial g_n\ z}{\partial z}|_{t_1} * t) \\
&\quad = \frac{\partial (\lambda a{:}A.t_1^*)\ u}{\partial y_1}|_{g_1\ t_1} * (\frac{\partial g_1\ z}{\partial z}|_{t_1} * t) \oplus ... \oplus \frac{\partial (\lambda a{:}A.t_n^*)\ u}{\partial y_n}|_{g_n\ t_1} * (\frac{\partial g_n\ z}{\partial z}|_{t_1} * t) \\
&\quad = (\frac{\partial t_1^*[u/a]}{\partial y_1}|_{g_1\ t_1} * (\frac{\partial g_1\ z}{\partial z}|_{t_1} * t)) \oplus ... \oplus (\frac{\partial t_n^*[u/a]}{\partial y_n}|_{g_n\ t_1} * (\frac{\partial g_n\ z}{\partial z}|_{t_1} * t))
\end{aligned}$$

Therefore, we prove the case.

- Case T_2 is base type, T_1 is any type, T is $(T_1, T_2, ..., T_n)$. We need to prove that for all j, we have $\pi_j(\frac{\partial f(g\ x)}{\partial x}|_{t_1} * t) = \pi_j(\frac{\partial f\ y}{\partial y}|_{(g\ t_1)} * (\frac{\partial g\ z}{\partial z}|_{t_1} * t))$. We may follow the proof for the case when T has type $A \to B$. Let $f' = \lambda x : T_1.\, \pi_j(f\ x), g' = g$, by induction, we have

$$\frac{\partial \pi_j(f(g\ x))}{\partial x}|_{t_1} * t = \frac{\partial \pi_j(f\ y)}{\partial y}|_{(g\ t_1)} * (\frac{\partial g\ z}{\partial z}|_{t_1} * t)$$

The rest of the proof is similar to that for the case when $T = A \to B$.

- Case T_2 is base type, T_1 is any type, T is $T_1 + T_2$. Notice that T_1 has to be base type to be well-typed. But either the case, the proof is similar to the case when $T = A \to B$.
- Case T_2 , T_1 and T are any type. Notice that T_2 does not contain no $\to$ or $+$ to be well-typed (i.e., no derivative over function types). We have proved the case when T_2 is base type, and we assume that T_2 has type $(T_1, T_2, ..., T_n)$. Suppose the normal form of t_1 is $(t_{11}', t_{12}', ..., t_{1n}')$ and the normal form of t is $(t_{21}', t_{22}', ..., t_{2n}')$, Then

$$\begin{aligned}
&\frac{\partial f(g\ x)}{\partial x}|_{t_1} * t \\
&\quad = \frac{\partial f(g\ x)}{\partial x}|_{(t_{11}', t_{12}', ..., t_{1n}')} * (t_{21}', t_{22}', ..., t_{2n}') \\
&\quad = (\frac{\partial f(g\ (x_1, t_{12}', ..., t_{1n}'))}{\partial x_1}|_{t_{11}'}, ..., \frac{\partial f(g\ (t_{11}', t_{12}', ..., x_n))}{\partial x_n}|_{t_{1n}'}) * (t_{21}', ..., t_{2n}') \\
&\quad = (\frac{\partial f(g\ (x_1, t_{12}', ..., t_{1n}'))}{\partial x_1}|_{t_{11}'} * t_{21}') \oplus ... \oplus (\frac{\partial f(g\ (t_{11}', t_{12}', ..., x_n))}{\partial x_n}|_{t_{1n}'} * t_{2n}')
\end{aligned}$$

On the other hand, we can use Lemma 17 (i.e., $t_1 * (t_2 \oplus t_3) = (t_1 * t_2) \oplus (t_1 * t_3)$) to do the following calculation.

$$\begin{aligned}
&\tfrac{\partial f\ y}{\partial y}|_{(g\ t_1)} * (\tfrac{\partial g\ z}{\partial z}|_{t_1} * t) \\
&\quad = \tfrac{\partial f\ y}{\partial y}|_{(g\ t_1)} * ((\tfrac{\partial g\ (x_1, t'_{12}, \ldots, t'_{1n})}{\partial x_1}|_{t'_{11}} * t'_{21}) \oplus \ldots \oplus (\tfrac{\partial g\ (t'_{11}, t'_{12}, \ldots, x_n)}{\partial x_n}|_{t'_{1n}} * t'_{2n})) \\
&\quad = \tfrac{\partial f\ y}{\partial y}|_{(g\ t_1)} * (\tfrac{\partial g\ (x_1, t'_{12}, \ldots, t'_{1n})}{\partial x_1}|_{t'_{11}} * t'_{21}) \oplus \ldots \oplus \\
&\qquad\qquad \tfrac{\partial f\ y}{\partial y}|_{(g\ t_1)} * (\tfrac{\partial g\ (t'_{11}, t'_{12}, \ldots, x_n)}{\partial x_n}|_{t'_{1n}} * t'_{2n})
\end{aligned}$$

Now by induction using $f' = f, g' = \lambda x : T_j.g\ (t'_{11}, t'_{12}, \ldots, t'_{1(j-1)}, x, t'_{1(j+1)}, \ldots, t'_{1n})$, we have

$$\begin{aligned}
&\tfrac{\partial f (g\ (t'_{11}, t'_{12}, \ldots, t'_{1(j-1)}, x_j, t'_{1(j+1)} \ldots, t'_{1n})}{\partial x_j}|_{t'_{1j}} * t'_{2j} \\
&\quad = \tfrac{\partial f\ y}{\partial y}|_{(g'\ t'_{1j})} * (\tfrac{\partial g\ (t'_{11}, t'_{12}, \ldots, t'_{1(j-1)}, x_j, t'_{1(j+1)} \ldots, t'_{1n})}{\partial x_j}|_{t'_{1j}} * t'_{2j}) \\
&\quad = \tfrac{\partial f\ y}{\partial y}|_{(g\ t_1)} * (\tfrac{\partial g\ (t'_{11}, t'_{12}, \ldots, t'_{1(j-1)}, x_j, t'_{1(j+1)} \ldots, t'_{1n})}{\partial x_j}|_{t'_{1j}} * t'_{2j})
\end{aligned}$$

Therefore by Lemma 52, we prove the case.

◀

Application: Automatic Differentiation

The Chain Rule provides another way to compute the derivatives. There are many applications of the chain rule, and here we give an example of how to associate it with the auto differentiation [10].

▶ **Example 24** (AD)**.** This is an example from [10]. Let *sqr* and *magSqr* be defined as follows.

$$\begin{array}{lcl}
sqr & :: & \mathbb{R} \to \mathbb{R} \\
sqr\ a & = & a * a \\
magSqr & :: & (\mathbb{R}, \mathbb{R}) \to \mathbb{R} \\
magSqr\ (a, b) & = & sqr\ a \oplus sqr\ b
\end{array}$$

First of all, let t_1 and t_2 two pairs, then it is easy to prove that $\frac{\partial (t_1 \oplus t_2)}{\partial x}|_{t_3} = \frac{\partial t_1}{\partial x}|_{t_3} \oplus \frac{\partial t_2}{\partial x}|_{t_3}$. Next, we can perform automatic differentiation on *magSqr* by the following calculation.

$$\begin{aligned}
&\tfrac{\partial (magSqr\ x)}{\partial x}|_{(a,b)} * t \\
&\quad = \tfrac{\partial (sqr(\pi_1 x) \oplus sqr(\pi_2 x))}{\partial x}|_{(a,b)} * t \\
&\quad = \tfrac{\partial (sqr\ y)}{\partial y}|_{\pi_1 (a,b)} * (\tfrac{\partial (\pi_1 x)}{\partial x}|_{(a,b)} * t) \oplus \tfrac{\partial (sqr\ y)}{\partial y}|_{\pi_2 (a,b)} * (\tfrac{\partial (\pi_2 x)}{\partial x}|_{(a,b)} * t) \\
&\quad = 2 * a * ((1, 0) * t) \oplus 2 * b * ((0, 1) * t)
\end{aligned}$$

Now, because the theorem applies for any t of pair type, we use $(1, 0)$ and $(0, 1)$ to substitute for t respectively, and we will get $\frac{\partial (magSqr\ x)}{\partial x}|_{(a,b)} = (2 * a, 2 * b)$, which means its derivative to a is $2 * a$ and its derivative to b is $2 * b$.

5 Taylor's Theorem

In this section, we discuss Taylor's Theorem, which is useful to give an approximation of a k-order differentiable function around a given point by a polynomial of degree k. In programming, it is important and has many applications in approximation and incremental computation. We first give an example and then we prove the theorem.

First of all, we introduce some high-order notations.

$$\begin{array}{lclcl}
\frac{\partial^0 t_1}{\partial x^0}|_{t_2} & = & t_1 & \frac{\partial^n t_1}{\partial x^n}|_{t_2} & = & \frac{\partial \frac{\partial^{n-1} t_1}{\partial x^{n-1}}|_x}{\partial x}|_{t_2} \\
t * t_1^0 & = & t & t * t_1^n & = & (t * t_1) * t_1^{n-1} \\
f^0 & = & f & f^n & = & (f')^{n-1} \\
(\lambda x : T.\, t)' & = & \lambda x : T.\, \frac{\partial t}{\partial x}|_x & & &
\end{array}$$

▶ **Example 25** (Taylor). Consider a function f on real numbers, usually defined in mathematics as $f(x, y) = (2 * x * y, 3 * x * x + y)$. In our calculus, it is defined as follows.

$$\begin{array}{lcl}
f & :: & (\mathbb{R}, \mathbb{R}) \to (\mathbb{R}, \mathbb{R}) \\
f & = & \lambda x : (\mathbb{R}, \mathbb{R}).(2 * \pi_1(x) * \pi_2(x), 3 * \pi_1(x) * \pi_1(x) \oplus \pi_2(x))
\end{array}$$

The following expand the Taylor's theorem up to 2-order derivative.

$$\begin{array}{lcl}
f\ (C_1, C_2) & = & (2 * C_1 * C_2, 3 * C_1 * C_1 \oplus C_2) \\
f\ (0, 0) & = & (0, 0) \\
f'\ (0, 0) * (C_1, C_2) & = & \quad \{ \text{ Application } \} \\
& & \frac{\partial (2*\pi_1(x)*\pi_2(x), 3*\pi_1(x)*\pi_1(x) \oplus \pi_2(x))}{\partial x}|_{(0,0)} * (C_1, C_2) \\
& = & \quad \{ \text{ Rule } \textsc{EAppDer3} \} \\
& & (\frac{\partial (2*x_1*0, 3*x_1*x_1 \oplus 0)}{\partial x_1}|_0, \frac{\partial (2*0*x_2, 3*0*0 \oplus x_2)}{\partial x_2}|_0) * (C_1, C_2) \\
& = & \quad \{ \text{ Lemma 16 } \} \\
& & ((0, 0), (0, 1)) * (C_1, C_2) \\
& = & \quad \{ \text{ Rule } \textsc{EAppMul4}, \text{ Rule } \textsc{EAppAdd1} \} \\
& & (0, C_2)
\end{array}$$

$$\begin{array}{lcl}
f''\ (0, 0) & = & \quad \{ \text{ Application } \} \\
& & \frac{\partial \frac{\partial (2*\pi_1(x)*\pi_2(x), 3*\pi_1(x)*\pi_1(x) \oplus \pi_2(x))}{\partial x}|_x}{\partial x}|_{(0,0)} \\
& = & \quad \{ \text{ Rule } \textsc{EAppDer3} \} \\
& & (\frac{\partial \frac{\partial (2*\pi_1(x)*\pi_2(x), 3*\pi_1(x)*\pi_1(x) \oplus \pi_2(x))}{\partial x}|_{(x_1, 0)}}{\partial x_1}|_0, \frac{\partial \frac{\partial (2*\pi_1(x)*\pi_2(x), 3*\pi_1(x)*\pi_1(x) \oplus \pi_2(x))}{\partial x}|_{(0, x_2)}}{\partial x_2}|_0) \\
& = & \quad \{ \text{ Rule } \textsc{EAppDer3} \} \\
& & (\frac{\partial (\frac{\partial (2*x_1'*0, 3*x_1'*x_1' \oplus 0)}{\partial x_1'}|_{x_1}, \frac{\partial (2*x_1*x_2', 3*x_1*x_1 \oplus x_2')}{\partial x_2'}|_0)}{\partial x_1}|_0, \frac{\partial (\frac{\partial (2*x_1'*x_2, 3*x_1'*x_1' \oplus x_2)}{\partial x_1'}|_0, \frac{\partial (2*0*x_2', 3*0*0 \oplus x_2')}{\partial x_2'}|_{x_2})}{\partial x_2}|_0) \\
& = & \quad \{ \text{ Lemma 16 } \} \\
& & (\frac{\partial ((0, 6*x_1), (2*x_1, 1))}{\partial x_1}|_0, \frac{\partial ((2*x_2, 0), (0, 1))}{\partial x_2}|_0) \\
& = & \quad \{ \text{ Lemma 16 } \} \\
& & (((0, 6), (2, 0)), ((2, 0), (0, 0)))
\end{array}$$

$$\begin{array}{lcl}
(f''\ (0, 0)) * (C_1, C_2)^2 & = & \quad \{ \text{ Rule } \textsc{EAppMul4}, \text{ Rule } \textsc{EAppAdd1} \} \\
& & ((2 * C_2, 6 * C_1), (2 * C_1, 0)) * (C_1, C_2) \\
& = & \quad \{ \text{ Rule } \textsc{EAppMul4} \} \\
& & (2 * C_2 * C_1, 6 * C_1 * C_1) \oplus (2 * C_1 * C_2, 0) \\
& = & \quad \{ \text{ Rule } \textsc{EAppAdd1} \} \\
& & (4 * C_1 * C_2, 6 * C_1 * C_1)
\end{array}$$

Thus we have $f\ (C_1, C_2) = (2 * C_1 * C_2, 3 * C_1 * C_1 \oplus C_2) = (0, 0) \oplus (0, C_2) \oplus (2 * C_1 * C_2, 3 * C_1 * C_1) = f\ (0, 0) \oplus \frac{1}{1!}(f'\ (0, 0)) * (C_1, C_2) \oplus \frac{1}{2!}((f''\ (0, 0)) * (C_1, C_2)) * (C_1, C_2)$

▶ **Theorem 26** (Taylor's Theorem)**.** If both $f\ t$ and $\sum\limits_{k=0}^{\infty} \frac{1}{k!}(f^{(k)}\ t_0) * (t \ominus t_0)^k$ are normalizable(normalizable means it has a normal form), then

$$f\ t = \sum_{k=0}^{\infty} \frac{1}{k!}(f^{(k)}\ t_0) * (t \ominus t_0)^k$$

Proof. Like in the proof of Theorem 20, for simplicity, we assume that f, g, t and t_1 are closed. Furthermore, we assume that t and t_1 are in normal form. We prove it by induction on the type of $f : T \to T'$.

