基于 Coq 的第三代微积分机器证明系统

献给张景中、杨路教授 85 华诞

郭礼权, 付尧顺, 郁文生*

北京邮电大学天地互联与融合北京市重点实验室, 北京 100876

E-mail: coderglq@bupt.edu.cn, fuyys@bupt.edu.cn, wsyu@bupt.edu.cn

收稿日期: 2019-12-27; 接受日期: 2020-03-03; 网络出版日期: 2020-08-07; * 通信作者
国家自然科学基金 (批准号: 61936008 和 61571064) 资助项目

摘要 本文基于证明辅助工具 Coq, 完整实现林群院士和张景中院士等倡导的第三代微积分—没有极限的微积分—理论构架的形式化验证, 包括对张景中等发表的题为“微积分基础的新视角”论文中全部定义和定理的 Coq 描述. 进而, 对定理无例外地给出 Coq 的机器证明代码, 所有形式化过程已被 Coq 验证, 并在计算机上运行通过, 体现了基于 Coq 的数学定理机器证明具有可读性和交互性的特点, 其证明过程规范、严谨、可靠. 本文是实践研究人员利用计算机学习、理解、构建乃至教育现代数学理论的一个尝试.

关键词 证明辅助工具 Coq 第三代微积分 形式化 机器证明

MSC (2010) 主题分类 03B35, 44A45, 68Q45

1 引言

人工智能研究是当前科技发展的热点和前沿方向, 夯实人工智能基础理论尤为重要, 数学定理机器证明是人工智能基础理论的深刻体现 (参见文献 [1–8]).

数学定理的计算机形式化证明, 近年来随着计算机科学的迅猛发展, 特别是证明辅助工具 Coq、Isabelle 和 HOL (higher order logic) 等 (参见文献 [9–16]) 的出现, 取得了长足的进展 (参见文献 [1–5, 8]). Coq 是一个交互式定理证明与程序开发系统平台, 是一个用于描述定理内容和验证定理证明的计算机工具. 这些定理可能涉及纯数学、证明理论或程序验证等 (参见文献 [9, 10, 13, 15]) 多个方面. 在推理和编程方面, Coq 都拥有足够强大的表达能力, 从构造简单的项、执行简单的证明, 直至建立完整的理论、学习复杂的算法等, 对学习者的能力有着不同层次的要求 (参见文献 [9, 10, 13, 15]).

Coq 是一个交互式的编译环境, 用户以人机对话的方式一问一答, 用户可以边设计、边修改, 使证明中的错误及时得到补证. Coq 系统的基本原理是归纳构造演算, 是一个形式化系统. Coq 支持自动

英文引用格式: Guo L Q, Fu Y S, Yu W S. A mechanized proof system of the third generation calculus in Coq (in Chinese). Sci Sin Math, 2021, 51: 115–136, doi: 10.1360/SSM-2019-0327

推理程序. Coq 通过命令式程序进行逻辑推导, 可以利用已证命题进行自动推理. Coq 中的归纳类型扩展了传统程序设计语言中有关类型定义的概念, 类似于大多数函数式程序设计语言中的递归类型定义 (参见文献 [9, 10, 13, 15]). Coq 有一支强大的全职研发队伍, 支持开源.

目前, Coq 已成为数学定理证明与计算逻辑领域的一个重要工具. 2005 年, 国际著名计算机专家 Gonthier 和 Werner 成功基于 Coq 给出了著名的“四色定理”的计算机证明 (参见文献 [2]), 进而, Gonthier 等又经过努力, 于 2012 年完成对“有限单群分类定理”的机器验证 (该证明过程约有 4,300 个定义和 15,000 条定理, 约 170,000 行 Coq 代码) (参见文献 [1, 3, 4]), 2015 年, Hales 等^[5]又在 HOL Light 和 Isabelle 上完成了对“Kepler 猜想”的机器验证, 使得各种形式化证明辅助工具在学术界得到广泛认可.

关于形式化数学, 在 20 世纪初对于数学基础的深入讨论中受到重视, 对整个 20 世纪数学的发展产生了深远的影响, 虽然有时也饱受过于“形式”的诟病 (参见文献 [17–20]). 20 世纪 90 年代、特别是进入 21 世纪以来, 随着计算机形式化工具的出现, 尤其是上述一系列著名数学难题形式化证明的实现, 形式化证明辅助工具在学术界的影响日益增强. Wiedijk^[16]指出, “数学历史上发生过三次革命. 第一次是公元前 3 世纪, 古希腊数学家 Euclid《几何原本》引入数学证明方法; 第二次是 19 世纪 Cauchy 等引入‘严格’数学方法, 以及后来的数理逻辑和集合论; 第三次就是当前正在进行的形式化数学” (参见文献 [16, 21]), 全球各相关研究团队已经或计划完成包括 Gödel 第一不完备性定理、Jordan 曲线定理、素数定理和 Fermat 大定理等在内的一百个著名数学定理的计算机形式化证明 (参见文献 [16]). 2002 年菲尔兹奖获得者 Voevodsky^[22]和 2010 年菲尔兹奖获得者 Villani^[23]都大力倡导发展可信数学 (reliable mathematics).

对于数学理论的形式化来说, 实现微积分的形式化更为基础. 微积分是数学史上最伟大的成就之一, 开启了近代科技的新纪元, 对物理学和天文学等其他学科的发展也起到了重要的促进作用. 微积分的创立距今已有三百年的历史 (参见文献 [17, 18, 24]), 由 Newton 和 Leibniz 创建的微积分, 是第一代微积分, 因为其对无穷小概念的定义模糊不清, 被称为说不清楚的微积分. Cauchy 和 Weierstrass 等以严格的极限理论为基础的微积分是第二代微积分, 由于其概念和推理烦琐迂回, 被称为听不明白的微积分. 为了使微积分严谨且直观易懂, 让学习者用较少的时间和精力就能明白其原理, 国内外学者长期致力于新一代微积分的研究 (参见文献 [25–27]).

早在 18 世纪后期, Lagrange 就试图不用 Newton 的“流数”和 Leibniz 的“无穷小”来建立微积分学, 但其中无穷级数的收敛问题, 仍无法避开极限的概念 (参见文献 [24]). 20 世纪 60 年代, Lusternik 和 Sobolev^[28]提出了一致可导的概念, 可以证明一致可导等价于连续可导, 直接使用一致可导概念可以简化微积分推理过程, 但其仍依赖于极限概念. 1999 年, Dovermann^[29]基于 Lipschitz 条件引入不用极限的可微性概念, 使其学生可以快速地入门微积分理论, 同时他也承认其所引入的可微性概念相对传统微积分理论是不严谨的. 2005 年, Sparks^[30]在其著作 *Calculus without Limits—Almost* 中避开极限概念, 用浅显易懂的例子对微积分的基本概念和公式作了直观的阐述, 但并未系统地建立不用极限的微积分完整理论. 近年来, Livshits^[31, 32]也提出了一种不依赖实数、极限和连续等基本概念而直接定义微分和积分的方法, 该方法也是建立在 Lipschitz 函数可微性概念的基础上的.

在我国, 林群和张景中也大力倡导第三代微积分, 即没有极限的微积分 (参见文献 [25–27, 33–35]). 2002 年, 林群指出采用一致微商的定义可以大大简化微积分基本定理的论证 (参见文献 [33]), 之后他将一致微商的思想与不等式刻画极限过程的思路结合起来, 提出了微积分的初等化方案. 2010 年, 张景中提出了甲函数和乙函数的概念, 在不使用极限或无穷小的前提下建立了初等化的微积分, 大大简化了微积分理论, 做到了“微分不微, 积分不积, 推理简化, 模型统一” (参见文献 [25, 27]). 特别地, 张

景中和冯勇^[26]发表于《中国科学 A 辑》上题为“微积分基础的新视角”的论文中, 提出了一个函数的差商是另一个函数中值的概念, 刻画了原函数和导函数的本质, 给出直观的积分系统定义, 得到强可导和一致可导的充分必要条件, 从而简单完整地建立了不用极限的微积分系统. 2019 年, 林群和张景中^[35]避开极限概念, 运用“差商控制函数”简捷地解决了判断函数的增减凹凸、计算曲边梯形的面积和做曲边的切线等几大类问题. 而一旦对初等函数类找出计算“差商控制函数”的一般公式 (参见文献 [25]), 即可简捷而严谨地建立微积分系统.

第三代微积分与前两代微积分, 在具体计算方法上没有不同, 不同的只是对原理的说明 (参见文献 [25–27]). 本文作者就此问题曾与中国科学院重庆绿色智能技术研究院张矩研究员有过深入的探讨. 第三代微积分的基本思想是微积分的初等化, 事实上, 由于在第三代微积分基本概念中使用了存在量词, 在逻辑上其原理与第二代微积分是完全等价的, 但其优点是明显的, 它将传统微积分极限概念中同时使用全称量词和存在量词、且存在量词依赖于全称量词等烦琐迂回的难点转移为仅使用存在量词的不等式描述, 得到的不等式形式也较为自然, 并且“将微分、积分系统统一成一个系统. 而不是像以前那样, 微分系统与积分系统被人为地割裂开来, 学生通常是先学习微分系统, 然后再学习积分系统” (参见文献 [26]), 这一点可从文献 [26] 在给出几个基本概念之后, 迅速得到微积分的核心定理“Newton-Leibniz 公式”, 进而建立级数理论的描述框架中体会到. 另外, 将微积分学初等化并代数化之后, 还可方便地利用新近不等式定理机器证明研究取得的成果 (参见文献 [7, 36, 37]), 为微积分系统机械化研究提供必要的准备. 这也是林群和张景中大力倡导第三代微积分的一个初衷.