- Case T' is a base type. T must contain no $\to$ by our typing, so for simplicity, we suppose T to be $(B, B, ..., B)$. Using the same technique in Theorem 23, we assume that

  $$f = \lambda x : T.\, (\lambda x_1 : B.\, \lambda x_2 : B., ...\lambda x_n : B.\, N)\ \pi_1(x)\ \pi_2(x) ...\ \pi_n(x)$$

  denoted by $f = \lambda a : T.\, t_2$, and we assume t to be $(t_{11}, t_{12}, ..., t_{1n})$, and t_0 to be $(t_{21}, t_{22}, ..., t_{2n})$, where each t_{ij} is a normal form of base type. Then we have

$$\begin{aligned}
&(f^{(n)}\ t_0) * (t \ominus t_0)^n \\
&\quad = \frac{\partial^n t_2}{\partial x^n}|_{t_0} * (t \ominus t_0)^n \\
&\quad = \Big(\frac{\partial \frac{\partial^{n-1} t_2}{\partial x^{n-1}}|_{(x_1,t_{22},\ldots,t_{2n})}}{\partial x_1}|_{t_{21}}, ..., \frac{\partial \frac{\partial^{n-1} t_2}{\partial x^{n-1}}|_{(t_{21},t_{22},\ldots,x_n)}}{\partial x_n}|_{t_{2n}}\Big) * (t \ominus t_0)^n \\
&\quad = \Big(\frac{\partial \frac{\partial^{n-1} t_2}{\partial x^{n-1}}|_{(x_1,t_{22},\ldots,t_{2n})}}{\partial x_1}|_{t_{21}} * (t_{11} \ominus t_{21}) \oplus ... \oplus \frac{\partial \frac{\partial^{n-1} t_2}{\partial x^{n-1}}|_{(t_{21},t_{22},\ldots,x_n)}}{\partial x_n}|_{t_{2n}} * (t_{1n} \ominus t_{2n})\Big) * (t \ominus t_0)^{n-1} \\
&\quad = \Big(\Big(\frac{\partial\big(\frac{\partial \frac{\partial^{n-2} t_2}{\partial x^{n-2}}|_{(x_1,t_{22},\ldots,t_{2n})}}{\partial x_1}|_{x_1}, ..., \frac{\partial \frac{\partial^{n-2} t_2}{\partial x^{n-2}}|_{(x_1,t_{22},\ldots,x_n)}}{\partial x_n}|_{t_{2n}}\big)}{\partial x_1}|_{t_{21}}\Big) * (t_{11} \ominus t_{21}) \oplus ... \oplus \\
&\qquad\qquad \Big(\frac{\partial\big(\frac{\partial \frac{\partial^{n-2} t_2}{\partial x^{n-2}}|_{(x_1,t_{22},\ldots,x_n)}}{\partial x_1}|_{t_{21}}, ..., \frac{\partial \frac{\partial^{n-2} t_2}{\partial x^{n-2}}|_{(t_{21},t_{22},\ldots,x_n)}}{\partial x_n}|_{x_n}\big)}{\partial x_n}|_{t_{2n}}\Big) * (t_{1n} \ominus t_{2n})\Big) * (t \ominus t_0)^{n-1} \\
&\quad = \Big(\Big(\frac{\partial \frac{\partial \frac{\partial^{n-2} t_2}{\partial x^{n-2}}|_{(x_1,t_{22},\ldots,t_{2n})}}{\partial x_1}|_{x_1}}{\partial x_1}|_{t_{21}}\Big) * (t_{11} \ominus t_{21})^2 \oplus ... \oplus \\
&\qquad\qquad \Big(\Big(\frac{\partial \frac{\partial \frac{\partial^{n-2} t_2}{\partial x^{n-2}}|_{(x_1,t_{22},\ldots,x_n)}}{\partial x_1}|_{t_{21}}}{\partial x_n}|_{t_{2n}}\Big) * (t_{1n} \ominus t_{2n})\Big) * (t_{11} \ominus t_{21}), \\
&\qquad \Big(\Big(\frac{\partial \frac{\partial \frac{\partial^{n-2} t_2}{\partial x^{n-2}}|_{(x_1,x_2,\ldots,t_{2n})}}{\partial x_2}|_{t_{22}}}{\partial x_1}|_{t_{21}}\Big) * (t_{11} \ominus t_{21})\Big) * (t_{12} \ominus t_{22}) \oplus ... \oplus \\
&\qquad\qquad \Big(\Big(\frac{\partial \frac{\partial \frac{\partial^{n-2} t_2}{\partial x^{n-2}}|_{(t_{11},x_2,\ldots,x_n)}}{\partial x_2}|_{t_{22}}}{\partial x_n}|_{t_{2n}}\Big) * (t_{1n} \ominus t_{2n})\Big) * (t_{12} \ominus t_{22}), \\
&\qquad ... \\
&\qquad \Big(\Big(\frac{\partial \frac{\partial \frac{\partial^{n-2} t_2}{\partial x^{n-2}}|_{(x_1,t_{22},\ldots,x_n)}}{\partial x_n}|_{t_{2n}}}{\partial x_1}|_{t_{21}}\Big) * (t_{11} \ominus t_{21})\Big) * (t_{1n} \ominus t_{2n}) \oplus ... \oplus \\
&\qquad\qquad \Big(\frac{\partial \frac{\partial \frac{\partial^{n-2} t_2}{\partial x^{n-2}}|_{(t_{11},t_{22},\ldots,x_n)}}{\partial x_n}|_{x_n}}{\partial x_n}|_{t_{2n}}\Big) * (t_{1n} \ominus t_{2n})^2\Big) * (t \ominus t_0)^{n-2} \\
&\quad = ...
\end{aligned}$$

  As seen in the above, every time we decompose a $\frac{\partial}{\partial x_i}|_{(\ldots)}$, apply Rule EAppDer1, and then make reduction with Rule EAppMul3 to lower down the exponent of $(t \ominus t_0)^n$. Finally, we will decompose the last derivative and get the term t_2 in the form of $t_2[t'_{21}/x_1, t'_{22}/x_2, ..., t'_{2n}/x_n]$ where $\forall j \in [1, n], t'_{2j}$ is either t_{2j} or x_j.
  Note that on base type we assume that we have Taylor's Theorem:

  $$f(x_0 + h) = f(x_0) + \sum_{k=1}^{\infty} \frac{1}{k!}\Big(\sum_{i=1}^{n} h_i \frac{\partial}{\partial x_i}\Big)^k f(x_0)$$

where x_0 and h is an n-dimensional vector, and x_j, h_j is its projection to its j-th dimension.
So we have $(f^{(k)}\ t_0) * (t \ominus t_0)^k$ corresponds to the k-th addend $\frac{1}{k!}(\sum_{i=1}^{n} h_i \frac{\partial}{\partial x_i})^k f(x_0)$.

- Case: T' is function type $A \to B$. Similar to the proof in Theorem 23, for all u of type A, we define $f^* = \lambda x : T.\ f\ x\ u$, and by using the inductive result on type B, we can prove the case simiarly as that in Theorem 23.
- Case: T' is a tuple type $(T_1, T_2, T_3, \ldots)$. Just define $f^* = \lambda x : T.\ \pi_j(f\ x)$ to use inductive result. The rest is simple.
- Case: T' is a tuple type $T_1 + T_2$. This case is impossible because the righthand is not well-typed.

◀

Application: Polynomial Approximation

Taylor's Theorem has many applications. Here we give an example of using Taylor's Theorem for approximation. Suppose there is a point $(1, 0)$ in the polar coordinate system, and we want to know where the point will be if we slightly change the radius r and the angle θ. Since it is extremely costive to compute functions such as sin() and cos(), Taylor's Theorem enables us to make a fast polynomial approximation.

▶ **Example 27.** Let function *polar2catesian* be defined by

$$\begin{array}{lcl} \mathit{polar2cartesian} & :: & (\mathbb{R}, \mathbb{R}) \to (\mathbb{R}, \mathbb{R}) \\ \mathit{polar2cartesian}(r, \theta) & = & (r * \cos(\theta), r * \sin(\theta)) \end{array}$$

We will demonstrate how to expand $\mathit{polar2cartesian}(r, \theta)$ at $(1, 0)$ up to 2nd-order derivative. Since

$$\begin{array}{ll} \frac{\partial(\mathit{polar2cartesian}(x))}{\partial x}|_{(1,0)} & = \frac{\partial(\pi_1 x * \mathit{cos}(\pi_2 x), \pi_1 x * \mathit{sin}(\pi_2 x))}{\partial x}|_{(1,0)} \\ & = (\frac{\partial(x_1 * \cos(0), x_1 * \sin(0))}{\partial x_1}|_1, \frac{\partial(1 * \cos(x_2), 1 * \sin(x_2))}{\partial x_2}|_0) \\ & = ((1, 0), (0, 1)) \end{array}$$

we have

$$\frac{\partial(\mathit{polar2cartesian}(x))}{\partial x}|_{(1,0)} * (\Delta r, \Delta\theta) = (\Delta r, \Delta\theta)$$

Again, we have

$$\begin{array}{ll} \frac{1}{2}\frac{\partial^2(\mathit{polar2cartesian}(x))}{\partial x^2}|_{(1,0)} * (\Delta r, \Delta\theta)^2 & = (((0, 0), (0, 1)), ((0, 1), (-1, 0))) * (\Delta r, \Delta\theta)^2 \\ & = (-\frac{1}{2}\Delta\theta^2, \Delta r * \Delta\theta) \end{array}$$

Combining the above, we can use $(1 \oplus \Delta r \ominus \frac{1}{2}\Delta\theta^2, \Delta\theta \oplus \Delta r * \Delta\theta)$ to make an approximation to $\mathit{polar2cartesian}(1 + \Delta r, \Delta\theta)$

6 Discussion

In this section, we makes remarks on generality of our approach, and on how to deal with discrete derivatives in our context.

Normal Form	nf	$::=$	nb	interpretable forms on base type
		$\mid$	$\lambda x : T.t$	function, t can't be further reduced
Interpretable Normal Forms	nb	$::=$	c	constants of interpretable type
		$\mid$	$nb\ nf$	primitive functions application
		$\mid$	$nb \oplus nf \quad \mid nf \oplus nb$	addition on interpretable type
		$\mid$	$nb \ominus nf \quad \mid nf \ominus nb$	subtraction on interpretable type
		$\mid$	x	interpretable types variables
		$\mid$	$\frac{\Delta nb}{\Delta x}\vert_{nf,nf}$	derivative on base type

Figure 6 Discrete normal form

6.1 More Theorems and Applications

We keep many mathematical structures in our calculus. As a result, we can prove more theorems under this framework. We select the most important three, but there are many other theorems that hold in our system:

- $(t_1 \ominus t_2) \oplus (t_2 \ominus t_3) \oplus ... \oplus (t_{n-1} \ominus t_n) = t_1 \ominus t_n$,
- $\frac{\partial t_1 \oplus t_2}{\partial x}|_{t_3} = \frac{\partial t_1}{\partial x}|_{t_3} \oplus \frac{\partial t_2}{\partial x}|_{t_3}$,
- $x \in B$, $\frac{\partial t_1 * t_2}{\partial x}|_{t_3} = \frac{\partial t_1}{\partial x}|_{t_3} * t_2[t_3/x] \oplus t_1[t_3/x] * \frac{\partial t_2}{\partial x}|_{t_3}$,
- $\forall t_1$, if t_1 contains no free x, $\frac{\partial t_1 * x}{\partial x}|_{t_2} = t_1$.

Associated with each of these theorems is a bunch of applications. For lack of space, we only discuss three theorems in detail.

Now it is natural to ask whether all the theorems on base types have correspondence in our system. The answer is that it depends on the mathematical structure of the base types. In our proof, we assume the commutative law and associative law of addition and multiplication, and the distributive law of multiplication. We can construct a counterexample under this case. Suppose there is a strange law on a base type that $\forall x, y, x * y = y$, which is interpreted by our system as $t_1 * t_2 = t_2$. Now let t_1 be $((r_1, r_2), (r_3, r_4))$, and t_2 be (r_5, r_6). Then

$$
\begin{aligned}
t_1 * t_2 &= (r_1 * r_5 \oplus r_3 * r_6, r_2 * r_5 \oplus r_4 * r_6) \\
&= (r_5 \oplus r_6, r_5 \oplus r_6)
\end{aligned}
$$

which does not equal to t_2. This means that our system does not preserve this strange law.

In our design of the calculus, we touch little on details of base types. So for some strange base types, we may not be able to preserve its mathematical structure. But as for the widely used $\mathbb{R}$ and $\mathbb{C}$, our system preserves most of their important theorems.

It is interesting to note that it is impossible to prove these theorems using the theory of change [7], because the theory of change does not tell difference between smooth functions and non-continuous functions and use the same calculation for them. In our calculus, we distribute these calculation to base types step by step, and use these calculation (such as on base types, we have $\int_{a_1}^{a_2} \frac{\partial f\ y}{\partial y}|_x = f\ a_2 \ominus f\ a_1$) to prove our theorems.

6.2 Discrete Derivatives

We can define discrete version of our calculus, where we represent changes as discrete deltas instead of through derivatives and integrations. We will show the equivalence between our discrete version and the change theory [7] by implementing function *Derive* in our calculus.

The normal form this time is defined in Figure 6. We use the term $\frac{\Delta t}{\Delta x}|_{t,t}$ to represent discrete derivative. This time we can easily manipulate values of base types because we only require the operator $\oplus$ and $\ominus$ to be well-defined. Also notice that this time we can implement derivatives on function type.

To show that our discrete version can be used to implement the change theory [7] it is sufficient to consider terms of base types or function types, without need to to consider tuples and the operator * and $\int$. We want to use our calculus to implement function *Derive* which satisfies the equation $(Derive\ f)x\ \Delta x = f(x \oplus \Delta x) \ominus f(x)$.

For interpretation of derivatives on base types, we just require they satisfy $\frac{\Delta t}{\Delta y}|_{t_1,t_2} = t[t_1 \oplus t_2/y] \ominus t[t_1/y]$. Then similarly to Newton-Leibniz Theorem we can prove $\frac{\Delta(f\ y)}{\Delta y}|_{x,\Delta x} = f\ (x \oplus \Delta x) \ominus f\ x$ (where f does not contain free y), which is our version of function *Derive*.

To see this clear, in the change theory, we write function *Derive* and the system will automatically calculate it by using the rules:

612

$$\begin{array}{lcl} Derive\ c & = & 0 \\ Derive\ x & = & \Delta x \\ Derive(\lambda x : T.t) & = & \lambda x : T.\, \lambda dx : \Delta T.\, Derive(t) \\ Derive(s\ t) & = & Derive(s)\ t\ Derive(t) \end{array}$$

In our calculus, one writes $\frac{\Delta f\ y}{\Delta y}|_{x,\Delta x}$, and the system will automatically calculate the following rules:

615

$$\begin{array}{lcl} \frac{\Delta c}{\Delta y}|_{x,\Delta x} & = & 0 \\ \frac{\Delta y}{\Delta y}|_{x,\Delta x} & = & \Delta x \\ \frac{\Delta \lambda y:T.t}{\Delta x}|_{t_0,t_1} & = & \lambda y : T.(\frac{\Delta t}{\Delta x}|_{t_0,t_1}) \\ (\lambda x.\lambda \Delta x.\, \frac{\Delta t}{\Delta y}|_{x,\Delta x})\ t_1\ t_2 & = & \lambda y : T.\, t\ (t_1 \oplus t_2) \ominus \lambda y : T.\, t\ t_1 \end{array}$$

Notice that the first three rules have good correspondence, while the last one is a bit different. This is because in the change theory's definition, we have $\Delta(A \to B) = A \to \Delta A \to \Delta B$, while in our calulus, we have $\Delta(A \to B) = A \to \Delta B$. We, fortunately, can achieve the same effect through Newton-Leibniz Formula Theorem.

7 Related Work

Differential Calculus and The Change Theory The differential λ-calculus [9, 8] has been studied for computing derivatives of arbitrary higher-order programs. In the differential λ-calculus, derivatives are guaranteed to be linear in its argument, where the incremental λ-calculus does not have this restriction. Instead, it requires that the function should be differentiable. The big difference between our calculus and differential lambda calculus is that we perform computation on terms instead of analysis on terms.

The idea of performing incremental computation using derivatives has been studied by Cai et al. [7], who give an account using change structures. They use this to provide a framework for incrementally evaluating lambda calculus programs. It is shown that the work can be enriched with recursion and fix-point computation [3]. The main difference between our work and the change theory is that we describe changes as mathematical derivatives while the change theory describe changes as (discrete) deltas.

Incremental/Self-Adaptive Computation Paige and Koenig [20] present derivatives for a first-order language with a fixed set of primitives for incremental computation. Blakeley et al. [17] apply these ideas to a class of relational queries. Koch [15] guarantees asymptotic speedups with a compositional query transformation and delivers huge speedups in realistic benchmarks, though still for a first-order database language. We have proved the Taylor's theorem in our framework, which provides us with another way to perform finite difference on the computation.

Self-adjusting computation [1] or adaptive function programming [2] provides a dynamic approach to incrementalization. In this approach, programs execute on the original input in an enhanced runtime environment that tracks the dependencies between values in a dynamic dependence graph; intermediate results are memoized. Later, changes to the input propagate through dependency graphs from changed inputs to results, updating both intermediate and final results; this processing is often more efficient than recomputation. Mathematically, the self-adjusting computation corresponds to differential equation (The change rate (or derivative) of a function can be represented by the computational result of function), which may be a future work of our calculus.

Automatic Differentiation Automatic differentiation [13] is a technique that allows for efficiently computing the derivative of arbitrary programs, and can be applied to probabilistic modeling [16] and machine learning [5]. This technique has been successfully applied to some higher-order languages [22, 10]. As pointed out in [3], while some approaches have been suggested [19, 14], a general theoretical framework for this technique is still a matter of open research. We prove the chain rule inside our framework, which lays a foundation for our calculus to perform automatic differentiation. And with more theorems in our calculus, we expect more profound application in differential calculus.

Three Theorems

We choose to prove three important theorems in our calculus. Each one has its own important meaning in mathematics. The Newton-Leibniz formula ensures the correct semantics of integration, which lays the solid foundation for mathematical analysis. The chain rule shows some of the most important characters of derivative. It applies to any general differentiable function f and g, and shows the deep connection between the function composition and their derivatives. The Taylor's theorem stands for one of the most beautiful theorems in mathematical analysis. It implies the nature of smooth function and their polynomial approximation.

For the Newton-Leibniz formula

$$\int_{t_1}^{t_2} \frac{\partial t}{\partial y}|_x dx = t[t_2/y] \ominus t[t_1/y]$$

it is much related to

$$f(x \oplus \Delta x) = f(x) \oplus (\mathit{Derive}\ f)\ x\ \Delta x$$

in the change theory [7]. They lay the foundation for both system. But one important difference is that their formula is built-in while our formula is an invariant property that is provable.

For our chain rule, it manifests the relation between function composition and their derivative, and shows the good transformation property of derivative, which may have many profound applications. There are built-in chain rules

$$\mathit{Derive}(s\ t) = \mathit{Derive}(s)\ s\ \mathit{Derive}(t)$$

and

$$\frac{\partial (s) t}{\partial x} \cdot u = (\frac{\partial s}{\partial x} \cdot u) t + (D\ s \cdot (\frac{\partial t}{\partial x} \cdot u)) t$$

in the change theory [7] and differential lambda calculus [9, 23], respectively. But in contrast, our chain rule

$$\frac{\partial f(g\ x)}{\partial x}|_{t_1} * t = \frac{\partial f\ y}{\partial y}|_{g\ t_1} * (\frac{\partial g\ z}{\partial z}|_{t_1} * t)$$

is a property and is provable based on the reduction semantics of our calculus. Also, many research works [14, 19] have been done on automatic differentiation based on the change theory and the differential lambda calculus, but the chain rule there is treated as a meta reduction rule in the AD methods while our chain rule is inherently in the calculus and applied more naturally in AD.

For Taylor's Theorem, it has important applications in the field of approximation. our theorem

$$f\ t = \sum_{k=0}^{\infty} \frac{1}{k!}(f^{(k)}\ t_0) * (t \ominus t_0)^k$$

looks much like

$$s\ u = \sum_{n=0}^{\infty} \frac{1}{n!}(D_1^n s \cdot u^n)0$$

in the differential lambda calculus [6, 24, 9], but their meaning is completely different: in the differential lambda calculus, the Taylor's theorem shows the alternative of linear substitution and useful for analyzing different alternatives, but in our calculus, it shows the property of approximation of computation itself.

8 Conclusion

In this paper, we propose an analytical differential calculus which is equipped with integration. This calculus, as far as we are aware, is the first one that has well-defined integration, which has not appeared in both differential lambda calculus and the change theory. Our calculus enjoys many nice properties such as soundness and strong normalizing (when fix is excluded), and has three important theorems, which have profound applications in computer science. We believe the following directions will be important in our future work.

- *Adding more theorems.* We may wish to write programs on many specialized base types besides $\mathbb{R}$ and $\mathbb{C}$. As we have demonstrated in this paper, our calculus preserves many important computational structures on base types. Therefore, it is possible to extend our system with theorems having ome unique mathematical structures and use these theorems to optimize computation.
- *Working on Derivatives on functions.* We did not talk about derivatives on continuous functions because we have not had a good mathematical definition for them from perspective of computation. But derivatives on functions would be useful; it would be nice if we could use $\int_{\oplus}^{*} \frac{\partial x(a_1,a_2)}{\partial x}|_x dx$ to compute $a_1 * a_2 \ominus (a_1 \oplus a_2)$.
- *Manipulating differential equations.* Differential equations would be very useful for users to program dynamic systems directly; one may write differential equations on data structures without writing the primitive forms of functions. It could be applied in many fields such as self-adjusting computation or self-adaptive system construction.

References

1 Umut A. Acar, Amal Ahmed, and Matthias Blume. Imperative self-adjusting computation. In George C. Necula and Philip Wadler, editors, *Proceedings of the 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2008, San Francisco, California, USA, January 7-12, 2008*, pages 309–322. ACM, 2008.

2 Umut A. Acar, Guy E. Blelloch, and Robert Harper. Adaptive functional programming. In John Launchbury and John C. Mitchell, editors, *Conference Record of POPL 2002: The 29th SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Portland, OR, USA, January 16-18, 2002*, pages 247–259. ACM, 2002.

3 Mario Alvarez-Picallo, Alex Eyers-Taylor, Michael Peyton Jones, and C.-H. Luke Ong. Fixing incremental computation - derivatives of fixpoints, and the recursive semantics of datalog. In Luís Caires, editor, *Programming Languages and Systems - 28th European Symposium on Programming, ESOP 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings*, volume 11423 of *Lecture Notes in Computer Science*, pages 525–552. Springer, 2019.

4 Hendrik Barendregt. *The Lambda-Calculus: Its Syntax and Semantics.* North Holland, 1984.

5 Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *J. Mach. Learn. Res.*, 18:153:1–153:43, 2017.

6 Pierre Boudes, Fanny He, and Michele Pagani. A characterization of the taylor expansion of λ-terms, 2013.

7 Yufei Cai, Paolo G. Giarrusso, Tillmann Rendel, and Klaus Ostermann. A theory of changes for higher-order languages: incrementalizing λ-calculi by static differentiation. In Michael F. P. O'Boyle and Keshav Pingali, editors, *ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '14, Edinburgh, United Kingdom - June 09 - 11, 2014*, pages 145–155. ACM, 2014.

8 Thomas Ehrhard. An introduction to differential linear logic: proof-nets, models and antiderivatives. *Math. Struct. Comput. Sci.*, 28(7):995–1060, 2018.

9 Thomas Ehrhard and Laurent Regnier. The differential lambda-calculus. *Theor. Comput. Sci.*, 309(1-3):1–41, 2003.

10 Conal Elliott. The simple essence of automatic differentiation. *Proc. ACM Program. Lang.*, 2(ICFP):70:1–70:29, 2018.

11 Paolo G. Giarrusso, Yann Régis-Gianas, and Philipp Schuster. Incremental \lambda -calculus in cache-transfer style - static memoization by program transformation. In Luís Caires, editor, *Programming Languages and Systems - 28th European Symposium on Programming, ESOP 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings*, volume 11423 of *Lecture Notes in Computer Science*, pages 553–580. Springer, 2019.

12 Jean-Yves Girard. *Proofs and Types.* Cambridge University Press, 1989.

13 Andreas Griewank and Andrea Walther. *Evaluating derivatives - principles and techniques of algorithmic differentiation, Second Edition.* SIAM, 2008.

14 Robert Kelly, Barak A. Pearlmutter, and Jeffrey Mark Siskind. Evolving the incremental λ calculus into a model of forward automatic differentiation (AD). *CoRR*, abs/1611.03429, 2016. URL: `http://arxiv.org/abs/1611.03429`.

15 Christoph Koch. Incremental query evaluation in a ring of databases. In Jan Paredaens and Dirk Van Gucht, editors, *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2010, June 6-11, 2010, Indianapolis, Indiana, USA*, pages 87–98. ACM, 2010.

16 Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman, and David M. Blei. Automatic differentiation variational inference. *J. Mach. Learn. Res.*, 18:14:1–14:45, 2017.

17 Per-Åke Larson and Jingren Zhou. Efficient maintenance of materialized outer-join views. In Rada Chirkova, Asuman Dogac, M. Tamer Özsu, and Timos K. Sellis, editors, *Proceedings*

of the 23rd International Conference on Data Engineering, ICDE 2007, The Marmara Hotel, Istanbul, Turkey, April 15-20, 2007, pages 56–65. IEEE Computer Society, 2007.

18 Yanhong A. Liu. Efficiency by incrementalization: An introduction. *High. Order Symb. Comput.*, 13(4):289–313, 2000. `doi:10.1023/A:1026547031739`.

19 Oleksandr Manzyuk. A simply typed λ-calculus of forward automatic differentiation. *Electronic Notes in Theoretical Computer Science*, 286:257 – 272, 2012. Proceedings of the 28th Conference on the Mathematical Foundations of Programming Semantics (MFPS XXVIII).

20 Robert Paige and Shaye Koenig. Finite differencing of computable expressions. *ACM Trans. Program. Lang. Syst.*, 4(3):402–454, 1982.

21 Benjamin C. Pierce. *Types and Programming Languages*. The MIT Press, 2002.

22 Jeffrey Mark Siskind and Barak A. Pearlmutter. Nesting forward-mode AD in a functional framework. *High. Order Symb. Comput.*, 21(4):361–376, 2008.

23 Lionel Vaux. The differential lambda-mu-calculus. *Theoretical Computer Science*, 379(1-2):166–209, 2007. URL: `https://hal.archives-ouvertes.fr/hal-00349304`, `doi:10.1016/j.tcs.2007.02.028`.

24 Lionel Vaux. Taylor Expansion, lambda-Reduction and Normalization. In Valentin Goranko and Mads Dam, editors, *26th EACSL Annual Conference on Computer Science Logic (CSL 2017)*, volume 82 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 39:1–39:16, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: `http://drops.dagstuhl.de/opus/volltexte/2017/7694`, `doi:10.4230/LIPIcs.CSL.2017.39`.

A Appendix-Calculus Property

A.1 Progress

▶ **Lemma 28** (Progress). *Suppose t is a well-typed term (Allow free variables of interpretable type iB), then t is either a normal form or there is some t' that t→ t'.*

Proof. We prove this by induction on form of t.