应当指出, 在数学定理机器证明方面, 由我国吴文俊开创的“吴方法”的研究在国际上是独具特色和领先的 (参见文献 [6, 38, 39]). 吴先生开创的定理机器证明“吴方法”主要基于多项式代数方法, 依赖符号计算和数值计算, 有完备的算法, 但算法复杂性高, 在变量或参数过多的情形下, 受硬件物理条件限制 (参见文献 [6, 7, 38–40]). 形式化证明主要基于逻辑的推理, 人机交互, 可将人的智慧和机器的智能结合起来, 但用形式化方法直接处理等式或不等式的代数运算代价较大, 将计算机辅助证明工具和基于符号数值混合计算的多项式代数方法结合, 是今后非常有吸引力的研究课题.

本文基于证明辅助工具 Coq (参见文献 [9, 10, 13, 15]), 完整实现林群和张景中等倡导的第三代微积分—没有极限的微积分—理论构架的形式化验证, 包括对张景中等发表于《中国科学 A 辑: 数学》上题为“微积分基础的新视角”的论文 [26] 中全部定义和定理的 Coq 描述. 进而, 对定理无例外地给出 Coq 的机器证明代码, 所有形式化过程已被 Coq 验证, 并在计算机上运行通过, 体现了基于 Coq 的数学定理机器证明具有可读性和交互性的特点 (参见文献 [9, 10, 13, 15]), 其证明过程规范、严谨、可靠. 本文是实践研究人员利用计算机学习、理解、构建乃至教育现代数学理论的一个尝试.

本文结构安排如下: 第 1 节给出背景知识的介绍; 第 2 节预备知识包括基本定义及其 Coq 描述; 第 3–5 节完整实现张景中和冯勇在文献 [26] 中第三代微积分理论的形式化构建, 我们严格按照文献 [26] 的结构顺序, 在相关定义和定理的人工描述之后, 相应给出其精确的 Coq 描述代码, 而所有定理的证明过程均通过 Coq 代码完成; 第 6 节总结我们的结果并给出相关注记.

2 预备知识

在建立第三代微积分机器证明系统之前, 我们首先给出实数理论的相关定义, 实现其规范而严谨的形式化描述, 并加入可读的数学符号.

Coq 是一种依赖类型的函数式编程语言, 提供了一种规范说明语言 Gallina, 该语言基于归纳构造

演算——一种带类型 λ 演算的变体实现, 具有十分强大的表达能力, 可准确描述项、类型、证明和程序 (参见文献 [9,10]). 在 Coq 中, 基本类型包括如下两种形式.

(1) 原子类型, 由单一的标识符组成, 如自然数类型 `nat`、整数类型 `Z`、布尔类型 `bool` 和命题类型 `Prop` 等.

(2) 形如 $(A \rightarrow B)$ 的类型, 其中 A 和 B 本身也是类型. 这种类型也被称为箭头类型, 表示函数类型: 将一个类型为 A 的对象作为参数, 返回一个类型 B 的对象.

形式化系统中涉及的自然数和实数等原子类型的定义, 在 Coq 的基本库中均已给出严谨的形式化描述. Coq 使用关键字 `Inductive` 将自然数归纳构造为数据类型 `nat`, 包括两个构造子 `O` 和 `S`. 构造子 `O` 类型为 `nat`, 表示自然数 0; 构造子 `S` 类型为 `nat  $\rightarrow$  nat`, 表示一个自然数的后继, 若 n 类型为 `nat`, 则 `S n` 类型也为 `nat`, 如 `S (S 0)` 表示自然数 2. Coq 中自然数类型 `nat` 定义如下:

```
Inductive nat : Set :=
| O : nat
| S : nat -> nat.
```

在 Coq 标准库中, 实数通过公理系统建立, 包括域公理、序公理、Archimedes 公理和实数完备性公理 (本文中涉及的微积分理论构架未直接用到实数完备性公理). 为简化输入表达式, Coq 系统提出解释辖域, 默认 `nat_scope`, 使用库中实数相关定义、定理前, 需引入实数库, 开启辖域 `R_scope`. 对应代码如下:

```
Require Import Reals.
Open Scope R_scope.
```

另外, 一些初等逻辑的实用知识是必要的, 事实上我们只用到古典二值逻辑: 任何命题只允许“真”或“假”的情形而没有其他情形, 并默认“排中律”成立. 使用排中律可证明常见的逻辑命题. 排中律可以以公理的形式给出, 其 Coq 代码如下:

```
Axiom classic : forall P : Prop, P  $\vee$   $\neg$ P.
```

本系统中, 通用的初等逻辑运算符号“ $\sim$ ”表示否定, 亦即关系词“非”;“ $\vee$ ”表示逻辑析取, 亦即关系词“或”;“ $\wedge$ ”表示逻辑合取, 亦即关系词“与”. 符号“ $\Rightarrow$ ”表示蕴涵. 符号“ $\Leftrightarrow$ ”表示“等价”.

微积分理论讨论区间上函数的性质, 对区间形式化描述, 需给出集合类型的定义:

```
Definition R_Ensemble := R -> Prop.
```

上述定义中, R 代表实数类型, `Prop` 为集合中的元素满足的性质, 当一个对象被定义为 `R_Ensemble` 类型时, 即该对象为一个集合.

定义“属于”, 符号为“ $\in$ ”, 表示元素与集合之间的关系.

```
Definition In (x:R)(A:R_Ensemble) :Prop := A x.
Notation "x  $\in$  A" := (In x A) (at level 10).
```

`In x A` 表示实数 x 是集合 A 中的一个元素, 即 $x \in A$. Coq 中的 `Notation` 命令可使代码中相关定义与通常的数学表达一致.

下面给出各种区间类型 (开区间、闭区间、半开半闭区间) 的形式化描述:

```
Definition oo (a b:R) := fun x:R => a < b /\ a < x < b.
Notation "( a , b )" := (oo a b).
Definition cc (a b:R) := fun x:R => a < b /\ a <= x <= b.
Notation "[ a , b ]" := (cc a b).
Definition oc (a b:R) := fun x:R => a < b /\ a < x <= b.
```

```

Notation "( a , b ]" := (oc a b).
Definition co (a b:R) := fun x:R => a<b/\a<=x<b.
Notation "[ a , b )" := (co a b).

```

对于函数的定义我们需要特别强调, 与集合论中把函数看成 Descartes 积 $A \times B$ 的特定子集的观点不同, 在 Coq 中一般以计算机科学的角度来考虑函数的概念, 即函数是把类型为 A 的值映射到类型为 B 的值的**有效计算过程 (或称为算法)**^[8,9]. 为简便起见, 仅用类型 $A \rightarrow B$ 表示所有将类型为 A 的变元映射到一个类型为 B 的结果值的函数.

微积分是建立在实数基础上的学科, 本系统定义和定理中涉及的一元函数均声明为 $R \rightarrow R$ 的类型. 假如声明一个类型为 $R \rightarrow R$ 的对象 f , 则 f 表示一个将类型为实数的变元映射为实数值的函数; 当其作用到一个实数 x 时, Coq 代码中 “ $f\ x$ ” 表示函数在变量为 x 时对应的函数值. 对于多元函数, 在 Coq 中可以用一元函数很方便地表示出来, 以二元函数为例, 二元函数即将类型为 A 的变元映射到一个类型为 $B \rightarrow C$ 的值. 本系统中将二元函数的类型声明为 $R \rightarrow R \rightarrow R$.

下面对函数在区间上的一些特殊性质进行形式化描述.

函数 $f(x)$ 在区间 q 上非严格单调递增:

```

Definition mon_increasing f (q:R_Ensemble) :=
  forall x y:R, x ∈ q /\ y ∈ q /\ y-x>0 -> f x <= f y.

```

函数 $f(x)$ 在区间 q 上严格单调递增:

```

Definition strict_mon_increasing f (q:R_Ensemble) :=
  forall x y:R, x ∈ q /\ y ∈ q /\ y-x>0 -> f x < f y.

```

函数 $f(x)$ 在区间 q 上非严格单调递减:

```

Definition mon_decreasing f (q:R_Ensemble) :=
  forall x y:R, x ∈ q /\ y ∈ q /\ y-x>0 -> f x >= f y.

```

函数 $f(x)$ 在区间 q 上严格单调递减:

```

Definition strict_mon_decreasing f (q:R_Ensemble) :=
  forall x y:R, x ∈ q /\ y ∈ q /\ y-x>0 -> f x > f y.

```

函数 $f(x)$ 在区间 q 上倒数无界:

```

Definition bounded_rec_f f (q:R_Ensemble) :=
  forall M:R, M>0 -> exists z:R, z ∈ q /\ M < Rabs(1/(f z)).

```

函数 $f(x)$ 在区间 q 上正值单调不减:

```

Definition pos_inc f (q:R_Ensemble) :=
  (forall z:R, z ∈ q -> f z > 0) /\
  (forall z1 z2:R, z1 ∈ q -> z2 ∈ q -> z1<z2 -> f z1 <= f z2).

```

函数 $f(x)$ 在区间 q 上正值单调不减:

```

Definition pos_dec f (q:R_Ensemble) :=
  (forall z:R, z ∈ q -> f z > 0) /\
  (forall z1 z2:R, z1 ∈ q -> z2 ∈ q -> z1<z2 -> f z1 >= f z2).

```

本文定理证明中出现的一些基本 Coq 指令是标准的 (参见文献 [9,10,13,15]), 这些指令在使用过程中是容易理解的, 各指令的详细功能也可参见文献 [9,10,13,15]. 指令清单见表 1.

上述预备知识及基本概念的 Coq 描述对构建第三代微积分理论已足够.