- Case c.
  It is a normal form.
- Case $t_1 \oplus t_2$.
  t is well-typed if and only if t_1 has the same type T with t_2. If t_1 or t_2 is not a normal form, we make reductions on t_1 or t_2. If both t_1 and t_2 are normal forms, if either t_1 or t_2 is **nb**, then $t_1 \oplus t_2$ is a **nb**, for other cases of normal forms, we have

$$\begin{array}{rcl}(t_{11}, t_{12}, ...t_{1n}) \oplus (t_{21}, t_{22}, ...t_{2n}) & \to & (t_{11} \oplus t_{21}, t_{12} \oplus t_{22}, ...t_{1n} \oplus t_{2n}) \\ (\lambda x : T.t_1) \oplus (\lambda y : T.t_2) & \to & \lambda x : T.t_1 \oplus (t_2[x/y])\end{array}$$

- Case $t_1 \ominus t_2$.
  It is the same case with the $t_1 \oplus t_2$.
- Case x.
  Then x is an interpretable type free variable, otherwise it is not well-typed. and an interpretable type free variable is a normal form.
- Case $inl/inr\ t$.
  if t is not a normal form, then we can make reduction in t, else this term itself is a normal form.
- Case $case\ t\ of\ inl\ x_1 \Rightarrow t_1 |\ inr\ x_2 \Rightarrow t_2$.
  To be well-typed t has to be the type of $T_1 + T_2$, if t is not a normal form, then we can make reduction in t, else t has to be $inl/inr\ t'$. So we can make reduction to $t_1[t'/x_1]$ or $t_2[t'/x_2]$
- Case $\lambda x : T.t$.
  It is a normal form if t can't be further reduced.
- Case $t_1\ t_2$.
  If t_1 is not a normal form then we make reductions on t_1.
  If t_1 is a normal form, then t_1 has to be $\lambda x : T.t$, or **nb**. For the former case we have

$$(\lambda x : T.t)t_1 \to t[t_1/x]$$

  For the latter case, t_2 must be a **nf**, or it can make further reductions. So $t_1\ t_2$ is a **nb**.
- Case $\int_{t_1}^{t_2} t_3 dx$.
  If t_1 or t_2 is not a normal form then we can make reductions on t_1 or t_2.
  If both t_1 and t_2 are normal forms, then t_1,t_2 have to be (**nf**, **nf**, ..**nf**) or base type to be well-typed. if it is the former case.

$$\begin{array}{rcl}\int_{(t_{11}, t_{12}, ...t_{1n})}^{(t_{21}, t_{22}, ..., t_{2n})} t dx & \to & \int_{t_{11}}^{t_{21}} \pi_1(t[(x_1, t_{12}, ...t_{1n})/x])dx_1 \oplus \\ & & \int_{t_{12}}^{t_{22}} \pi_2(t[(t_{21}, x_2, ..., t_{1n})/x])dx_2 \oplus \\ & & \vdots \\ & & \int_{t_{1n}}^{t_{2n}} \pi_n(t[(t_{21}, t_{22}, ..., x_n)/x])dx_n\end{array}$$

If it is the latter case, let us inspect t_3. if t_3 is not a normal form, then we can make reductions on t_3 (notice that we only introduce a base type free variable into t_3).
If t_3 is a normal form, if t_1,t_2 and t_3 are **nb**, then $\int_{t_1}^{t_2} t_3 dx$ is a normal form, for other cases of normal forms:

$$\frac{t_1, t_2 : \mathsf{B}}{\int_{t_1}^{t_2} (t_{11}, t_{12}, ...t_{1n}) dx \ \rightarrow (\int_{t_1}^{t_2} t_{11} dx, \int_{t_1}^{t_2} t_{12} dx, ..., \int_{t_1}^{t_2} t_{1n} dx)}$$

$$\frac{t_1, t_2 : \mathsf{B}}{\int_{t_1}^{t_2} \lambda y : T_2.t dx \ \rightarrow \lambda y : T_2. \int_{t_1}^{t_2} t dx}$$

$$\frac{t_1, t_2 : \mathsf{B}}{\int_{t_1}^{t_2} inl/inr \ t \ dx \rightarrow inl/inr \ \int_{t_1}^{t_2} t \ dx}$$

- Case $(t_1, t_2, ..., t_n)$.
  If t_i is not a normal form, then we make reductions on t_i. If all the t_i are normal forms, then t is a normal form.
- Case $\pi_j(t_1)$.
  If t_1 is not a normal form, then we make reductions on t_1.
  If t_1 is a normal form, then it has to be $(\mathbf{nf}, \mathbf{nf}, ..., \mathbf{nf})$ to be well-typed, then we have

  $$\pi_j(t'_1, t'_2, ...t'_n) \rightarrow t'_j$$

- Case $\frac{\partial t_1}{\partial x}|_{t_2}$.
  If t_2 is not a normal form, then we make reductions on t_2.
  If t_2 is a normal form, then it has to be $(t_1, ..., t_n)$ or an **nb**. if it is the form case:

  $$\frac{\forall i, (t_1, t_2 ..., t_{i-1}, x_i, t_{i+1} ..., t_n) is \ written \ as \ t_{i*}}{\frac{\partial t}{\partial x}|_{(t_1, t_2, ..., t_n)} \rightarrow (\frac{\partial t[t_{1*}/x]}{\partial x_1}|_{t_1}, \frac{\partial t[t_{2*}/x]}{\partial x_2}|_{t_2}, ..., \frac{\partial t[t_{n*}/x]}{\partial x_n}|_{t_n})}$$

  If it is the latter case, if t_1 is not a normal form, then we can make reductions on t_1 (notice that we only introduce a base type free variable into t_1), if t_1 is a **nb**, then t is a **nb**, else we have

  $$\frac{t_0 : \mathsf{B}}{\frac{\partial (t_1, t_2, ..., t_n)}{\partial x}|_{t_0} \ \rightarrow \ (\frac{\partial t_1}{\partial x}|_{t_0}, \frac{\partial t_2}{\partial x}|_{t_0}, ..., \frac{\partial t_n}{\partial x}|_{t_0})}$$

  $$\frac{t_0 : \mathsf{B}}{\frac{\partial (\lambda y : T.t)}{\partial x}|_{t_0} \ \rightarrow \ \lambda y : T. \frac{\partial t}{\partial x}|_{t_0}}$$

  $$\frac{t_0 : \mathsf{B}}{\frac{\partial inl/inr \ t}{\partial x}|_{t_0} \rightarrow inl/inr \ \frac{\partial t}{\partial x}|_{t_0}}$$

- Case $t_1 * t_2$.
  If t_1 or t_2 is not a normal form, then we can make reductions on t_1 or t_2.
  If both t_1 and t_2 are normal forms, t_2 has to be $(t_1, ..., t_n)$ or a **nb**. if it is the former case, t_1 has also to be $(t_1, ..., t_n)$, then we have

  $$\frac{t_1 : (t_{11}, t_{12}, ...t_{1n}), t_2 : (t_{21}, t_{22}, ...t_{2n})}{t_1 * t_2 \rightarrow (t_{11} * t_{21}) \oplus (t_{12} * t_{22}) \oplus ... \oplus (t_{1n} * t_{2n})}$$

If t_2 is a **nb**, if t_1 is a **nb**, then $t_1 * t_2$ is a **nb**, else we have

$$\frac{t_2 : \mathsf{B}}{(\lambda x : T.t) * t_2 \to \lambda x : T.(t * t_2)}$$

$$\frac{t_0 : \mathsf{B}}{(t_1, t_2, ...t_n) * t_0 \to (t_1 * t_0, t_2 * t_0, ...t_n * t_0)}$$

$$\frac{t_0 : \mathsf{B}}{(inl/inr\ t) * t_0 \to inl/inr\ (t * t_0)}$$

- Case $fix\ f$.
  Then we have $fix\ f \to f\ (fix\ f)$

◀

A.2 Preservation

▶ **Lemma 29** (Preservation). *If $t : T$ and $t \to t'$, then $t' : T$. (Allowing free variable of iB)*

▶ **Lemma 30** (Preservation under substitution). *If $\Gamma, x : S \vdash t : T$ and $\Gamma \vdash s : S$, then we have $\Gamma \vdash t[s/x] : T$.*

Proof. First we prove preservation under substitution.

- Case c.
  Then $c[s/x]$ is c, therefore $\Gamma \vdash t[s/x]$:B
- Case $t_1 \oplus t_2$.
  Suppose $\Gamma, x : S \vdash t_1 \oplus t_2 : T$, then we have $\Gamma, x : S \vdash t_1 : T, t_2 : T$, based on induction we have $\Gamma \vdash t_1[s/x] \oplus t_2[s/x] : T$, therefore $\Gamma \vdash (t_1 \oplus t_2)[s/x] : T$
  Using the same techniques we can prove the case of $t_1 \ominus t_2$, $t_1 * t_2$, $t_1\ t_2$, $\lambda x : T.t$, $\frac{\partial t_1}{\partial x}|_{t_2}$, $\int_{t_1}^{t_2} t_3 dx$, $(t_1, t_2, ..., t_n)$, $\pi_j(t)$ and $fix\ f$, $inl/inr\ t$, $case\ t\ of\ inl\ x_1 \Rightarrow t_1|\ inr\ x_2 \Rightarrow t_2$.
- Case y.
  if y = x then y[s/x] = s, so $\Gamma \vdash y[s/x] : T$ if y is other than x, then y[s/x] = y, so $\Gamma \vdash y[s/x] : T$

Then we prove the preservation

- Case $(\lambda x : T.t)t_1 \to t[t_1/x]$: It is straightforward by using the Lemma. (Preservation under substitution)

- Case $fix\ f \to f\ (fix\ f)$
  Suppose $\Gamma \vdash f : A \to A$, then $\Gamma \vdash fix\ f : A$ and $\Gamma \vdash f\ (fix\ f) : A$, so they have the same type.
- Case $\pi_j(t_1, t_2, ...t_n) \to t_j$
  Suppose $\Gamma \vdash (t_1, t_2, ...t_n) : (T_1, T_2, ..., T_n)$, then $\Gamma \vdash \pi_j(t_1, t_2, ...t_n) : T_j$ and $\Gamma \vdash t_j : T_j$, so they have the same type.
- Case

$$\frac{t_0 : \mathsf{B}}{\frac{\partial(t_1,t_2,...,t_n)}{\partial x}|_{t_0} \ \to\ (\frac{\partial t_1}{\partial x}|_{t_0}, \frac{\partial t_2}{\partial x}|_{t_0}, ..., \frac{\partial t_n}{\partial x}|_{t_0})}$$

  Suppose $\Gamma \vdash \frac{\partial(t_1,t_2,...,t_n)}{\partial x}|_{t_0} : (T_1, T_2, .., T_n)$, Then, Γ, x :B$\vdash$ $t_j : T_j$, then $\Gamma \vdash (\frac{\partial t_1}{\partial x}|_{t_0}, \frac{\partial t_2}{\partial x}|_{t_0}, ..., \frac{\partial t_n}{\partial x}|_{t_0}) : (T_1, T_2, .., T_n)$, so they have the same type.
  Using the same technique, we can prove the case

$$\frac{t_0 : \mathsf{B}}{(t_1, t_2, ...t_n) * t_0 \to (t_1 * t_0, t_2 * t_0, ...t_n * t_0)}$$

$$\frac{t_1, t_2 : \mathsf{B}}{\int_{t_1}^{t_2} (t_{11}, t_{12}, ...t_{1n})dx \ \to (\int_{t_1}^{t_2} t_{11}dx, \int_{t_1}^{t_2} t_{12}dx, ..., \int_{t_1}^{t_2} t_{1n}dx)}$$

- Case

$$\frac{t_0 : \mathsf{B}}{\frac{\partial(\lambda y:T.t)}{\partial x}|_{t_0} \ \to \ \lambda y : T.\frac{\partial t}{\partial x}|_{t_0}}$$

Suppose $\Gamma \vdash \frac{\partial \lambda y:T.t}{\partial x}|_{t_0} : A \to B$, then $\Gamma, y : A \vdash \frac{\partial t}{\partial x}|_{t_0} : B$, therefore $\Gamma \vdash \lambda y : T.\frac{\partial t}{\partial x}|_{t_0} : A \to B$, so they have the same type.

Using the same techniques, we can prove the case

$$\frac{t_1, t_2 : \mathsf{B}}{\int_{t_1}^{t_2} \lambda y : T_2.tdx \ \to \lambda y : T_2. \int_{t_1}^{t_2} tdx}$$

$$\frac{t_2 : \mathsf{B}}{(\lambda x : T.t) * t_2 \to \lambda x : T.(t * t_2)}$$

- Case $(t_{11}, t_{12}, ...t_{1n}) \oplus (t_{21}, t_{22}, ...t_{2n}) \to (t_{11} \oplus t_{21}, t_{12} \oplus t_{22}, ...t_{1n} \oplus t_{2n})$

Suppose that $\Gamma \vdash (t_{11}, t_{12}, ...t_{1n}) \oplus (t_{21}, t_{22}, ...t_{2n}) : T$
Then $\Gamma \vdash (t_{11}, t_{12}, ...t_{1n}) : T, (t_{21}, t_{22}, ...t_{2n}) : T$
Let's suppose T is $(T_1, T_2, ..., T_n)$
Then $\Gamma \vdash t_{1i} : T_i, t_{2i} : T_i$
therefore $\Gamma \vdash t_{1i} \oplus t_{2i} : T_i$
So we have $\Gamma \vdash (t_{11} \oplus t_{21}, t_{12} \oplus t_{22}, ...t_{1n} \oplus t_{2n}) : (T_1, T_2, ..., T_n) = T$.