表 1 定理证明中涉及的 Coq 常用指令简表

Coq 指令	用法
intro/intros	引入目标中的条件
unfold	将定义展开
rewrite H	将 H 的右边当作左边代入目标
rewrite < - H	将 H 的左边当作右边代入目标
elim	消去, 对条件进行分解
apply	应用指定条件到证明目标
destruct	拆掉指定条件中的或和与
split	拆分目标中间的与
left	证明“或”左边的目标
right	证明“或”右边的目标
generalize	引入假设条件
assert	指定一个新的假设条件并证明
auto	自动重复使用 intros、apply、rewrite 等简单策略

3 导数和积分的初等定义

自本节起, 后续内容严格按照文献 [26] 的结构顺序, 在相关定义和定理的人工描述之后, 相应给出其精确的 Coq 描述代码, 并实现所有定理的完整 Coq 形式化证明.

本节给出一致连续、一致 (强) 可导、积分系统和其相关定义, 在此基础上验证一致 (强) 可导的基本性质和积分估值定理.

定义 3.1 (函数的一致连续) 设 $f(x)$ 是定义在区间 $[a, b]$ 上的函数, 若对于任意的 $x \in [a, b]$ 和任意的 h , 均有下列不等式成立:

$$|f(x+h) - f(x)| \leq d(h), \quad (3.1)$$

其中, $d(h)$ 为 $(0, b-a]$ 上与点 x 无关的正值单调不减函数, 并且倒数无界, 则称 $f(x)$ 在 $[a, b]$ 上一致连续.

```
Definition uniform_continuous f (a b:R) :=
  exists d, pos_inc d (0,b-a] /\ bounded_rec_f d (0,b-a] /\
  (forall x h:R, x ∈ [a,b] /\ (x+h) ∈ [a,b] /\ h<>0 ->
    (Rabs (f(x+h) - f(x))) <= d(Rabs h)).
```

定义 3.2 (一致可导) 设函数 $F(x)$ 在区间 $[a, b]$ 上有定义, 如果存在一个在 $[a, b]$ 上有定义的函数 $f(x)$ 和正数 M , 使得对 $[a, b]$ 上任意的 x 和 $x+h$, 有下列不等式成立:

$$|F(x+h) - F(x) - f(x)h| \leq M|h|d(|h|), \quad (3.2)$$

其中, $d(h)$ 为 $(0, b-a]$ 上与点 x 无关的正值单调不减函数, 并且倒数无界, 则称 $F(x)$ 在 $[a, b]$ 上一致可导, 并且称 $f(x)$ 为 $F(x)$ 的导数, 记作 $F'(x) = f(x)$.

```
Definition uniform_derivability F (a b:R) :=
  exists d, pos_inc d (0,b-a] /\ bounded_rec_f d (0,b-a] /\
  exists f M, 0<M /\ forall x h:R, x ∈ [a,b] /\ (x+h) ∈ [a,b] /\ h<>0 ->
  Rabs (F(x+h) - F(x) - f(x)*h) <= M*(Rabs h)*d(Rabs h).
```

```

Definition derivative F f a b :=
  exists d, pos_inc d (0,b-a] /\
  bounded_rec_f d (0,b-a] /\
  exists M:R, 0<M /\ forall x h:R, x ∈ [a,b] /\ (x+h) ∈ [a,b] /\ h>0 ->
  Rabs (F(x+h) - F(x) - f(x)*h) <= M*(Rabs h)*d(Rabs h).

```

在定义 3.1 中取 $d(x) = x$, 就是强可导的定义:

```

Definition str_derivability F (a b:R) :=
  exists f M, 0<M /\ forall x h:R, x ∈ [a,b] /\ (x+h) ∈ [a,b] ->
  Rabs (F(x+h) - F(x) - f(x)*h) <= M*(h^2).

```

```

Definition str_derivative F f (a b:R) :=
  exists M:R, 0<M /\ forall x h:R, x ∈ [a,b] /\ (x+h) ∈ [a,b] ->
  Rabs (F(x+h) - F(x) - f(x)*h) <= M*(h^2).

```

易见, 强可导蕴涵一致可导, 且导数相同. 容易证明如下定理:

定理 3.1 设 $F(x)$ 和 $G(x)$ 一致 (强) 可导, 并且导数分别为 $f(x)$ 和 $g(x)$, 则

- (1) 对任意常数 c , $cF(x)$ 一致 (强) 可导, 且其导数是 $cf(x)$;
- (2) $F(x) + G(x)$ 一致 (强) 可导, 且其导数为 $f(x) + g(x)$;
- (3) $F(cx + d)$ 一致 (强) 可导, 且其导数为 $cf(cx + d)$.

```

Theorem Theorem3_1_1 : forall (a b c:R) F f,
  derivative F f a b -> derivative (mult_real_f c F) (mult_real_f c f) a b.

```

```

Theorem Theorem3_1_1' : forall (a b c:R) F f,
  str_derivative F f a b ->
  str_derivative (mult_real_f c F) (mult_real_f c f) a b.

```

```

Theorem Theorem3_1_2 : forall F G f g (a b:R),
  derivative F f a b -> derivative G g a b ->
  derivative (plus_Fu F G) (plus_Fu f g) a b.

```

```

Theorem Theorem3_1_2' : forall F G f g (a b:R),
  str_derivative F f a b -> str_derivative G g a b ->
  str_derivative (plus_Fu F G) (plus_Fu f g) a b.

```

```

Theorem Theorem3_1_3a : forall F f (a b c d:R),
  c > 0 -> derivative F f a b ->
  derivative(Com_F F c d)(mult_real_f c(Com_F f c d))((a-d)/c)((b-d)/c).

```

```

Theorem Theorem3_1_3b : forall F f (a b c d:R),
  c < 0 -> derivative F f a b ->
  derivative(Com_F F c d)(mult_real_f c(Com_F f c d))((b-d)/c)((a-d)/c).

```

```

Theorem Theorem3_1_3a' : forall F f (a b c d:R),
  c > 0 -> str_derivative F f a b ->
  str_derivative(Com_F F c d)(mult_real_f c(Com_F f c d))((a-d)/c)((b-d)/c).

```

```

Theorem Theorem3_1_3b' : forall F f (a b c d:R),
  c < 0 -> str_derivative F f a b ->
  str_derivative(Com_F F c d)(mult_real_f c(Com_F f c d))((b-d)/c)((a-d)/c).

```

定理 3.1 中几个子定理的证明过程基本一致, 为节省空间, 仅用图 1 展示定理 3.1(1) 的证明代码.

```

Theorem Theorem3_1_1 : forall (a b c:R) F f,
  derivative F f a b -> derivative (mult_real_f c F) (mult_real_f c f) a b.
Proof.
  intros a b c F f .
  unfold derivative; intro.
  destruct H, H, H0, H1, H1.
  exists x; split; auto.
  generalize (total_eq_or_neq c 0); intro. destruct H3.
  - exists x0. split; auto; intros.
    apply H2 in H4. unfold mult_real_f; rewrite H3.
    repeat rewrite Rmult_0_l. unfold Rminus; rewrite Ropp_0;
    repeat rewrite Rplus_0_l; rewrite Rabs_R0.
    apply Rle_trans with (r2:=Rabs(F(x1+h)-F x1-f x1*h)); auto.
    apply Rabs_pos.
  - exists (Rabs c * x0). split.
    apply Rmult_lt_0_compat; auto.
    apply Rabs_pos_lt; auto. intros. apply H2 in H4.
    unfold mult_real_f. rewrite Rmult_assoc.
    rewrite <- Rmult_minus_distr_1_3 with
      (r:=c)(r1:=(F(x1+h)))(r2:=F x1)(r3:=f x1*h).
    rewrite Rabs_mult; repeat rewrite Rmult_assoc; apply Rmult_le_compat_l.
    apply Rabs_pos. repeat rewrite <- Rmult_assoc; apply H4.
Qed.

```

图 1 定理 3.1(1) 证明代码

定义 3.3 (积分系统) 设 $f(x)$ 在区间 $[a, b]$ 上有定义, 如果有一个二元函数 $S(u, v)$ ($u \in [a, b]$, $v \in [a, b]$), 满足

- (I) 可加性: 对 $[a, b]$ 上任意的 w_1, w_2 和 w_3 , 有 $S(w_1, w_2) + S(w_2, w_3) = S(w_1, w_3)$;
- (II) 非负性: 在 $[a, b]$ 上的任意子区间 $[w_1, w_2]$ 上, 如果 $m \leq f(x) \leq M$, 就必然有

$$m(w_2 - w_1) \leq S(w_1, w_2) \leq M(w_2 - w_1),$$

则称 $S(u, v)$ 是 $f(x)$ 在 $[a, b]$ 上的一个积分系统. 如果 $f(x)$ 在 $[a, b]$ 上有唯一的积分系统 $S(u, v)$, 则称 $f(x)$ 在 $[a, b]$ 上可积, 并称数值 $S(w_1, w_2)$ 为 $f(x)$ 在 $[w_1, w_2]$ 上的定积分.

上述导数和积分的定义都没有用到极限概念, 比传统教材上的导数和 Riemann 积分的定义简明得多, 以此为基础建立起来的微积分学也很容易被学生接受和掌握 (参见文献 [26]). 注意, 在上述定义中, 积分系统要满足的非负性是一个非严格的不等式, 但按照通常定积分的几何意义就是曲线与坐标轴所围成的面积, 当然面积满足严格不等式 (参见文献 [26]). 所谓的严格不等式就是:

定义 3.4 (积分严格不等式) 设 $S(u, v)$ 是 $f(x)$ 在区间 $[a, b]$ 上的一个积分系统, 在任意的子区间 $[w_1, w_2] \subseteq [a, b]$ 上, 如果 $m < f(x) < M$, 就必有

$$m(w_2 - w_1) < S(w_1, w_2) < M(w_2 - w_1),$$

则称积分系统 $S(u, v)$ 在区间 $[a, b]$ 上满足积分严格不等式.

```

Definition strict_unequal (S:R->R->R)(f:R->R)(a b:R) :=
  integ_sys S f a b ->
  forall w1 w2:R, w1 ∈ [a,b] /\ w2 ∈ [a,b] /\ w2 - w1 > 0 ->
  (forall m:R, (forall x:R, x ∈ [w1,w2] -> m < f x) ->
  m*(w2 - w1) < S w1 w2) /\
  (forall M:R, (forall x:R, x ∈ [w1,w2] -> M > f x) ->
  S w1 w2 < M*(w2 - w1)).

```


对于定义 3.3 进一步研究可知, 若 $S(u, v)$ 是 $f(x)$ 在 $[a, b]$ 上的一个积分系统, 令 $G(x) := S(a, x)$, 由于它满足可加性, 所以, $S(u, v)$ 可以写成如下的等价形式:

$$S(u, v) = S(a, v) - S(a, u) = G(v) - G(u).$$

下面证明, 如果积分系统满足积分严格不等式, 则可以得到估值定理.

定理 3.2 (估值定理) 假设 $S(x, y) = G(y) - G(x)$ 是 $f(x)$ 在区间 $[a, b]$ 上定义的积分系统, 并且满足积分严格不等式, 则对于任意的子区间 $[u, v] \subseteq [a, b]$, 都存在 $x_1, x_2 \in [u, v]$ 使得下式成立:

$$f(x_1)(v - u) \leq G(u) - G(v) \leq f(x_2)(v - u). \quad (3.3)$$

```
Theorem Valuation_Theorem : forall (S:R->R->R) (f G:R->R) (a b:R),
  integ_sys S f a b ->
  forall y, y ∈ [a,b] -> (forall x:R, x ∈ [a,b] -> G x = S y x) ->
  Strict_unequal S f a b ->
  forall u v, u ∈ [a,b] /\ v ∈ [a,b] /\ v - u > 0 ->
  exists x1 x2, x1 ∈ [u,v] /\ x2 ∈ [u,v] /\
  (f x1)*(v-u) <= G(v) - G(u) <= (f x2)*(v-u).
```

本定理证明的完整代码见附录.

4 积分与微分

本节引入一个函数的差商是另一个函数中值的概念, 在此基础上探讨积分系统、强可导和一致可导的性质.

定义 4.1 (差商与中值) 设 $F(x)$ 和 $f(x)$ 为在 $[a, b]$ 上定义的两个函数, 对任意区间 $[u, v] \subseteq [a, b]$ 均存在 $p \in [u, v]$ 和 $q \in [u, v]$ 满足不等式

$$f(p) \leq \frac{F(v) - F(u)}{(v - u)} \leq f(q), \quad (4.1)$$

则称 $F(x)$ 的差商是 $f(x)$ 的中值.

```
Definition diff_quo_median F f a b :=
  forall u v, u ∈ [a,b] /\ v ∈ [a,b] /\ v-u>0 ->
  exists (p q:R), p ∈ [u,v] /\ q ∈ [u,v] /\ f p <= (F v-F u)/(v-u) <= f q.
```

若 $f(x)$ 的差商是 $g(x)$ 的中值, 求导数和求积分的问题实际上就是, 知道了一个函数, 求另一个函数的问题.

下面用一个函数的差商是另一个函数的中值来刻画积分和微分.

定理 4.1 设 $S(u, v)$ 为 $f(x)$ 在 $[a, b]$ 上的积分系统, 并且满足积分严格不等式. 在 $[a, b]$ 上取一点 c , 令 $F(x) = S(c, x)$, 则在 $[a, b]$ 上 $F(x)$ 的差商是 $f(x)$ 的中值; 反之, 若在 $[a, b]$ 上 $F(x)$ 的差商是 $f(x)$ 的中值, 令 $S(u, v) = F(v) - F(u)$, 则 $S(u, v)$ 是 $f(x)$ 上的一个积分系统, 并且满足积分严格不等式.

```
Theorem Theorem4_1_1 : forall (S: R->R->R) (f F:R->R) (a b:R),
  integ_sys S f a b ->
```

```

strict_unequal S f a b ->
forall y, y ∈ [a,b] ->(forall x, x ∈ [a,b] -> F x = S y x) ->
diff_quo_median F f a b.
Proof. ... Qed.

Theorem Theorem4_1_2 : forall (S: R->R->R) (f F:R->R) (a b:R),
  (forall u v, u ∈ [a,b] /\ v ∈ [a,b] -> S u v = F v - F u) ->
  diff_quo_median F f a b ->
  integ_sys S f a b /\ strict_unequal S f a b.
Proof. ... Qed.

```

本节定理是下节微积分基本定理的预备工作, 本节及下节的个别定理证明过程较复杂, 代码量较大, 不可能在文中和附录中全部给出, 仅给出全部定义及定理形式化代码, 证明细节省略, 完整代码可见 <https://github.com/LittleGavin/calculus.3rd>.

在 Coq 形式化构建第三代微积分理论的构建过程中, 若省略了证明代码, 也可使用 Coq 的 “Admitted” 命令承认相关命题, 不影响后续调用. 本文中所有使用 “Admitted” 命令处的证明代码都已完成, 均可在上述网址中下载.

在上节定义中, 积分系统不一定满足积分严格不等式. 但常用的积分系统都满足积分严格不等式. 如果只对满足积分严格不等式的积分系统感兴趣, 由上面的定理知, 可用下面更直观的定义来代替以前所给出的积分系统的定义.

定义 4.2 (积分系统和定积分) 设 $f(x)$ 在区间 $[a, b]$ 上有定义, 如果存在一个二元函数 $S(x, y)$, $x \in [a, b]$, $y \in [a, b]$, 满足

- (I) 可加性: 对 $[a, b]$ 上任意的 u, v 和 w , 有 $S(u, v) + S(v, w) = S(u, w)$;
- (II) 中值性: 对 $[a, b]$ 上任意的 $u < v$, 在 $[u, v]$ 上必有两点 p 和 q , 使得

$$f(p)(v - u) \leq S(u, v) \leq f(q)(v - u),$$

则称 $S(x, y)$ 是 $f(x)$ 在 $[a, b]$ 上的一个积分系统. 如果 $f(x)$ 在 $[a, b]$ 上有唯一的积分系统 $S(x, y)$, 则称 $f(x)$ 在 $[a, b]$ 上可积, 并称数值 $S(u, v)$ 为 $f(x)$ 在 $[u, v]$ 上的定积分.

```

Definition additivity' (S:R->R->R)(a b:R):=
  forall(u v w:R), u ∈ [a,b] /\ v ∈ [a,b] /\ w ∈ [a,b] ->
  S u v + S v w = S u w.

Definition median (S:R->R->R)(f:R->R)(a b:R):=
  forall(u v:R), u ∈ [a,b] /\ v ∈ [a,b] /\ v - u > 0 ->
  exists p q:R, p ∈ [u,v] /\ q ∈ [u,v] /\
  f p *(v - u) <= S u v <= f q *(v - u).

Definition integ_sys' (S:R->R->R)(f:R->R)(a b:R) :=
  additivity' S a b /\ median S f a b.
Notation " S ∫ f '(x)dx' " := (integ_sys' S f)(at level 10).

Definition integrable' f a b :=
  exists S, integ_sys' S f a b /\
  (forall S':R->R->R, integ_sys' S' f a b -> S' = S).

```

以下探讨 $f(x)$ 在什么条件下, 存在唯一积分系统, 该积分系统就是它的定积分.

定理 4.2 设在 $[a, b]$ 上, $F(x)$ 的差商是 $f(x)$ 的中值, 且 $f(x)$ 一致连续, 则 $f(x)$ 在 $[a, b]$ 上具有唯一的积分系统 $S(x, y) = F(y) - F(x)$, 其中 $x, y \in [a, b]$.

```
Theorem Theorem4_2 : forall F f a b,
  diff_quo_median F f a b -> uniform_continuous f a b ->
  integ_sys' (fun u v => F v - F u) f a b /\
  (forall G, diff_quo_median G f a b ->
    forall u v, u ∈ [a,b] /\ v ∈ [a,b] /\ v-u>0 ->
      G v - G u = F v - F u).
```

Proof. Admitted.

下面考察强可导函数和一致可导函数的性质, 有如下定理:

定理 4.3 在 $[a, b]$ 上, $F(x)$ 的差商是 $f(x)$ 的中值, 且 $f(x)$ 的差商有界 (即存在正数 $M > 0$, 使得 $|f(x+h) - f(x)| \leq M|h|$), 当且仅当 $F(x)$ 在 $[a, b]$ 上强可导 (即满足不等式 $|F(x+h) - F(x) - f(x)h| \leq Mh^2$).

```
Theorem Theorem4_3 : forall F f a b,
  diff_quo_median F f a b /\
  (exists M, M>0 /\ forall x h, x ∈ [a,b] /\ (x+h) ∈ [a,b] ->
    Rabs(f(x+h) - f(x)) <= M*(Rabs h))<->
  str_derivative F f a b.
```

Proof. Admitted.

关于一致可导的问题, 有如下定理:

定理 4.4 函数 $F(x)$ 在 $[a, b]$ 上一致可导, 其导函数为 $f(x)$, 当且仅当 $F(x)$ 的差商是 $f(x)$ 的中值, 并且 $f(x)$ 在 $[a, b]$ 上一致连续.

```
Theorem Theorem4_4 : forall F f a b, derivative F f a b <->
  diff_quo_median F f a b /\ uniform_continuous f a b.
```

Proof. Admitted.

这里采用了 Coq 中的 Admitted 命令, 而省略了定理 4.2–4.4 的详细证明代码.