Using the same techniques, we can prove the preservation of the following rules

$$(\lambda x : T.t_1) \oplus (\lambda y : T.t_2) \to \lambda x : T.t_1 \oplus (t_2[x/y])$$
$$(t_{11}, t_{12}, ...t_{1n}) \ominus (t_{21}, t_{22}, ...t_{2n}) \to (t_{11} \ominus t_{21}, t_{12} \ominus t_{22}, ...t_{1n} \ominus t_{2n})$$
$$(\lambda x : T.t_1) \ominus (\lambda y : T.t_2) \to \lambda x : T.t_1 \ominus (t_2[x/y])$$

and reduction for $case\ t\ of\ inl\ x_1 \Rightarrow t_1|\ inr\ x_2 \Rightarrow t_2$

- Case

$$\frac{\forall i, (t_1, t_2..., t_{i-1}, x_i, t_{i+1}..., t_n) is\ written\ as\ t_{i*}}{\frac{\partial t}{\partial x}|_{(t_1,t_2,...,t_n)} \to (\frac{\partial t[t_{1*}/x]}{\partial x_1}|_{t_1}, \frac{\partial t[t_{2*}/x]}{\partial x_2}|_{t_2}, ..., \frac{\partial t[t_{n*}/x]}{\partial x_n}|_{t_n})}$$

Let's suppose $\frac{\partial t}{\partial x}|_{(t_1,t_2,...,t_n)}$ has the type $\frac{\partial T}{\partial T_0}$
Then we have $\Gamma, x : T_0 \vdash t : T$
Let's suppose $T_0 = (T_1, T_2, ..., T_n)$
Then we got $\Gamma, x_i : T_i \vdash t[t_{i*}/x] : T$
So $\Gamma \vdash \frac{\partial t[t_{i*}/x]}{\partial x_i}|_{t_i} : \frac{\partial T}{\partial T_i}$
Therefore $(\frac{\partial t[t_{1*}/x]}{\partial x_1}|_{t_1}, \frac{\partial t[t_{2*}/x]}{\partial x_2}|_{t_2}, ..., \frac{\partial t[t_{n*}/x]}{\partial x_n}|_{t_n}) : (\frac{\partial T}{\partial T_1}, \frac{\partial T}{\partial T_2}, ..., \frac{\partial T}{\partial T_n}) = \frac{\partial T}{\partial T_0}$.

- Case $\int_{(t_{11},...,t_{1n})}^{(t_{21},...,t_{2n})} tdx \rightarrow \int_{t_{11}}^{t_{21}} \pi_1(t[(x_1, t_{12}, ...t_{1n})/x])dx_1 \oplus \cdots \oplus \int_{t_{1n}}^{t_{2n}} \pi_n(t[(t_{21}, t_{22}, ..., x_n)/x])dx_n$

  Let's suppose $\Gamma, x : T_0 \vdash t : \frac{\partial T}{\partial T_0}$ and $\Gamma \vdash \int_{(t_{11},t_{12},...t_{1n})}^{(t_{21},t_{22},...,t_{2n})} tdx : T$.

  Suppose $T_0 = (T_1, T_2, ..., T_n)$

  Then for all j, we have $\Gamma, x_j : T_j \vdash t[(t_{21}, ..., t_{2(j-1)}, x_j, t_{1(j+1)}, ..., t_{1n})/x] : \frac{\partial T}{\partial T_0}$

  So $\Gamma, x_j : T_j \vdash \int_{t_{1j}}^{t_{2j}} \pi_j(t[(t_{21}, ..., t_{2(j-1)}, x_j, t_{1(j+1)}, ..., t_{1n})/x])dx_j : T_j$

  Therefore $\Gamma \vdash \int_{t_{11}}^{t_{21}} \pi_1(t[(x_1, t_{12}, ...t_{1n})/x_1])dx_1 \oplus \int_{t_{12}}^{t_{22}} \pi_2(t[(t_{21}, x_2, ..., t_{1n})/x_2])dx_2 \oplus$
  $... \oplus \int_{t_{1n}}^{t_{2n}} \pi_n(t[(t_{21}, t_{22}, ..., x_n)/x_n])dx_n : T$.

  Using the same technique, we can prove the case.

$$\frac{t_1 : (t_{11}, t_{12}, ...t_{1n}), t_2 : (t_{21}, t_{22}, ...t_{2n})}{t_1 * t_2 \rightarrow (t_{11} * t_{21}) \oplus (t_{12} * t_{22}) \oplus ... \oplus (t_{1n} * t_{2n})}$$

Therefore, we prove the preservation of the system. ◀

A.3 Confluence

Define a binary relation $\twoheadrightarrow$ by induction on relation on terms.

$M \twoheadrightarrow M$

$$\frac{M \twoheadrightarrow M', N \twoheadrightarrow N'}{M\ N \twoheadrightarrow M'\ N', M \oplus N \twoheadrightarrow M' \oplus N', M \ominus N \twoheadrightarrow M' \ominus N', M * N \twoheadrightarrow M' * N', \frac{\partial M}{\partial x}|_N \twoheadrightarrow \frac{\partial M'}{\partial x}|_{N'}}$$

$$\frac{\forall j \in [1, n], M_j \twoheadrightarrow M_j'}{(M_1, M_2, .., M_n) \twoheadrightarrow (M_1', M_2', .., M_n')}$$

$$\frac{M_1 \twoheadrightarrow M_1', M_2 \twoheadrightarrow M_2', M_3 \twoheadrightarrow M_3'}{\int_{M_1}^{M_2} M_3 dx \twoheadrightarrow \int_{M_1'}^{M_2'} M_3' dx, case\ M_1\ of\ inl\ x_1 \Rightarrow M_2|\ inr\ x_2 \Rightarrow M_3 \twoheadrightarrow case\ M_1'\ of\ inl\ x_1 \Rightarrow M_2'|\ inr\ x_2 \Rightarrow M_3'}$$

$$\frac{M_1 \twoheadrightarrow M_1', M_2 \twoheadrightarrow M_2', M_3 \twoheadrightarrow M_3'}{case\ inr\ M_1\ of\ inl\ x_1 \Rightarrow M_2|\ inr\ x_2 \Rightarrow M_3 \twoheadrightarrow M_3'[M_1'/x_2]}$$

$$\frac{M_1 \twoheadrightarrow M_1', M_2 \twoheadrightarrow M_2', M_3 \twoheadrightarrow M_3'}{case\ inl\ M_1\ of\ inl\ x_1 \Rightarrow M_2|\ inr\ x_2 \Rightarrow M_3 \twoheadrightarrow M_2'[M_1'/x_1]}$$

$$\frac{M \twoheadrightarrow M'}{\lambda x : T.M \twoheadrightarrow \lambda x : T.M', \pi_j(M) \twoheadrightarrow \pi_j(M'), inl/inr\ M \twoheadrightarrow inl/inr\ M}$$

$$\frac{\forall j \in [1, n], M_j \twoheadrightarrow M_j'}{\pi_j(M_1, M_2, .., M_n) \twoheadrightarrow M_j'}$$

$$\frac{M \twoheadrightarrow M', N \twoheadrightarrow N'}{(\lambda x : T.M)N \twoheadrightarrow M'[N'/x]}$$

$$\frac{\forall j \in [1, n], M_{1j} \twoheadrightarrow M_{1j}', M_{2j} \twoheadrightarrow M_{2j}'}{(M_{11}, M_{12}, ...M_{1n}) \oplus (M_{21}, M_{22}, ...M_{2n}) \twoheadrightarrow (M_{11}' \oplus M_{21}', M_{12}' \oplus M_{22}', ...M_{1n}' \oplus M_{2n}')}$$

$$\frac{\forall j \in [1,n], M_{1j} \twoheadrightarrow M'_{1j}, M_{2j} \twoheadrightarrow M'_{2j}}{(M_{11}, M_{12}, ...M_{1n}) \ominus (M_{21}, M_{22}, ...M_{2n}) \twoheadrightarrow (M'_{11} \ominus M'_{21}, M'_{12} \ominus M'_{22}, ...M'_{1n} \ominus M'_{2n})}$$

$$\frac{M \twoheadrightarrow M', N \twoheadrightarrow N'}{(\lambda x : T.M) \oplus (\lambda y : T.N) \twoheadrightarrow \lambda x : T.M' \oplus N'[y/x]}$$

$$\frac{M \twoheadrightarrow M', N \twoheadrightarrow N'}{(\lambda x : T.M) \ominus (\lambda y : T.N) \twoheadrightarrow \lambda x : T.M' \ominus N'[y/x]}$$

$$\frac{\forall j \in [1,n], M_j \twoheadrightarrow M'_j, N \twoheadrightarrow N', N : \mathsf{B}}{\frac{\partial (M_1, M_2, .., M_n)}{\partial x}|_N \twoheadrightarrow (\frac{\partial M'_1}{\partial x}|_{N'}, \frac{\partial M'_2}{\partial x}|_{N'}, ..., \frac{\partial M'_n}{\partial x}|_{N'})}$$

$$\frac{M \twoheadrightarrow M', N \twoheadrightarrow N', N : \mathsf{B}}{\frac{\partial \lambda y{:}T.M}{\partial x}|_N \twoheadrightarrow \lambda y : T.\frac{\partial M'}{\partial x}|_{N'}}$$

$$\frac{\forall j \in [1,n], M_j \twoheadrightarrow M'_j, (M'_1, M'_2..., M'_{j-1}, x_j, M'_{j+1}..., M'_n) is\ written\ as\ M'_{j*}, M_0 \twoheadrightarrow M'_0}{\frac{\partial M_0}{\partial x}|_{(M_1, M_2, ..., M_n)} \twoheadrightarrow (\frac{\partial M'_0[M'_{1*}/x]}{\partial x_1}|_{M'_1}, \frac{\partial M_0[M'_{2*}/x]}{\partial x_2}|_{M'_2}, ..., \frac{\partial t[M'_{n*}/x]}{\partial x_n}|_{M'_n})}$$

$$\frac{M_0 \twoheadrightarrow M'_0, \forall j \in [1,n], M_{1j} \twoheadrightarrow M'_{1j}, M_{2j} \twoheadrightarrow M'_{2j}, (M'_{11}..., M'_{1j-1}, x_j, M'_{2j+1}..., M'_{2n}) is\ written\ as\ M'_{j*}}{\int_{(M_{11}, M_{12}, ...M_{1n})}^{(M_{21}, M_{22}, ..., M_{2n})} M_0 dx \ \twoheadrightarrow \int_{M'_{11}}^{M'_{21}} \pi_1(M'_0[M'_{1*}/x])dx_1 \oplus ... \oplus \int_{M'_{1n}}^{M'_{2n}} \pi_n(M'_0[M'_{n*}/x])dx_n}$$

$$\frac{M_0 \twoheadrightarrow M'_0, M_1 \twoheadrightarrow M'_1, M_2 \twoheadrightarrow M'_2, M_1, M_2 : \mathsf{B}}{\int_{M_1}^{M_2} \lambda y : T_2.M_0 dx \ \twoheadrightarrow \lambda y : T_2. \int_{M'_1}^{M'_2} M'_0 dx}$$

$$\frac{N \twoheadrightarrow N', M \twoheadrightarrow M', M, N : \mathsf{B}, \forall \mathsf{j} \in [1, \mathsf{n}], \mathsf{M_j} \twoheadrightarrow \mathsf{M'_j}}{\int_M^N (M_1, M_2, ...M_n)dx \twoheadrightarrow (\int_{M'}^{N'} M'_1 dx, \int_{M'}^{N'} M'_2 dx, ..., \int_{M'}^{N'} M'_n dx)}$$

$$\frac{N \twoheadrightarrow N', M \twoheadrightarrow M', N : \mathsf{B}}{(\lambda x : T.M) * N \twoheadrightarrow \lambda x : T.(M' * N')}$$

$$\frac{\forall j \in [1,n], M_j \twoheadrightarrow M'_j, N \twoheadrightarrow N', N : \mathsf{B}}{(M_1, M_2, ...M_n) * N \twoheadrightarrow (M'_1 * N', M'_2 * N', ...M'_n * N')}$$

$$\frac{\forall j \in [1,n], M_{1j} \twoheadrightarrow M'_{1j}, M_{2j} \twoheadrightarrow M'_{2j}}{(M_{11}, M_{12}, ...M_{1n}) * (M_{21}, M_{22}, ...M_{2n}) \twoheadrightarrow M'_{11} * M'_{21} \oplus M'_{12} * M'_{22} \oplus ... \oplus M'_{1n} * M'_{2n}}$$

$$\frac{M \twoheadrightarrow M'}{fix\ M \twoheadrightarrow M'\ (fix\ M'), fix\ M \twoheadrightarrow fix\ M'}$$

▶ **Lemma 31** (Preservation of $\twoheadrightarrow$). *if $N : B, N \twoheadrightarrow N'$, then $N' : B$.*

Proof. if we name the one-step relation of our calculus as ρ, and its transitive closure as ρ^*, then we have $\twoheadrightarrow \subseteq \rho^*$. So we have $N \rho^* N'$. Notice that we have the preservation property of our calculus, thus we have $N' : B$.

◀

▶ **Lemma 32** ($\twoheadrightarrow$ under substitution). *$M \twoheadrightarrow M'$, $N \twoheadrightarrow N'$, then we have $M[N/x] \twoheadrightarrow M'[N'/x]$*

Proof. Induction on $M \twoheadrightarrow M'$

- Case $M \twoheadrightarrow M$, make induction on the form of M.
  - Subcase c, Then c[N/x] = c = c[N'/x], using $M \twoheadrightarrow M$ we have $M[N/x] \twoheadrightarrow M[N'/x]$.
  - Subcase $(t_1, t_2, ..., t_n)$, using induction we have $t_i[N/x] \twoheadrightarrow t_i[N'/x]$, Then using $\forall i, M_i \twoheadrightarrow M'_i \Rightarrow (M_1, M_2, .., M_n) \twoheadrightarrow (M'_1, M'_2, .., M'_n)$ we have $M \twoheadrightarrow M$ we have $M[N/x] \twoheadrightarrow M[N'/x]$.
    Using the same technique, we can prove the subcase of $t \oplus t$, $t \ominus t$, $t * t$, $\lambda x : T.t$, $t\ t$, $\frac{\partial t}{\partial x}|_t$, $\int_t^t tdx$, $\pi_j(t)$, $\int_{M_1}^{M_2} M_3 dx \twoheadrightarrow \int_{M'_1}^{M'_2} M'_3 dx, inl/inr\ M, case\ inr\ M_1\ of\ inl\ x_1 \Rightarrow M_2 |\ inr\ x_2 \Rightarrow M_3$.
  - Subcase variable y, if y = x then y[N/x] = N, y[N'/x] = N', then $y[N/x] \twoheadrightarrow y[N'/x]$, if y is not x then same as the subcase c.

The rest cases can be divided into three categories.