5 微积分系统的基本定理

本节将在新定义的基础上证明微积分系统的基本定理, 从而建立起微积分体系.

定理 5.1 设 $F(x)$ 在 $[a, b]$ 上一致可导, 其导数为 $f(x)$. 若在 $[a, b]$ 上, 恒有 $f(x) \geq 0$, 则 $F(x)$ 单调增; 若在 $[a, b]$ 上, 恒有 $f(x) \leq 0$, 则 $F(x)$ 单调减. 当不等式中等号不成立时, $F(x)$ 是严格单调增或严格单调减.

```
Theorem Theorem5_1 : forall F f a b, derivative F f a b ->
  ((forall x, x ∈ [a,b] -> f(x) >= 0) -> mon_increasing F (cc a b)) /\
  ((forall x, x ∈ [a,b] -> f(x) <= 0) -> mon_decreasing F (cc a b)) /\
  ((forall x, x ∈ [a,b] -> f(x)>0) -> strict_mon_increasing F (cc a b)) /\
  ((forall x, x ∈ [a,b] -> f(x)<0) -> strict_mon_decreasing F (cc a b)).
```

由定理 4.4 可知, $F(x)$ 的差商是 $f(x)$ 的中值, 进而容易得到定理的证明. 因为 4 种情形证明过程类似, 图 2 仅展示第一种情形的 Coq 证明代码.

定理 5.2 (微积分基本定理) 设 $F(x)$ 在 $[a, b]$ 上一致可导, 其导数为 $f(x)$, 则有 Newton-Leibniz

```

Theorem Theorem5_1 : forall F f a b , derivative F f a b ->
  ((forall x, x ∈ [a,b] -> f(x)>=0) -> mon_increasing F [a,b]) /\
  ((forall x, x ∈ [a,b] -> f(x)<=0) -> mon_decreasing F [a,b]) /\
  ((forall x, x ∈ [a,b] -> f(x)>0) -> strict_mon_increasing F [a,b]) /\
  ((forall x, x ∈ [a,b] -> f(x)<0) -> strict_mon_decreasing F [a,b]).

Proof.
  intros.
  apply Itheorem4_4 in H. destruct H.
  unfold diff_quo_median in H.
  assert(forall x0 x y a b, x0 ∈ [x, y] ->
    x ∈ [a, b] -> y ∈ [a, b] -> x0 ∈ [a, b]) as Domain_x0.
  { intros.
    destruct H1, H2, H3. split; auto. split.
    apply Rle_trans with (r2:=x); tauto.
    apply Rlc_trans with (r2:=y); tauto. }
  repeat split; intros.
  - unfold mon_increasing; intros.
    generalize H2 as H3; intro.
    generalize H2 as H3; intro.
    apply H in H2.
    destruct H3, H4, H2, H2, H2, H6, H7.
    apply Domain_x0 with (x0:=x0)(x:=x) in H4; auto.
    apply H1 in H4.
    apply Rle_trans with (r1:=0) in H7.
    apply Rmult_le_compat_r with (r:= y-x) in H7.
    rewrite Rmult_0_l in H7.
    unfold Rdiv in H7; rewrite Rinv_mult_rgt0 in H7; auto.
    apply Rplus_le_reg_r with (r:=-F x). rewrite Rplus_opp_r; auto.
    apply Rlt_le; auto. apply Rge_le; auto.

```

图 2 定理 5.1 部分 Coq 证明代码

公式

$$\int_a^b f(x) dx = F(b) - F(a). \quad (5.1)$$

```

Theorem Theorem5_2 : forall F f a b , derivative F f a b ->
  (fun u v => F v - F u) ∫ f (x) dx a b.

```

已知 $F(x)$ 在 $[a, b]$ 上一致可导, 其导数为 $f(x)$, 由定理 4.4 可得 $F(x)$ 的差商是 $f(x)$ 的中值, 再由定理 4.2 可知, $F(b) - F(a)$ 是 $f(x)$ 在 $[a, b]$ 上的定积分. Newton-Leibniz 公式只是 $f(x)$ 在 $[a, b]$ 上的定积分的记号. 该定理完整证明代码如图 3 所示.

定理 5.3 (变上限定积分的可导性) 设 $f(x)$ 在 $[a, b]$ 上一致连续并且其定积分存在, 定义 $G(x) = \int_a^x f(t) dt$, 则 $G(x)$ 的差商在 $[a, b]$ 上一致可导, 并且 $G'(x) = f(x)$.

```

Theorem Theorem5_2 : forall F f a b , derivative F f a b ->
  (fun u v => F v - F u) ∫ f (x) dx a b.

Proof.
  intros.
  apply Theorem4_4 in H.
  destruct H.
  apply Theorem4_2 in H; auto.
  destruct H; auto.

Qed.

```

图 3 定理 5.2 的 Coq 证明代码

```
Theorem Theorem5_3 : forall G f a b, uniform_continuous f a b ->
  (exists (S: R->R->R), integ_sys' S f a b /\
  forall x, x ∈ [a,b] -> G(x) = S a x) ->
  derivative G f a b.
```

由函数 $G(x)$ 的定义可知, $G(v) - G(u)$ 为 $f(x)$ 在 $[a, b]$ 上的积分系统, 容易推出 $G(x)$ 的差商是 $f(x)$ 的中值, 又因为 $f(x)$ 在 $[a, b]$ 上一致连续, 由定理 4.4 可知, $G(x)$ 一致可导且导函数为 $f(x)$. 该定理完整证明代码如图 4 所示.

下面证明 Taylor 公式. 首先给出一个预备定理.

定理 5.4 (Taylor 公式预备定理) 若 H 在 $[a, b]$ 上 n 阶一致 (强) 可导 (n 为正整数), 且有

(1) $k = 0, 1, \dots, n-1$ 时, $H^{(k)}(a) = 0$;

(2) 在 $[a, b]$ 上有 $m \leq H^{(n)}(x) \leq M$, 则在 $[a, b]$ 上有 $\frac{m(x-a)^k}{k!} \leq H^{(n-k)}(x) \leq \frac{M(x-a)^k}{k!}$, $k = 0, 1, \dots, n$.

```
Theorem Taylor_Lemma : forall H a b (n:nat), (n>0)%nat ->
  N_uniform_str_derivability H a b n ->
  (forall k, (k<n)%nat -> N_der H k a = 0 /\ N_der H 0 = H) ->
  forall m M, (forall x, x ∈ [a,b] -> m <= N_der H n x <= M) ->
  forall k, (k<=n)%nat -> forall x, x ∈ [a,b] ->
  m*power k (x-a)/INR(factorial k) <= N_der H (n-k) x <=
  M*power k (x-a)/INR(factorial k).
```

Proof. Admitted.

```
Theorem Theorem5_3 : forall G f a b, uniform_continuous f a b ->
  (exists (S: R->R->R), integ_sys' S f a b /\
  forall x, x ∈ [a,b] -> G(x) = S a x) ->
  derivative G f a b.
```

Proof.

```
intros.
destruct H0, H0. rename x into S.
assert(diff_quo_median G f a b).
{ unfold diff_quo_median. intros.
  unfold integ_sys' in H0. destruct H0.
  unfold additivity' in H0. unfold median in H3.
  assert(a ∈ [a, b] /\ u ∈ [a, b] /\ v ∈ [a, b]).
  { split; try tauto. destruct H2, H2.
    split; auto. split.
    apply Req_le; reflexivity.
    apply Rlt_le; auto. }
  apply H0 in H4.
  generalize (H1 u); intro.
  generalize (H1 v); intro.
  rewrite H5; try tauto. clear H5.
  rewrite H6; try tauto. clear H6.
  assert(v - u > 0) as gt_v_u . { tauto. }
  apply H3 in H2. destruct H2, H2, H2, H5.
  exists x,x0. split; auto. split; auto.
  apply Rmult_le_r with(r:=v-u); auto.
  rewrite Rplus_comm in H4; rewrite <- H4.
  unfold Rminus. rewrite Rplus_assoc.
  rewrite Rplus_opp_r; rewrite Rplus_0_r.
  unfold Rdiv. rewrite Rinv_mult_rgt0; auto. }
apply Theorem4_4. split; auto.
```

Qed.

图 4 定理 5.3 Coq 证明代码

定理 5.5 (Taylor 公式) 若 $F(x)$ 在 $[a, b]$ 上 n 阶一致 (强) 可导, 且在 $[a, b]$ 上有 $|F^n(x)| \leq M$, 对 $[a, b]$ 上任意点 c 和 x , 记

$$T_n(x, c) = F(c) + F^{(1)}(c)(x - c) + \frac{F^{(2)}(c)(x - c)^2}{2!} + \cdots + \frac{F^{(n)}(c)(x - c)^n}{n!},$$

则有

$$|F(x) - T_n(x, c)| \leq M \frac{(x - c)^n}{n!}.$$

先使用关键字 `Fixpoint` 递归地给出定理 5.5 中 Taylor 公式的 Coq 描述:

```
Fixpoint Taylor_Formula F (x:R) c n :=
  match n with
  | 0 => 0
  | S p => (N_der F p c)*(power p (x-c))/INR(factorial p)+
    Taylor_Formula F x c p
  end.
```

对应定理 5.5 的 Coq 描述如下:

```
Theorem Taylor_Theorem : forall F a b n M, (n>0)%nat ->
  N_uniform_str_derivability F a b n ->
  (forall F, N_der F 0 = F) ->
  (forall x, x ∈ [a, b] -> Rabs(N_der F n x)<=M) ->
  forall c x, c ∈ [a, b] -> x ∈ [a, b] ->
  Rabs(F(x)-(Taylor_Formula F x c n)) <=
  M*Rabs(power n (x-c))/INR(factorial n).
Proof. Admitted.
```

这里同样也采用了 Coq 中的 `Admitted` 命令, 而省略了定理 5.4 和 5.5 的详细证明代码. 核心代码见附录.