- Case relation based on the relation of subterms.
  - Subcase $M\ N \twoheadrightarrow M'\ N'$, using induction we have $M[K/x] \twoheadrightarrow M'[K'/x]$, $N[K/x] \twoheadrightarrow N'[K'/x]$, using $M \twoheadrightarrow M', N \twoheadrightarrow N' \Rightarrow M\ N \twoheadrightarrow M'\ N'$ we have $(M\ N)[K/x] \twoheadrightarrow (M'\ N')[K'/x]$.
  - Subcase $M \oplus N \twoheadrightarrow M' \oplus N'$, $M \ominus N \twoheadrightarrow M' \ominus N'$, $M * N \twoheadrightarrow M' * N'$, $\frac{\partial M}{\partial x}|_N \twoheadrightarrow \frac{\partial M'}{\partial x}|_{N'}$, $(M_1, M_2, .., M_n) \twoheadrightarrow (M'_1, M'_2, .., M'_n)$, $\lambda x : T.M \twoheadrightarrow \lambda x : T.M', \pi_j(M) \twoheadrightarrow \pi_j(M')$, $fix\ M \twoheadrightarrow fix\ M'$, $inl/inr\ M \twoheadrightarrow inl/inr\ M'$: same as $M\ N \twoheadrightarrow M'\ N'$.

- Case reduction changes the structure
  - Subcase $(M_{11}, M_{12}, ...M_{1n}) \oplus (M_{21}, M_{22}, ...M_{2n}) \twoheadrightarrow (M'_{11} \oplus M'_{21}, M'_{12} \oplus M'_{22}, ...M'_{1n} \oplus M'_{2n})$, using induction we have $\forall i \in [1, 2], \forall j \in [1, n], M_{ij}[K/x] \twoheadrightarrow M'_{ij}[K'/x]$, so we have
    $((M_{11}, M_{12}, ...M_{1n}) \oplus (M_{21}, M_{22}, ...M_{2n}))[K/x] \twoheadrightarrow (M'_{11} \oplus M'_{21}, M'_{12} \oplus M'_{22}, ...M'_{1n} \oplus M'_{2n})[K'/x]$.
  - Subcase $(M_{11}, M_{12}, ...M_{1n}) \ominus (M_{21}, M_{22}, ...M_{2n}) \twoheadrightarrow (M'_{11} \ominus M'_{21}, M'_{12} \ominus M'_{22}, ...M'_{1n} \ominus M'_{2n})$, $\frac{\partial (M_1, M_2, .., M_n)}{\partial x}|_N \twoheadrightarrow (\frac{\partial M'_1}{\partial x}|_{N'}, \frac{\partial M'_2}{\partial x}|_{N'}, ..., \frac{\partial M'_n}{\partial x}|_{N'})$, $\frac{\partial \lambda y : T.M}{\partial x}|_N \twoheadrightarrow \lambda y : T. \frac{\partial M'}{\partial x}|_{N'}$, $\int_{M_1}^{M_2} \lambda y : T_2.M_0 dx \twoheadrightarrow \lambda y : T_2. \int_{M'_1}^{M'_2} M'_0 dx$, $\int_M^N (M_1, M_2, ...M_n)dx \twoheadrightarrow (\int_{M'}^{N'} M'_1 dx, \int_{M'}^{N'} M'_2 dx, ..., \int_{M'}^{N'} M'_n dx)$, $(\lambda x : T.M) * N \twoheadrightarrow \lambda x : T.(M' * N')$, $(M_1, M_2, ...M_n) * N \twoheadrightarrow (M'_1 * N', M'_2 * N', ...M'_n * N')$, $(M_{11}, M_{12}, ...M_{1n}) * (M_{21}, M_{22}, ...M_{2n}) \twoheadrightarrow M'_{11} * M'_{12} \oplus M'_{21} * M'_{22} \oplus ... \oplus M'_{1n} * M'_{2n}$, $fix\ M \twoheadrightarrow M'\ (fix\ M')$: same as $(M_{11}, M_{12}, ...M_{1n}) \oplus (M_{21}, M_{22}, ...M_{2n}) \twoheadrightarrow (M'_{11} \oplus M'_{21}, M'_{12} \oplus M'_{22}, ...M'_{1n} \oplus M'_{2n})$.

- Case reduction involves substitution
  - Subcase $(\lambda x : T.M)N \twoheadrightarrow M'[N'/x]$, by induction hypothesis, we have $M[K/y] \twoheadrightarrow M'[K'/y]$, $N[K/y] \twoheadrightarrow N'[K'/y]$, thus $((\lambda x : T.M)N)[K/y] = ((\lambda x : T.M[K/y])N[K/y]) \twoheadrightarrow M'[K'/y]([(N'[K'/y])/x)$, and we have $M'[N'/x][K'/y] = M'[K'/y]([(N'[K'/y])/x)$, Therefore we prove the case.

- Subcase$\forall i, (M_1', M_2' ..., M_{i-1}', x_i, M_{i+1}' ..., M_n') is\ written\ as\ M_{i*}', \frac{\partial M_0}{\partial x}|_{(M_1,M_2,...,M_n)} \twoheadrightarrow (\frac{\partial M_0'[M_{1*}'/x]}{\partial x_1}|_{M_1'}, \frac{\partial M_0[M_{2*}'/x]}{\partial x_2}|_{M_2'}, ..., \frac{\partial t[M_{n*}'/x]}{\partial x_n}|_{M_n'})$
  Notice that $M_0'[M_{i*}'/x][K'/y] = M_0'[K'/y][(M_{i*}'[K'/y])/x] = (M_0'[K'/y])[(M_1'[K'/y], M_2'[K'/y]..., M_{i-1}'[K'/y], x_i[K'/y], M_{i+1}'[K'/y], ..., M_n'[K'/y])/x] = (M_0'[K'/y])[((M'[K'/y])_{i*}')/x]$ Using induction, we know that $M_i[K/y] \twoheadrightarrow M_i'[K'/y]$, so we prove the case.
- Subcase $(\lambda x : T.M) \oplus (\lambda y : T.N) \twoheadrightarrow \lambda x : T.M' \oplus N'[y/x]$, $(\lambda x : T.M) \ominus (\lambda y : T.N) \twoheadrightarrow \lambda x : T.M' \ominus N'[y/x]$: same as $(\lambda x : T.M)N \twoheadrightarrow M'[N'/x]$.
- Subcase $\int_{(M_{11},M_{12},...M_{1n})}^{(M_{21},M_{22},...,M_{2n})} M_0 dx \twoheadrightarrow \int_{M_{11}'}^{M_{21}'} \pi_1(M_0'[M_{1*}'/x])dx_1 \oplus ... \oplus \int_{M_{1n}'}^{M_{2n}'} \pi_n(M_0'[M_{n*}'/x])dx_n$, $case\ inl\ M_1\ of\ inl\ x_1 \Rightarrow M_2|\ inr\ x_2 \Rightarrow M_3 \twoheadrightarrow M_2'[M_1'/x_1]$,$case\ inr\ M_1\ of\ inl\ x_1 \Rightarrow M_2|\ inr\ x_2 \Rightarrow M_3 \twoheadrightarrow M_3'[M_1'/x_2]$: same as $\forall i, (M_1', M_2', M_{i-1}', x_i, M_{i+1}' ..., M_n')$is written as M_{i*}' ,$\frac{\partial M_0}{\partial x}|_{(M_1,M_2,...,M_n)} \twoheadrightarrow (\frac{\partial M_0'[M_{1*}'/x]}{\partial x_1}|_{M_1'}, \frac{\partial M_0[M_{2*}'/x]}{\partial x_2}|_{M_2'}, ..., \frac{\partial t[M_{n*}'/x]}{\partial x_n}|_{M_n'})$.

Thus we complete the proof.

◀

▶ **Lemma 33** (diamond property). *for $M \twoheadrightarrow M_1, M \twoheadrightarrow M_2$, there exists a M_3, that $M_1 \twoheadrightarrow M_3, M_2 \twoheadrightarrow M_3$*

Proof. we make induction on the case of $M \twoheadrightarrow M_1$.

- Case $M \twoheadrightarrow M$,Then we choose M_3 as M_2.
- Case $M\ N \twoheadrightarrow M'\ N'$
  - Subcase $M \twoheadrightarrow M_2$ as $M \twoheadrightarrow M$
    Then we choose M_3 as M_1.
  - Subcase $M \twoheadrightarrow M_2$ as $M\ N \twoheadrightarrow M''\ N''$ ①
    Then we use induction hypothesis, we have M^* that $M' \twoheadrightarrow M^*$,$M'' \twoheadrightarrow M^*$,we have N^* that $N' \twoheadrightarrow N^*$,$N'' \twoheadrightarrow N^*$, so we choose M_3 as $M^*\ N^*$.
  - Subcase $M \twoheadrightarrow M_2$ as $M = \lambda x : T.P$, $(\lambda x : T.P)N \twoheadrightarrow P''[N''/x]$.②
    Then we first have that $M = \lambda x : T.P$ Then $M' = \lambda x : T.P'$, So we choose $M_3 = P^*[N^*/x]$.
- Case $M \oplus N \twoheadrightarrow M' \oplus N'$
  - Subcase $M \twoheadrightarrow M_2$ as $M \twoheadrightarrow M$
    Then we choose M_3 as M_1.
  - Subcase $M \twoheadrightarrow M_2$ as $M \twoheadrightarrow M'', N \twoheadrightarrow N'' \Rightarrow M \oplus N \twoheadrightarrow M'' \oplus N''$: same as ①.
  - Subcase $M \twoheadrightarrow M_2$ as $(\lambda x : T.M) \oplus (\lambda y : T.N) \twoheadrightarrow \lambda x : T.M' \oplus N'[y/x]$, Then $M \twoheadrightarrow M_1$ must be $(\lambda x : T.M) \oplus (\lambda y : T.N) \twoheadrightarrow (\lambda x : T.M'') \oplus (\lambda y : T.N'')$, Then we choose M_3 to be $\lambda x : T.M^* \oplus N^*[y/x]$.
  - Subcase $M \twoheadrightarrow M_2$ as $(M_{11}, M_{12}, ...M_{1n}) \oplus (M_{21}, M_{22}, ...M_{2n}) \twoheadrightarrow (M_{11}' \oplus M_{21}', M_{12}' \oplus M_{22}', ...M_{1n}' \oplus M_{2n}')$: same as $(\lambda x : T.M) \oplus (\lambda y : T.N) \twoheadrightarrow \lambda x : T.M' \oplus N'[y/x]$.

All the other cases are similar the case of application and $\oplus$, except that we may have more subcases on these cases, but the extra subcases are all similar to ②.

◀

▶ **Lemma 34** (Confluence). *One term has at most one normal form*

Proof. The relation $\twoheadrightarrow$ has the diamond property, and reduction relation ρ satisfy that $\rho \subseteq \twoheadrightarrow \subseteq \rho^*$, Also notice that $\twoheadrightarrow^*$ has the diamond property, and $\twoheadrightarrow^* = \rho^*$, so the relation ρ^* has the diamond property, There comes the confluence. ◀

A.4 Strong normalization

Here we write $\rightsquigarrow$ as ρ^*.

▶ **Lemma 35** (existence of ν). *t is strongly normalisable iff there is a number $\nu(t)$ which bounds the length of every normalisation sequence beginning with t. (Proofs and Types P27)*

▶ **Definition 36.** *We define a set $\boldsymbol{RED}_T$ by induction on the type T.*

1. For t of base type, t is reducible iff it is strongly normalisable.

2. For t of type $(T_1, T_2, ..., T_n)$, t is reducible iff $\forall j, \pi_j(t)$ is reducible.

3. For t of type $U \to V$, t is reducible iff, for all reducible u of type U, t u is reducible of type V .

4. For t of type $T_1 + T_2$, t is reducible iff, case t of inl $x_1 \Rightarrow 0$ | inr $x_2 \Rightarrow 0$is reducible term of base type.

▶ **Definition 37.** *t is neutral if t is not of the form $(t_1, t_2, ..., t_n)$ or $\lambda x : T.t$ or inl/inr t*

We would verify the following 3 properties by induction on types.

(CR 1) If t $\in$ $\mathbf{RED}_T$, then t is strongly normalisable.

(CR 2) If t $\in$ $\mathbf{RED}_T$ and t$\rightsquigarrow$t', then t' $\in$ $\mathbf{RED}_T$.

(CR 3) If t is neutral, and whenever we convert a redex of t we obtain a term t'$\in$ $\mathbf{RED}_T$, then t$\in$ $\mathbf{RED}_T$.

- Case base type
  (CR 1) is a tautology.
  (CR 2) If t is strongly normalisable then every term t'to which t reduces is also.
  (CR 3) A reduction path leaving t must pass through one of the terms t' , which are strongly normalisable, and so is finite. In fact, it is immediate that ν(t) is equal to the greatest of the numbers ν(t')+1, as t' varies over the (one-step) conversions of t.
- Case tuple type
  (CR 1) Suppose that t, of type $(T_1, T_2, ..., T_n)$, is reducible; then $\pi_1(t)$ is reducible and by induction hypothesis (CR 1) for T_1, $\pi_1(t)$ is strongly normalisable. Moreover, $\nu(t) \leq \nu(\pi_1(t))$. since to any reduction sequence t, t_1, t_2, . . ., one can apply $\pi_1()$ to construct a reduction sequence $\pi_1(t)$, $\pi_1(t_1)$, $\pi_1(t_2)$... (in which the $\pi_1()$ is not reduced). So $\nu(t)$ is finite, and t is strongly normalisable.
  (CR 2) If $t \rightsquigarrow t'$, then $\forall j, \pi_j(t) \rightsquigarrow \pi_j(t')$. induction hypothesis for type T_j on (CR 2), we have $\forall j, \pi_j(t')$ is reducible, so t' is reducible
  (CR 3) Let t be neutral and suppose all the t' one step from t are reducible. Applying a conversion inside $\pi_j(t)$, the result is a $\pi_j(t')$, since $\pi_j(t)$ cannot itself be a redex (t is not a tuple), and $\pi_j(t')$ is reducible, since t' is. But as $\pi_j(t)$ is neutral, and all the terms one step from $\pi_j(t)$ are reducible, the induction hypothesis (CR 3) for T_j ensures that $\pi_j(t)$ is reducible. so t is reducible.

- Case arrow type
  (CR 1) If t is reducible of type U→V , let x be a variable of type U; And we have $\nu(t) \leq \nu(t\ x)$
  (CR 2) If $t \rightsquigarrow t'$, and t is reducible, take u reducible of type U; then t u is reducible and $t\ u \rightsquigarrow t'\ u$ The induction hypothesis (CR 2) for V gives that $t'\ u$ is reducible. So t' is reducible.
  (CR 3) Let t be neutral and suppose all the t' one step from t are reducible. Let u be a reducible term of type U; we want to show that t u is reducible. By induction hypothesis (CR 1) for U, we know that u is strongly normalisable; so we can reason by induction on $\nu(u)$.
  In one step, t u converts to
  - 1.t' u with t' one step from t; but t'is reducible, so t' u is.
  - 2.t u', with u' one step from u. u' is reducible by induction hypothesis(CR 2) for U, and $\nu(u') < \nu(u)$; so the induction hypothesis for u' tells us that t u' is reducible.
  - 3. There is no other possibility, for t u cannot itself be a redex (t is not of the form $\lambda x : T.t$).
- Case sum type
  (CR 1) If t is reducible of type $T_1 + T_2$, Then we have $\nu(t) \leq \nu(case\ t\ of\ inl\ x_1 \Rightarrow 0 | inr\ x_2 \Rightarrow 0)$
  (CR 2) same as tuple type.
  (CR 3) same as arrow type.

▶ **Lemma 38.** *if $t_1, t_2, ..., t_n$ are reducible terms, then so $(t_1, t_2, ..., t_n)$ is*

Proof. Because of (CR 1), we can reason by induction on $\nu(t_1) + \nu(t_2) + ... + \nu(t_n)$ to show that $\pi_j(t_1, t_2, ..., t_n)$, is reducible. This term converts to

- 1. t_j, then it is reducible.
- 2.$(t_1, ..., t_{k-1}, t'_k, t_{k+1}, ..., t_n)$, based on induction, it is reducible.

◀

▶ **Lemma 39.** *if for all reducible u of type U, t[u/x] is reducible, then so is $\lambda x : T.t$.*

Proof. To show $\lambda x : T.t\ u$ is reducible, We make reductions on $\nu(u) + \nu(t)$, $\lambda x : T.t\ u$ can be reduced to

- 1. $t[u/x]$, then it is reducible.
- 2. $(\lambda x : T.t')\ u$ or $(\lambda x : T.t)\ u'$, based on induction we know it is reducible.

◀

▶ **Lemma 40.** *if t is reducible, then so is $inl/inr\ t$.*

Proof. same as the case $\lambda x : T.t$. ◀

▶ **Lemma 41.** *if for all reducible t_1 and t_2 of type T_1 and T_2, we have $t_3[t_1/x_1]$ and $t_4[t_2/x_2]$ are reducible, and t is reducible term of type $T_1 + T_2$, then so is $case\ t\ of\ inl\ x_1 \Rightarrow t_3 \mid inr\ x_2 \Rightarrow t_4$.*

Proof. same as the case $\lambda x : T.t$. ◀

▶ **Lemma 42.** *if t_1 and t_2 are reducible terms of T, then so is $t_1 \oplus t_2$.*

Proof. We prove this by induction on type.

- Case base type, then it can only be reduced to $t_1' \oplus t_2'$, so $\nu(t_1 \oplus t_2) = \nu(t_1) + \nu(t_2)$, Therefore it is strongly normalisable, and thus reducible.
- Case $(T_1, T_2, ..., T_n)$, we make induction on $\nu(t_1) + \nu(t_2)$, $t_1 \oplus t_2$ can be reduced to.
  - 1.Subcase $(t_{11}, t_{12}, ...t_{1n}) \oplus (t_{21}, t_{22}, ...t_{2n}) \to (t_{11} \oplus t_{21}, t_{12} \oplus t_{22}, ...t_{1n} \oplus t_{2n})$ Because t_1 and t_2 si reducible, then $\forall i \forall j, t_{ij}$ is reducible, based on induction on types, we have $\forall j, t_{1j} \oplus t_{2j}$ Thus, $(t_{11} \oplus t_{21}, t_{12} \oplus t_{22}, ...t_{1n} \oplus t_{2n})$ is reducible.
  - 2.Subcase $(t_1' \oplus t_2)$ or $(t_1 \oplus t_2')$. Based on induction, we know it is reducible.
- Case $A \to B$ for all reducible term u of type A, we make induction on $\nu(t_1)+\nu(t_2)+\nu(u)$.
  - 1. Subcase $(\lambda x : T.t_1) \oplus (\lambda y : T.t_2) \to \lambda x : T.t_1 \oplus (t_2[x/y])$ notice that for all reducible u of type A $(t_1 \oplus t_2[x/y])[u/x]$, notice that this term is equal to $(t_1[u/x] \oplus t_2[u/y])$, because $(\lambda x : T.t_1)$ is a reducible term, then so is $t_1[u/x]$. Because $t_1[u/x]$ and $t_2[u/y]$ are reducible terms based on induction. So we have $\lambda x : T.t_1 \oplus (t_2[x/y])$'s reducibility.
  - 2 Subcase $(t_1' \oplus t_2)\ u$ or $(t_1 \oplus t_2')\ u$ or $(t_1 \oplus t_2)\ u'$, based on induction we can prove the case.

◀

▶ **Lemma 43.** *if t_1 and t_2 are reducible terms of T, then so is $t_1 \ominus t_2$.*

Proof. Same as $\oplus$. ◀

▶ **Lemma 44.** *if t_1 and t_2 are reducible terms of $\frac{\partial T_1}{\partial T_2}$ and T_2, then so is $t_1 * t_2$.*

Proof. We prove this by induction on type.

- Case T_1: base type, T_2: base type: same as the case of $\oplus$.
- Case T_1: $(T_1, T_2, ..., T_n)$, T_2: base type: same as the case of $\oplus$.
- Case T_1: $A \to B$, T_2: base type: same as the case of $\oplus$.
- Case T_1: $(T_1, T_2, ..., T_n)$, T_2: $(T_1', T_2', ..., T_n')$
  Suppose $t_1 : (t_{11}, t_{12}, ...t_{1n}), t_2 : (t_{21}, t_{22}, ...t_{2n})$, we make induction on $\nu(t_1) + \nu(t_2)$.
  - Subcase$t_1 * t_2 \to (t_{11} * t_{21}) \oplus (t_{12} * t_{22}) \oplus ... \oplus (t_{1n} * t_{2n})$: Because t_1, t_2 is reducible, then so is $\forall i \forall j, t_{ij}$, based on induction on types we have $\forall j, t_{1j} * t_{2j}$ is reducible, then so is $(t_{11} * t_{21}) \oplus (t_{12} * t_{22}) \oplus ... \oplus (t_{1n} * t_{2n})$.
  - Subcase $t_1' * t_2$ or $t_1 * t_2'$, based on induction we know it is reducible.

◀

▶ **Lemma 45.** *if t_1 and t_2 are reducible terms of T_1 and T_2, and for all reducible u of type T_2, we have $t_1[u/x]$ is reducible then so is $\frac{\partial t_1}{\partial x}|_{t_2}$.*

Proof. we prove this by induction on type.

- Case T_1: $(T_1, T_2, ..., T_n)$ or $A \to B$ or base type, T_2:base type. Same as the case *.
- Case T_1: $(T_1, T_2, ..., T_n)$, T_2: $(T_1', T_2', ..., T_n')$, we make induction on $\nu(t_1) + \nu(t_2)$.

- Subcase$\forall i, (t_1, t_2..., t_{i-1}, x_i, t_{i+1}..., t_n) is\ written\ as\ t_{i*}$,
  $\frac{\partial t}{\partial x}|_{(t_1,t_2,...,t_n)} \to (\frac{\partial t[t_{1*}/x]}{\partial x_1}|_{t_1}, \frac{\partial t[t_{2*}/x]}{\partial x_2}|_{t_2}, ..., \frac{\partial t[t_{n*}/x]}{\partial x_n}|_{t_n})$, note that t_{i*} is reducible so based on induction we have $\frac{\partial t[t_{j*}/x]}{\partial x_j}|_{t_j}$ is reducible. Note that this induction is based on the hypothesis $(t[t_{j*}/x])[u/x_j]$ is reducible for all the reducible u of type T_j', and $(t[t_{j*}/x])[u/x_j] = (t[(t_{j*}[u/x_j])/x])$ because t has no occurrence of x_j, and it is easy to show that $(t_{j*}[u/x_j])$ is a reducible term of type T_2, so we finish the induction, then we have
  $(\frac{\partial t[t_{1*}/x]}{\partial x_1}|_{t_1}, \frac{\partial t[t_{2*}/x]}{\partial x_2}|_{t_2}, ..., \frac{\partial t[t_{n*}/x]}{\partial x_n}|_{t_n})$ is reducible.
- Subcase $\frac{\partial t_1'}{\partial x}|_{t_2}$ or $\frac{\partial t_1}{\partial x}|_{t_2'}$, based on induction we have the proof.

◀

▶ **Lemma 46.** *if t_1,t_2 and t_3 are reducible terms of T_1, T_1 and T_2, and for all reducible u of type T_1, we have $t_3[u/x]$ is reducible then so is $\int_{t_1}^{t_2} t_3 dx$.*

Proof. Same as the case of $\frac{\partial}{\partial x}|_{...}$ ◀

▶ **Lemma 47.** *if t_1,t_2 and t_3 are reducible terms of $T_1 + T_2$, T and T, and for all reducible u_1 of type T_1, u_2 of type T_2, we have $t_2[u_1/x_1]$ and $t_3[u_2/x_2]$ are reducible then so is case t_1 of inl $x_1 \Rightarrow t_2$ | inr $x_2 \Rightarrow t_3$.*

Proof. Same as the case of $\frac{\partial}{\partial x}|_{...}$ ◀

▶ **Lemma 48.** *Let t be any term (not assumed to be reducible), and suppose all the free variables of t are among $x_1, ..., x_n$ of types $U_1, ..., U_n$. If $u_1, ..., u_n$ are reducible terms of types $U_1, ..., U_n$ then $t[u_1/x_1, ..., u_n/x_n]$ is reducible.*

Proof. By induction on t. We write t[$\underline{u}$/$\underline{x}$] for $t[u_1/x_1, ..., u_n/x_n]$.

- 1. t is x_i, then it t[u/x_i] is reducible.
- 2. t is c, then t_i has no free variable, and c itself is reducible, so it is reducible.
- 3. t is $(t_1, t_2, ..., t_n)$, based on induction we prove $t_j[\underline{u}/\underline{x}]$ is reducible, based on the lemma we know it is reducible.
- 4. t is $t \oplus t$, $t \ominus t$, $t * t$,*inl*/*inr* t or $\pi_j(t)$: same as the case $(t_1, t_2, ..., t_n)$.
- 5. t is $\lambda y : T.t$, by induction we have t[$\underline{u}$/$\underline{x}$,v/y] is reducible, then by lemma we have $\lambda y : T.(t[\underline{u}/\underline{x}])$ is reducible, so $(\lambda y : T.t)[\underline{u}/\underline{x}]$ is reducible.
- 6. t is $\frac{\partial t}{\partial x}|_t$, *case t of inl $x_1 \Rightarrow t_1$ | inr $x_2 \Rightarrow t_2$* or $\int_t^t tdx$: same as the case $\lambda y : T.t$.

◀

▶ **Theorem 49.** *All terms are reducible.*

▶ **Corollary 50.** *All terms are strongly normalisable.*

B Appendix-some useful lemma

▶ **Lemma 51.** if $t_1\rho^*t_1', t_2\rho^*t_2', t_1[t_2/x]\rho^*t_1'[t_2'/x]$

Proof. Using the confluence property, it is easy to see. ◀

▶ **Lemma 52.** if $t_1 = t_1', t_2 = t_2'$, then $t_1 \oplus t_2 = t_1' \oplus t_2'$

Proof. if t_1 or t_2 is not closed, then we use the substitution $[u_1/x_1, ..., u_n/x_n]$ to make it closed. For simplicity of notation, we just use t_1 and t_2 to be the closed-term of themselves.

Based on the equality defintion, we can assume that t_1, t_2, t_1', t_2' are all normal forms and we prove this by induction on type.

- Case t_1 is of base type. then t_2, t_1' and t_2' have to be base type to be well-typed. and for base type normal forms, we have $t_1 \oplus t_2 = t_1' \oplus t_2'$.
- Case t_1 is $A \to B$ type, let's suppose $t_1 : \lambda x : T.t_3$, $t_1' : \lambda x' : T.t_3'$, $t_2 : \lambda y : T.t_4$, $t_2' : \lambda y' : T.t_4'$.
  if t_1 or t_2 or t_1' or t_2' 's normal form are not $\lambda x : T.t$, then we know their normal form are all interpretable in base type, thus we have $t_1 \oplus t_2 = t_1' \oplus t_2'$.
  Else for all u
  $$\begin{aligned}&(t_1 \oplus t_2)\ u\\ &\quad= (\lambda x : T.t_3 \oplus \lambda y : T.t_4)\ u\\ &\quad= (\lambda x : T.t_3 \oplus t_4[x/y])\ u\\ &\quad= t_3[u/x] \oplus t_4[u/y]\end{aligned}$$
  Similarly, we have $(t_1' \oplus t_2')\ u' = t_3'[u/x'] \oplus t_4'[u/y']$.
  And notice that because $t_1 = t_1'$, so $t_1\ u = t_1'\ u$, so $t_3[u/x] = t_3'[u/x']$, based on induction of type B, we have $t_3[u/x] \oplus t_4[u/y] = t_3'[u/x'] \oplus t_4'[u/y']$, so we prove the case.
- Case t_1 is of type $(T_1, T_2, .., T_n)$. Then we suppose $t_1 : (t_{11}, t_{12}, .., t_{1n})$, $t_1' : (t_{11}', t_{12}', .., t_{1n}')$, $t_2 : (t_{21}, t_{22}, .., t_{2n})$, $t_2' : (t_{21}', t_{22}', .., t_{2n}')$
  Then
  $$\begin{aligned}&t_1 \oplus t_2\\ &= (t_{11}, t_{12}, .., t_{1n}) \oplus (t_{21}, t_{22}, .., t_{2n})\\ &= (t_{11} \oplus t_{21}, t_{12} \oplus t_{22}, .., t_{1n} \oplus t_{2n})\end{aligned}$$
  Similarly we have $t_1' \oplus t_2' = (t_{11}' \oplus t_{21}', t_{12}' \oplus t_{22}', .., t_{1n}' \oplus t_{2n}')$, and based on induction we have $\forall j, t_{1j} \oplus t_{2j} = t_{1j}' \oplus t_{2j}'$, so we have $t_1 \oplus t_2 = t_1' \oplus t_2'$.
- Case t_1 is of type $T_1 + T_2$. This case is impossible because it is not well-typed.