6 结论

本文利用交互式定理证明工具 Coq, 在给出 Coq 的一些预备知识后, 严格按照文献 [26] 中的结构, 给出相应定义和定理的精确 Coq 描述代码, 建立了第三代微积分理论的机器证明框架系统, 包括一致连续、一致 (强) 可导、积分系统、积分严格不等式等定义及估值定理的形式化描述和证明; 在此基础上, 迅速而自然地实现了微积分的基本定理: 函数的单调性与导函数的关系定理、Newton-Leibniz 公式、变上限积分可导性和 Taylor 公式的形式化验证.

本文所搭建的机器证明系统是第三代微积分首次形式化实现, 通过证明辅助工具 Coq 给出了其理论的完整构架. 传统微积分的形式化实现在 Coq 库中已有一些工作, 较为通用的可参见文献 [41]. 在传统微积分中, 很多定理的证明要用中值定理, 如导数正则函数增和 Taylor 公式等, 而中值定理的证明要用实数理论. 第三代微积分理论表明, 这些定理可以不依赖实数理论 (参见文献 [25, 26]). 在我们实现第三代微积分形式化的过程中, 也未直接用到实数完备性公理. 但对于定积分的存在问题则需要依赖实数理论 (参见文献 [25, 26]), 由于本系统直接调用了 Coq 中已有的实数库, 已包含实数的完备性, 当然可以进一步对定积分的存在性问题等进行形式化处理, 本文不再赘述.

本系统基于 Coq8.9.0 实现, 包括 55 个定义和 71 个定理, 约 3,700 行代码, 全部的定义和定理均给出了准确的机器证明代码, 并且编译运行通过. 完整代码可见 https://github.com/LittleGavin/calculus_3rd.

在形式化验证过程中, 对定理证明辅助工具 Coq 的使用有几点笔记如下.

- Coq 的形式化语言与自然数学语言十分相似, 可以看成是一种忠实的“翻译”. 当然, 这种“翻译”是建立在严格、准确的形式化基础上的. Coq 定理证明过程具有规范、严格、可靠的特点.

- Coq 证明体现其交互性, 定理的前提条件会实时地显示在编译界面的交互窗口; 证明过程中的每一个细节, 都需要经过 Coq 的检测, 若运行失败, 则会在提示窗口中显示失败的原因, 方便使用者修改和纠正.

- Coq 证明具有高效、智能的特点, 充分运用 Coq 的智能性, 优化证明策略, 可以写出更简明、可读、甚至结构化的证明代码.

需要说明, 在本系统中, 读者即便暂时忽略定理证明详细过程, 仅通过学习具体定义和定理的精确 Coq 描述, 也可领会第三代微积分理论的机器证明框架. 本文是实践研究人员利用计算机学习、理解、构建教育乃至发展数学的一个尝试.

致谢 2021 年即将迎来张景中院士和杨路先生 85 华诞. 张先生和杨先生自 20 世纪 80 年代即追随吴文俊院士开展数学定理机器证明的研究, 在可读性证明、数值并行法、数值符号混合计算和不等式机器证明等方面做出突出成果. 本文作者 20 世纪 90 年代关注到两位先生在定理机器证明中的相关成果, 又有幸与杨先生合作多年, 收益良多. 谨以此文庆祝张景中院士和杨路先生 85 大寿. 祝两位先生健康长寿.

参考文献

- 1 Beeson M. Mixing computations and proofs. *J Formalized Reason*, 2016, 9: 71–99
- 2 Gonthier G. Formal proof—the four-color theorem. *Notices Amer Math Soc*, 2008, 55: 1382–1393
- 3 Knies R. Six-year journey leads to proof of Feit-Thompson theorem. <https://phys.org/news/2012-10-six-year-journey-proof-feit-thompson-theorem.html/>, 2012
- 4 Gonthier G, Asperti A, Avigad J, et al. A machine-checked proof of the odd order theorem. In: *Interactive Theorem Proving. Lecture Notes in Computer Science*, vol. 7998. Berlin: Springer-Verlag, 2013, 163–179
- 5 Hales T C, Adams M, Bauer G, et al. A formal proof of the Kepler conjecture. *Forum Math Pi*, 2017, 5: e2
- 6 Wu W T. *Mechanical Theorem Proving in Geometries: Basic Principles*. New York: Springer-Verlag, 1994
- 7 Yang L, Xia B C. *Machine Proof and Automatic Discovery of Inequalities (in Chinese)*. Beijing: Science Press, 2008 [杨路, 夏壁灿. 不等式机器证明与自动发现. 北京: 科学出版社, 2008]
- 8 Yu W S, Sun T Y, Fu Y S. *Machine Proof System of Axiomatic Set Theory (in Chinese)*. Beijing: Science Press, 2020 [郁文生, 孙天宇, 付尧顺. 公理化集合论机器证明系统. 北京: 科学出版社, 2020]
- 9 Bertot Y, Castéran P. *Interactive Theorem Proving and Program Development—Coq’Art: The Calculus of Inductive Constructions*. Berlin: Springer-Verlag, 2004
- 10 The Coq development team. *The Coq proof assistant reference manual (version 8.9.0)*. <https://coq.inria.fr/distrib/current/refman/>, 2019
- 11 Hales T C. Formal proof. *Notices Amer Math Soc*, 2008, 55: 1370–1380
- 12 Harrison J. Formal proof—theory and practice. *Notices Amer Math Soc*, 2008, 55: 1395–1406
- 13 Huet G, Kahn G, Paulin-Mohring C. *The Coq proof assistant: A tutorial (version 8.5)*. Technical Report 178. <https://coq.inria.fr/tutorial/>, 2016
- 14 Nipow T, Paulson L C, Wenzel M. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. Berlin: Springer-Verlag, 2002
- 15 Pierce B C, de Amorim A A, Casinghino C, et al. *Software foundations*. <http://softwarefoundations.cis.upenn.edu/>, 2017
- 16 Wiedijk F. Formal proof—getting started. *Notices Amer Math Soc*, 2008, 55: 1408–1414
- 17 Katz V J. *A History of Mathematics: An Introduction*, 3rd ed. Boston: Addison-Wesley, 2009
- 18 Kline M. *Mathematical Thought from Ancient to Modern Times*. Oxford: Oxford University Press, 1972
- 19 Ruelle D. *The Mathematician’s Brain*. Princeton-Oxford: Princeton University Press, 2007

- 20 Thurston W P. On proof and progress in mathematics. Bull Amer Math Soc, 1994, 30: 161–177
- 21 Chen G. Formalized mathematics and proof engineering (in Chinese). Commun CCF, 2016, 12: 40–44 [陈钢. 形式化数学和证明工程. 中国计算机学会通讯, 2016, 12: 40–44]
- 22 Voevodsky V. Univalent foundations of mathematics. In: Proceedings of the 18th International Workshop on Logic, Language, Information and Computation. Lecture Notes in Computer Science, vol. 6642. Berlin-Heidelberg: Springer-Verlag, 2011, 4
- 23 Villani C. Théorème Vivant. Paris: Bernard Grasset, 2011
- 24 Li X C. The Founder and Pioneer of Calculus, 3rd ed (in Chinese). Beijing: Higher Education Press, 2007 [李心灿. 微积分的创立者及其先驱 (第三版). 北京: 高等教育出版社, 2007]
- 25 Zhang J Z. Straightforward Calculus (in Chinese). Beijing: Science Press, 2010 [张景中. 直来直去的微积分. 北京: 科学出版社, 2010]
- 26 Zhang J Z, Feng Y. A new way viewing the foundation of calculus (in Chinese). Sci China Ser A, 2009, 39: 247–256 [张景中, 冯勇. 微积分基础的新视角. 中国科学 A 辑, 2009, 39: 247–256]
- 27 Zhang J Z, Feng Y. The third generation calculus (in Chinese). Chinese J Nature, 2010, 32: 67–71 [张景中, 冯勇. 第三代的微积分. 自然杂志, 2010, 32: 67–71]
- 28 Lusternik L A, Sobolev V J. Elements of Functional Analysis, 3rd ed. New York: John Wiley & Sons, 1975
- 29 Dovermann K H. Applied calculus. <https://math.hawaii.edu/%7Eheiner/calculus.pdf>, 1999
- 30 Sparks J C. Calculus without Limits—Almost. Bloomington: AuthorHouse, 2005
- 31 Livshits M. Simplifying calculus by using uniform estimates. <http://www.mathfoolery.com/talk-2004.pdf>, 2004
- 32 Livshits M. You could simplify calculus. arXiv:0905.3611, 2009
- 33 Lin Q. Free Calculus: A Liberation from Concepts and Proofs. Singapore: World Scientific, 2008
- 34 Lin Q. Fast Calculus, 3rd ed (in Chinese). Beijing: Science Press, 2013 [林群. 微积分快餐 (第三版). 北京: 科学出版社, 2013]
- 35 Lin Q, Zhang J Z. What can be done prior to calculus (in Chinese). Stud College Math, 2019, 22: 1–15 [林群, 张景中. 微积分之前可以做些什么. 高等数学研究, 2019, 22: 1–15]
- 36 Yang L, Hou X R, Zeng Z B. A complete discrimination system for polynomials. Sci China Ser E, 1996, 39: 628–646
- 37 Zhang J Z, Liang X S. Complete discriminant system of complex coefficient polynomial and its automatic generation (in Chinese). Sci China Ser E, 1999, 29: 61–75 [张景中, 梁新松. 复系数多项式完全判别系统及其自动生成. 中国科学 E 辑, 1999, 29: 61–75]
- 38 Wu W T. Mathematics Mechanization: Mechanical Geometry Theorem-Proving, Mechanical Geometry Problem-Solving and Polynomial Equations-Solving. Mathematics and Its Applications, vol. 489. Beijing-Dordrecht: Science Press and Kluwer Academic Publishers, 2000
- 39 Wu W T. Mathematics mechanizations: Its origin, status, and prospect (in Chinese). J Systems Sci Math Sci, 2008, 28: 898–904 [吴文俊. 数学机械化研究回顾与展望. 系统科学与数学, 2008, 28: 898–904]
- 40 Zhang J Z, Li Y B. Automatic theorem proving for three decades (in Chinese). J Systems Sci Math Sci, 2009, 29: 1155–1168 [张景中, 李永彬. 几何定理机器证明三十年. 系统科学与数学, 2009, 29: 1155–1168]
- 41 Boldo S, Lelay C, Melquiond G. Coquelicot: A user-friendly library of real analysis for Coq. Math Comput Sci, 2015, 9: 41–62

附录 A 主要定理的核心机器证明代码

考虑到一些定理的证明代码较长, 不方便在正文中完整展示, 附录给出了一些主要定理的核心机器证明代码, 供读者参考.

(1) 定理 3.2 (估值定理) Coq 证明代码

```
Theorem Valuation_Theorem :forall (S:R->R->R) (f G:R->R)(a b:R),
  integ_sys S f a b ->
  forall y, y ∈ [a,b] -> (forall x:R, x ∈ [a,b] -> G x = S y x) ->
  strict_unequal S f a b ->
  forall u v, u ∈ [a,b] /\ v ∈ [a,b] /\ v - u > 0 ->
  exists x1 x2, x1 ∈ [u,v] /\ x2 ∈ [u,v] /\
  (f x1)*(v-u) <= G(v) - G(u) <= (f x2)*(v-u).
Proof.
```



```

intros.
unfold strict_unequal in H2.
generalize H as l; intro.
apply H2 with (w1:=u)(w2:=v) in H; auto. clear H2.
destruct H, H3, H4.
assert (exists x1, x1 ∈ [u,v] /\ (f x1)*(v-u) <= G(v) - G(u)).
{ generalize(classic(exists x1, x1∈[u,v] /\ (f x1)*(v-u)<=G(v)-G(u))).
  intros. destruct H6; auto.
  assert (G v - G u < S u v).
  { generalize (H ((G v - G u)/(v-u))); intros; clear H.
    unfold Rdiv in H7.
    rewrite Rinv_mult_rgt0 with (r1:=(G v-G u))(r:=(v-u)) in H7; auto.

    apply H7; intros.
    apply not_exist with (x:=x) in H6. apply Rnot_le_gt in H6.
    apply Rgt_mult in H6; auto. apply H. }
  rewrite H1 in H7; auto. rewrite H1 in H7; auto.
  assert ( S u v = S y v - S y u /\ S y v - S y u = G v - G u).
  { apply equ_s with (f:=f)(a:=a)(b:=b); auto. }
  destruct H8.
  rewrite H8 in H7; apply Rlt_irrefl in H7; contradiction. }
assert (exists x2, x2 ∈ [u,v] /\ (f x2)*(v-u) >= G(v) - G(u)).
{ generalize(classic(exists x2, x2∈[u,v] /\ (f x2)*(v-u)>=G(v)-G(u))).
  intros. destruct H7; auto.
  assert (G v - G u > S u v).
  { generalize (H2 ((G v - G u)/(v-u))); intros; clear H2.
    unfold Rdiv in H8.
    rewrite Rinv_mult_rgt0 with (r1:=(G v-G u))(r:=(v-u)) in H8; auto.
    apply H8; intros.
    apply not_exist with (x:=x) in H7. apply Rnot_ge_lt in H7.
    apply Rlt_mult in H7; auto. apply H2. }
  rewrite H1 in H8; auto. rewrite H1 in H8; auto.
  assert ( S u v = S y v - S y u /\ S y v - S y u = G v - G u).
  { apply equ_s with (f:=f)(a:=a)(b:=b); auto. }
  destruct H9.
  rewrite H9 in H8; apply Rlt_irrefl in H8; contradiction. }
destruct H6, H6, H7, H7.
exists x, x0. split; auto. split; auto. split; auto.
apply Rge_le; auto.
Qed.

```

(2) 定理 5.4 (Taylor 公式预备定理) 形式化证明代码

为证定理 5.4, 引进一些辅助定义和引理, 其 Coq 描述如下:

```

Fixpoint factorial (n:nat) : nat :=
  match n with
  | 0 => 1
  | S p => S p * (factorial p)
  end.

```

```

Fixpoint power (n:nat) x : R :=
  match n with
  | 0 => 1

```

```

| S p => x * power p x
end.

Fixpoint accumulation x1 x2 (n i:nat) : R :=
  match i with
  | 0 => power n x1
  | S p => (power (n - S p) x1)*(power (S p) x2)+accumulation x1 x2 n p
  end.

Lemma fix_property : forall x y (n i:nat), (i<=n)%nat ->
  accumulation x y (n+1) i = x*accumulation x y n i.
Proof. Admitted.

Lemma decompose_th : forall x y (n:nat),
  power (n+1) x - power (n+1) y = (x-y)*accumulation x y n n.
Proof. Admitted.

Lemma strder_Deduce_der : forall F f a b,
  a<b -> str_derivative F f a b -> derivative F f a b.
Proof. Admitted.

Lemma accumul_th : forall a b n,
  a>=0 -> b>=0 -> a>b -> accumulation a b n n <= INR(n+1)* power n a /\
  accumulation a b n n >= INR(n+1)%nat* power n b.
Proof. Admitted.

Lemma factorial_Sn : forall n, factorial (S n) = ((S n)*factorial n)%nat.
Proof. Admitted.

Lemma factorial_pos : forall n, 0 < INR(factorial n).
Proof. Admitted.

Lemma power_inv_fac_der : forall (x m a b:R)(n:nat), x ∈ [a, b] ->
  str_derivative (fun x => m*power(S n)(x-a)/INR(factorial(S n)))
  (fun x => m*power n (x-a)/INR(factorial n)) a b.
Proof. Admitted.

Theorem Taylor_Lemma : forall H a b (n:nat), (n>0)%nat ->
  N_uniform_str_derivability H a b n ->
  (forall k, (k<n)%nat -> N_der H k a = 0 /\ N_der H 0 = H ) ->
  forall m M, (forall x, x ∈ [a,b] -> m <= N_der H n x <= M) ->
  forall k, (k<=n)%nat -> forall x, x ∈ [a,b] ->
  m*power k (x-a)/INR(factorial k) <= N_der H (n-k) x <=
  M*power k (x-a)/INR(factorial k).
Proof.
  intros H a b n H0 H1 H2 m M H3 k H4.
  induction k.
  - simpl. intros. unfold Rdiv.
    repeat rewrite Rinv_r_simpl_l; try apply R1_neq_R0.
    apply H3 in H5; try apply Nat.le_refl.
    rewrite Nat.sub_0_r; auto.
  - assert((k<=n)%nat) as le_k_n.
    { apply Nat.le_trans with (m:= S k); auto. }

```

```

generalize(IHk le_k_n) as IHk_x; intros. clear IHk.
generalize(power_inv_fac_der x m a b k H5); intro.
generalize(power_inv_fac_der x M a b k H5); intro.
assert(str_derivative (N_der H (n-S k))(N_der H(n-k)) a b).
{ assert((n-S k=n-k-1 /\ n-k=S(n-k-1))%nat).

  { split. auto. rewrite <- Nat.sub_add_distr.
    rewrite Nat.add_1_r; auto. rewrite <- Nat.add_1_r.
    rewrite Nat.sub_add; auto. apply plus_le_reg_l with (p:=k).
    rewrite Nat.add_comm with (m:=(n-k)%nat).
    rewrite Nat.sub_add. rewrite Nat.add_1_r; auto.
    apply Nat.le_trans with (m:=S k); auto. }
  destruct H8. rewrite H8. rewrite H9 at 2.
  apply N_uniform_str_derivative. }
assert(a<=x) as le_a_x.
{ unfold In in H5. unfold cc in H5. tauto. }
split.
+ destruct le_a_x.
* apply Theorem3_1_1' with (c:=-1) in H6.
  apply (Theorem3_1_2' _ _ _ _ _ H8) in H6.
  apply Theorem4_3 in H6. destruct H6. clear H10.
  unfold diff_quo_median in H6.
  assert(a ∈ [a, b] /\ x ∈ [a, b] /\ x - a > 0).
  { destruct H5, H10. repeat split; auto.
    apply Rle_refl. apply Rlt_le; auto.
    apply Rgt_minus; auto. }
  apply H6 in H10. destruct H10, H10, H10, H11, H12.
  clear H6 H13. unfold plus_Fu in H12; unfold mult_real_f in H12.
  assert(n-S k<n)%nat.
  { apply Nat.sub_lt; auto. apply Nat.lt_0_succ. }
  apply H2 in H6. destruct H6. rewrite H6 in H12.
  rewrite Rminus_diag_eq with (r1:=a) in H12; auto.
  rewrite Rplus_0_l in H12.
  assert(power (S k) 0 = 0).
  { simpl. rewrite Rmult_0_l; auto. }
  rewrite H14 in H12. rewrite Rmult_0_r in H12.
  unfold Rdiv in H12; rewrite Rmult_0_l in H12.
  rewrite Rmult_0_r in H12.
  assert(x0 ∈ [a, b]).
  { destruct H5, H15, H10, H17. split; auto.
    split; auto. apply Rle_trans with (r2:=x); auto. }
  apply IHk_x in H15.
  apply Rplus_le_reg_r with
    (r:=(m*power(S k)(x-a)/INR(factorial(S k)))).
  rewrite Rplus_opp_r. destruct H15. clear H16.
  apply Rplus_le_compat_r with
    (r:=(m*power k (x0-a)/INR(factorial k))) in H15.
  rewrite Rplus_opp_r in H15. unfold Rdiv in H15.
  repeat rewrite Ropp_mult_distr_l_reverse with (r1:=1) in H12.
  repeat rewrite Rmult_1_l in H12.
  apply Rle_trans with (r1:=0) in H12; auto.
  unfold Rdiv. rewrite Rminus_0_r in H12.