◀

▶ **Lemma 53.** *For a term t, for any subterm s, if the term s'=s, then t[s'/s]=t. (We only substitute the subterm s, but not other subterms same as s)*

Proof. We prove by induction, we first substitute for all the free variables in t. then

$$\begin{aligned}&t[s'/s][u_1/x_1, ..., u_n/x_n]\\ &\quad= t[u_1/x_1, ..., u_n/x_n][s'[u_1/x_1, ..., u_n/x_n]/s[u_1/x_1, ..., u_n/x_n]]\end{aligned}$$

notice that $s'[u_1/x_1, ..., u_n/x_n] = s[u_1/x_1, ..., u_n/x_n]$ because $s' = s$, and we just substitute for some of the free variables in s' and s. So we only need to prove that for a closed-term t, for any subterm s, if the term s'=s, then t[s'/s]=t.

And notice that if we choose the subterm s to be the t itself, then we have t[s'/s]=s'=s=t, And we prove the case. So we next make induction on the form of t.

- Case t is $(t_1, t_2, ..., t_n)$

  Using induction, we know $t_i[s'/s] = t_i$, and we want to prove

  $$\begin{aligned}&(t_1, t_2, ..., t_n)\\&\quad= (t_1[s'/s], t_2[s'/s], ..., t_n[s'/s])\end{aligned}$$

  And because we have the transitive property of equality, then we just reduce both of them to normal forms, then by definition we know they equal to each other, thus we prove the case.

  Using the same technique, we can prove the case $\lambda x : T.t$ and $inl/int\ t$.

- Case t is c

  Then it has no subterm except c itself, if s'=c, then $c[s'/c] = s'$, thus we prove the case.

  Using the same technique, we can prove the case x.

- Case t is $t_1 \oplus t_2$

  Using induction we have $t_1[s'/s] = t_1$,$t_2[s'/s] = t_2$, and we have proved the Lemma 52 that if $t_1 = t'_1, t_2 = t'_2$, then $t_1 \oplus t_2 = t'_1 \oplus t'_2$, thus we prove the case.

  Using the same technique, we can prove the case $t \ominus t$, $t * t$, $\pi_j(t)$.

- Case t is $t_1\ t_2$

  We want to prove if $t_1 = t'_1, t_2 = t'_2$, then $t_1\ t_2 = t'_1\ t'_2$.

  By definition we know if $t_1 = t'_1$, then $t_1\ t'_2 = t'_1\ t'_2$.

  Then we prove $t_1\ t'_2 = t_1\ t_2$, Using confluence property, we can reduce the t_1 to $\lambda x : A.t$ or a **nb**. If it is the former case, then we using induction we have $t[t'_2/x] = t[t_2/x]$ .Thus we have $t_1\ t'_2 = t_1\ t_2$. If it is the latter case, then t_2's normal form can be is interpretable, and on base type interpretation we have if x=x', then f(x)=f(x'), Thus we prove the case.

- Case t is $\frac{\partial t_1}{\partial x}|_{t_2}$

  if t_2 is base type, then we can use the same technique of that we prove the case $t_1 * t_2$, if t_2 is of type $(T_1, T_2, ..., T_n)$, we can reduce the t_2 and t'_2 to the normal forms $(t_1, t_2, ..., t_n)$ and $(t'_1, t'_2, ..., t'_n)$, and then we have

  $$\frac{\forall i, (t_1, t_2 ..., t_{i-1}, x_i, t_{i+1} ..., t_n) is\ written\ as\ t_{i*}}{\frac{\partial t}{\partial x}|_{(t_1,t_2,...,t_n)} \to (\frac{\partial t[t_{1*}/x]}{\partial x_1}|_{t_1}, \frac{\partial t[t_{2*}/x]}{\partial x_2}|_{t_2}, ..., \frac{\partial t[t_{n*}/x]}{\partial x_n}|_{t_n})}$$

  , using induction we have $t[t_{j*}/x] = t[t'_{j*}/x]$, then based on induction we have $\frac{\partial t[t_{j*}/x]}{\partial x_j}|_{t_j} = \frac{\partial t[t'_{j*}/x]}{\partial x'_j}|_{t'_j}$, Thus we prove the case.

  Using the same technique, we can prove the case $\int_{t_1}^{t_2} t_3 dx$ adn $case\ t\ of\ inl\ x_1 \Rightarrow t_1 \mid inr\ x_2 \Rightarrow t_2$.

Thus we prove the lemma.

◀

▶ **Lemma 54.** If $t_1 * (t_2 \oplus t_3)$ and $(t_1 * t_2) \oplus (t_1 * t_3)$ are normalizable, then $t_1 * (t_2 \oplus t_3) = (t_1 * t_2) \oplus (t_1 * t_3)$

Proof. if t_1, t_2 and t_3 are not closed, then we use the substitution $[u_1/x_1, ..., u_n/x_n]$ to make it closed. For simplicity of notation, we just use t_1, t_2 and t_3 to be the closed-term of themselves.

Because of the confluence and strong normalization property of the system, we can assume that t_1, t_2, t_3 are all normal formss and we prove this by induction on type.

- Case:t_1, t_2 and t_3 are of base type. then based on base type interpretation, we have $t_1 * (t_2 \oplus t_3) = (t_1 * t_2) \oplus (t_1 * t_3)$.
- Case:t_1 is of type $A \to B$, t_2 and t_3 are of base type. Suppose t_1 is $\lambda x : A.t$.
  If t_1 's normal form is not $\lambda x : A.t$, then we notice that $t_1 = \lambda x : A.t_1\ x$, use the Lemma 16, we know we can use $\lambda x : A.t_1\ x$ to substitute for t_1, Thus we can suppose $\lambda x : A.t$.
  Then we have for all u of type A,

$$\begin{aligned} & t_1 * (t_2 \oplus t_3)\ u \\ & \quad = (\lambda x : A.t) * (t_2 \oplus t_3)\ u \\ & \quad = \lambda x : A.(t * (t_2 \oplus t_3))\ u \\ & \quad = t[u/x] * (t_2 \oplus t_3) \end{aligned}$$

  And

$$\begin{aligned} & (t_1 * t_2) \oplus (t_1 * t_3)\ u \\ & \quad = ((\lambda x : A.t) * t_2 \oplus (\lambda x : A.t) * t_3)\ u \\ & \quad = (\lambda x : A.(t * t_2) \oplus \lambda x : A.(t * t_3))\ u \\ & \quad = (\lambda x : A.(t * t_2) \oplus (t * t_3))\ u \\ & \quad = (t[u/x] * t_2) \oplus (t[u/x] * t_3) \end{aligned}$$

  Based on induction on type B, we have $t[u/x] * (t_2 \oplus t_3) = (t[u/x] * t_2) \oplus (t[u/x] * t_3)$, Therefore we prove the case.
- Case:t_1 is of type $(T_1, T_2, .., T_n)$, t_2 and t_3 are of base type. Suppose t_1 is $(t'_1, t'_2, .., t'_n)$.
  Then

$$\begin{aligned} & t_1 * (t_2 \oplus t_3) \\ & \quad = (t'_1, t'_2, .., t'_n) * (t_2 \oplus t_3) \\ & \quad = (t'_1 * (t_2 \oplus t_3), t'_2 * (t_2 \oplus t_3), .., t'_n * (t_2 \oplus t_3)) \end{aligned}$$

$$\begin{aligned} & (t_1 * t_2) \oplus (t_1 * t_3) \\ & \quad = (t'_1, t'_2, .., t'_n) * t_2 \oplus (t'_1, t'_2, .., t'_n) * t_3 \\ & \quad = (t'_1 * t_2, t'_2 * t_2, .., t'_n * t_2) \oplus (t'_1 * t_3, t'_2 * t_3, .., t'_n * t_3) \\ & \quad = (t'_1 * t_2 \oplus t'_1 * t_3, t'_2 * t_2 \oplus t'_2 * t_3, .., t'_n * t_2 \oplus t'_n * t_3) \end{aligned}$$

  And based on induction we have $t'_j * (t_2 \oplus t_3) = (t'_j * t_2) \oplus (t'_j * t_3)$, so we have $t_1 * (t_2 \oplus t_3) = (t_1 * t_2) \oplus (t_1 * t_3)$.
- Case:t_1 is of type $T_1 + T_2$, t_2 and t_3 are of base type: this case is not possible because the righthand term is not well-typed.
- Case:t_1 is of type $(T_1, T_2, .., T_n)$, t_2 and t_3 are of type $(T'_1, T'_2, .., T'_n)$. Suppose t_1 : $(t'_{11}, t'_{12}, .., t'_{1n})$,$t_2$: $(t'_{21}, t'_{22}, .., t'_{2n})$ and $t_3 = (t'_{31}, t'_{32}, .., t'_{3n})$.
  Then

$$\begin{aligned} &t_1 * (t_2 \oplus t_3) \\ &\quad = (t'_{11}, t'_{12}, .., t'_{1n}) * ((t'_{21}, t'_{22}, .., t'_{2n}) \oplus (t'_{31}, t'_{32}, .., t'_{3n})) \\ &\quad = t'_{11} * (t'_{21} \oplus t'_{31}) \oplus t'_{12} * (t'_{22} \oplus t'_{32}) \oplus ... \oplus t'_{1n} * (t'_{2n} \oplus t'_{3n}) \end{aligned}$$

And we have

$$\begin{aligned} &(t_1 * t_2) \oplus (t_1 * t_3) \\ &\quad = (t'_{11}, t'_{12}, .., t'_{1n}) * (t'_{21}, t'_{22}, .., t'_{2n}) \oplus (t'_{11}, t'_{12}, .., t'_{1n}) * (t'_{31}, t'_{32}, .., t'_{3n}) \\ &\quad = ((t'_{11} * t'_{21}) \oplus (t'_{11} * t'_{31})) \oplus ((t'_{12} * t'_{22}) \oplus (t'_{12} * t'_{32})) \oplus ... \oplus ((t'_{1n} * t'_{2n}) \oplus (t'_{1n} * t'_{3n})) \end{aligned}$$

Based on induction we have $\forall j, t'_{1j} * (t'_{2j} \oplus t'_{3j}) = ((t'_{1j} * t'_{2j}) \oplus (t'_{1j} * t'_{3j}))$, and using Lemma 52 $t_1 = t'_1, t_2 = t'_2$, then $t_1 \oplus t_2 = t'_1 \oplus t'_2$ we prove the case.

◀

▶ **Lemma 55.** If $(t_1 \ominus t_2) \oplus (t_2 \ominus t_3)$ and $t_1 \ominus t_3$ are normalizable, then $(t_1 \ominus t_2) \oplus (t_2 \ominus t_3) = t_1 \ominus t_3$

Proof. If t_1, t_2 or t_3 is not closed, then we just substitute them to be closed. Because of the confluence and strong normalize property, we can assume that t_1, t_2 and t_3 are all normal forms.

Then we make induction on types of t_1.

- Case base type
  Then because on base type, we require that $(t_1 \ominus t_2) \oplus (t_2 \ominus t_3) = t_1 \ominus t_3$, Thus we prove the case.
- Case A→B
  Then we need to prove that $\forall u, ((t_1 \ominus t_2) \oplus (t_2 \ominus t_3))u = (t_1 \ominus t_3)u$.
  Then we can suppose that t_1, t_2 and t_3 are of the form $\lambda a : A.t$, if they are not, then we use $\lambda a : A.t_i\ a$ to substitute for t_i.
  Then we have

$$\begin{aligned} &((t_1 \ominus t_2) \oplus (t_2 \ominus t_3))u \\ &\quad = ((\lambda a : A.t'_1 \ominus \lambda a : A.t'_2) \oplus (\lambda a : A.t'_2 \ominus \lambda a : A.t'_3))u \\ &\quad = \lambda a : A.((t'_1 \ominus t'_2) \oplus (t'_2 \ominus t'_3))u \\ &\quad = ((t'_1[u/a] \ominus t'_2[u/a]) \oplus (t'_2[u/a] \ominus t'_3[u/a])) \\ &\quad = ((\lambda a : A.t'_1\ u \ominus \lambda a : A.t'_2\ u) \oplus (\lambda a : A.t'_2\ u \ominus \lambda a : A.t'_3\ u)) \\ &\quad = ((t_1\ u \ominus t_2\ u) \oplus (t_2\ u \ominus t_3\ u)) \end{aligned}$$

  And similarly we have $(t_1 \ominus t_3)u = (t_1\ u \ominus t_3\ u)$
  Base on induction on type B, we have $(t_1\ u \ominus t_3\ u) = ((t_1\ u \ominus t_2\ u) \oplus (t_2\ u \ominus t_3\ u))$
  Thus we prove the case.
- Case $(T_1, T_2, ..., T_n)$
  Let's suppose t_1 to be $(t_{11}, t_{12}, ..., t_{1n})$, t_2 to be $(t_{21}, t_{22}, ..., t_{2n})$ and t_3 to be $(t_{31}, t_{32}, ..., t_{3n})$.
  Then we have

$$\begin{aligned} &((t_1 \ominus t_2) \oplus (t_2 \ominus t_3)) \\ &\quad = (((t_{11} \ominus t_{21}) \oplus (t_{21} \ominus t_{31})), ..., ((t_{1n} \ominus t_{2n}) \oplus (t_{2n} \ominus t_{3n}))) \end{aligned}$$

  And

$$\begin{aligned} &(t_1 \ominus t_3) \\ &\quad = ((t_{11} \ominus t_{31}), ..., (t_{1n} \ominus t_{3n})) \end{aligned}$$

Base on induction on type T_i, we have $(t_{1i} \ominus t_{2i}) \oplus (t_{2i} \ominus t_{3i}) = (t_{1i} \ominus t_{3i})$ Thus we prove the case.
item Case $T_1 + T_2$: This case is not possible because it is not well-typed.

Thus we prove the theorem.

◀