```

```

    apply Rmult_le_reg_r with (r:= / (x - a)).
    apply Rinv_0_lt_compat; apply Rlt_Rminus; auto.
    rewrite Rmult_0_l; auto.

* rewrite <- H9. rewrite Rminus_diag_eq; auto.
  simpl. rewrite Rmult_0_l. rewrite Rmult_0_r.
  unfold Rdiv; rewrite Rmult_0_l.
  assert(n - S k < n)%nat.
  { apply Nat.sub_lt; auto. apply Nat.lt_0_succ. }
  apply H2 in H10. destruct H10;
  rewrite H10; apply Rle_refl.
+ destruct le_a_x.
* apply Theorem3_1_1' with (c:=-1) in H7.
  apply (Theorem3_1_2' _ _ _ _ _ H8) in H7.
  apply Theorem4_3 in H7. destruct H7. clear H10.
  unfold diff_quo_median in H7.
  assert(a ∈ [a, b] /\ x ∈ [a, b] /\ x - a > 0).
  { destruct H5, H10. repeat split; auto. apply Rle_refl.
    apply Rlt_le; auto. apply Rgt_minus; auto. }
  apply H7 in H10. destruct H10, H10, H10, H11, H12.
  clear H7 H12. unfold plus_Fu in H13; unfold mult_real_f in H13.
  assert(n - S k < n)%nat.
  { apply Nat.sub_lt; auto. apply Nat.lt_0_succ. }
  apply H2 in H7. destruct H7. rewrite H7 in H13.
  rewrite Rminus_diag_eq with (r1:=a) in H13; auto.
  rewrite Rplus_0_l in H13.
  assert(power (S k) 0 = 0).
  { simpl. rewrite Rmult_0_l; auto. }
  rewrite H14 in H13. rewrite Rmult_0_r in H13.
  unfold Rdiv in H13; rewrite Rmult_0_l in H13.
  rewrite Rmult_0_r in H13.
  assert(x1 ∈ [a, b]).
  { destruct H5, H15, H11, H17. split; auto. split; auto.
    apply Rle_trans with (r2:=x); auto. }
  apply IHk_x in H15.
  apply Rplus_le_reg_r with
    (r:=(M*power(S k)(x-a)/INR(factorial(S k)))).
  rewrite Rplus_opp_r. destruct H15. clear H15.
  apply Rplus_le_compat_r with
    (r:=(M*power k (x1-a)/INR(factorial k))) in H16.
  rewrite Rplus_opp_r in H16. unfold Rdiv in H16.
  repeat rewrite Ropp_mult_distr_l_reverse with (r1:=1) in H13.
  repeat rewrite Rmult_1_l in H13.
  apply Rle_trans with (r3:=0) in H13; auto.
  unfold Rdiv. rewrite Rminus_0_r in H13.
  apply Rmult_le_reg_r with (r:=/(x-a)).
  apply Rinv_0_lt_compat; apply Rlt_Rminus; auto.
  rewrite Rmult_0_l; auto.
* rewrite <- H9. rewrite Rminus_diag_eq; auto.
  simpl. rewrite Rmult_0_l. rewrite Rmult_0_r.
  unfold Rdiv; rewrite Rmult_0_l.
  assert(n - S k < n)%nat.

```

```

    { apply Nat.sub_lt; auto. apply Nat.lt_0_succ. }

    apply H2 in H10. destruct H10; rewrite H10; apply Rle_refl.
Qed.

```

(3) 定理 5.5 (Taylor 公式) Coq 证明代码

同样, 为证定理 5.5, 其预备命题也以引理形式给出, 其 Coq 描述如下:

```

Lemma Rabs_power : forall a b n,
  Rabs (power n (-a-b))=Rabs(power n (a+b)).
Proof. Admitted.

```

```

Lemma power_x_pos : forall x n, 0<=x -> 0 <= power n x.
Proof. Admitted.

```

```

Lemma Domain_x_c : forall a b x c,
  c ∈ [a, b] -> x ∈ [a, b]-> x ∈ [c, b] \ / x ∈ [a, c].
Proof. Admitted.

```

```

Lemma Taylor_Nder: forall F n M a b c,
  N_uniform_str_derivability F a b n -> c ∈ [a, b] ->
  (forall x, x ∈ [a, b]->Rabs(N_der F n x)<=M) ->
  (N_uniform_str_derivability(fun x:R=>F x-Taylor_Formula F x c n) c b n /\
  (forall k, (k<n)%nat->N_der(fun x=>F x-Taylor_Formula F x c n) k c = 0)
  /\ N_der (fun x => F x-Taylor_Formula F x c n) 0=
  (fun x => F x-Taylor_Formula F x c n)/\ (forall x, x ∈ [a,b]->-M <=
  N_der (fun x => F x-Taylor_Formula F x c n) n x <= M)).
Proof. Admitted.

```

```

Theorem Taylor_Theorem : forall F a b n M, (n>0)%nat ->
  N_uniform_str_derivability F a b n ->
  (forall F, N_der F 0= F) ->
  (forall x, x ∈ [a, b] -> Rabs(N_der F n x)<=M) ->
  forall c x, c ∈ [a, b] -> x ∈ [a, b] ->
  Rabs(F(x)-(Taylor_Formula F x c n))<=M*Rabs(power n (x-c))/INR(factorial n).
Proof.

```

```

  intros.
  generalize H3 as Domain_c; intro.
  apply Domain_x_c with (x:=x) in H3; auto.
  destruct H3.
  - apply Rabs_le. rewrite Rabs_pos_eq; try apply power_x_pos.
    assert(-(M*power n (x-c))/INR(factorial n))=
      -M*power n (x-c)/INR(factorial n)).
    { unfold Rdiv. repeat rewrite Rmult_assoc.
      rewrite Ropp_mult_distr_l; auto. }
  rewrite H5. clear H5.
  assert(N_der (fun x => F x-Taylor_Formula F x c n) 0 x =
    F x-Taylor_Formula F x c n). { rewrite H1; auto. }
  rewrite <- H5. clear H5. rewrite <- Nat.sub_diag with (n:=n).
  apply Taylor_Nder with (c:=c)(M:=M) in H0; auto. destruct H0.
  apply Taylor_Lemma with (b:=b); try tauto. intros. apply H5 in H6.
  split; auto. destruct H5; auto. destruct H6.

```

```

intros. apply H7; auto. split. destruct H4; auto.
destruct Domain_c, H10, H8, H12. split; auto.
apply Rle_trans with (r2:=c); auto. apply Nat.le_refl.
destruct H3, H5. apply Rge_le. apply Rge_minus; apply Rle_ge; auto.
- assert(exists u, x = -u). { exists (0-x). ring. }
destruct H5. rename x0 into u. rewrite H5.
assert(u ∈ [-c, -a]).
{ destruct H3. split. apply Ropp_lt_contravar; auto.
  assert(u = - x). { rewrite H5. ring. }
  destruct H6. rewrite H7. split; apply Ropp_le_contravar; auto. }
rewrite Rabs_power. apply Rabs_le.
rewrite Rabs_pos_eq; try apply power_x_pos.
assert(-(M*power n (u+c)/INR(factorial n))=
  -M*power n (u+c)/INR(factorial n)).
{ unfold Rdiv. repeat rewrite Rmult_assoc.
  rewrite Ropp_mult_distr_l; auto. }
rewrite H7. clear H7.
assert(u+c=u-(-c)). { ring. } rewrite H7.
assert(N_der (fun x => F (-x)-Taylor_Formula F (-x) c n) 0 u =
  F (-u)-Taylor_Formula F (-u) c n). { rewrite H1; auto. }
rewrite <- H8. clear H8.
apply Taylor_Nder' with (c:=c)(M:=M) in H0; auto.
destruct H0. rewrite <- Nat.sub_diag with (n:=n).
apply Taylor_Lemma with (b:=(-a)); auto.
intros. destruct H8. apply H8 in H9; auto.
intros. destruct H8, H10 ; auto.
assert(u+c=u-(-c)). { ring. }
apply Rge_le. rewrite H7. destruct H6, H8.
apply Rge_minus; apply Rle_ge; auto.
Qed.

```

考虑到文章篇幅有限, 本文仅给出了主要定理的核心代码, 涉及引理采用 `Admitted` 命令, 省略其证明过程, 完整代码读者可前往网址 https://github.com/LittleGavin/calculus_3rd 下载查阅。

A mechanized proof system of the third generation calculus in Coq

Liquan Guo, Yaoshun Fu & Wensheng Yu

Abstract Based on the proof assistant Coq, a formal verification of the third generation calculus theory advocated by Academicians Qun Lin and Jingzhong Zhang is completely implemented, including Coq descriptions of all the definitions and theorems in Jingzhong Zhang's paper entitled "A new way viewing the foundation of calculus". A proof code of all the theorems is given without exception, and all the formalization processes have been verified by Coq. The formal proof demonstrates that the Coq-based mechanized proof has the characteristics of readability and interactivity. The proof process is standardized, rigorous and reliable. This paper is an attempt of the idea that researchers learn, understand, construct, and even educate mathematics by computer assistant.

Keywords proof assistant Coq, the third generation calculus, formalization, mechanized proof

MSC(2010) 03B35, 44A45, 68Q45

doi: 10.1360/SSM-2019-0327