q-類似の Coq による形式化

アドバイザー： Jacques Garrigue 教授
学籍番号： 322101289
氏名： 中村　薫

2023 年 2 月 6 日

序文

主結果

本論文の主結果は, q-類似の初等的な結果を Coq によって形式化するものである. 形式化とは, 証明のために用意された人工言語に数学的な主張とその証明を翻訳し, 証明が論理学の推論規則に沿って正しく書かれていることをコンピュータを用いて機械的に検証することである. また Coq はこの形式化を行うためのソフトウェアの一つであり, このようなソフトウェアを総称して定理証明支援系と呼ぶ.

具体的な形式化の対象は Victor Kac, Pokman Cheung の *Quantum Calculus* [4] の 4 章 (4.1) 式の q-Taylor 展開, 及びその系として得られる Gauss's binomial formula である. 本論文での q-類似に関する定義や定理, 証明は [4] によるものだが, その形式化を行ったという点において独自性がある. 形式化したコード全体は `https://github.com/nakamurakaoru/q-analogue/tree/thesis` [7] にある. q_analogue.v が [4] の形式化をしたファイルであり, q_tool.v は直接 q-類似に関係はしないが, 形式化をするために自分で用意した補題をまとめたファイルである (どちらもテキストファイルである). 本論文の主目的である q-Taylor 展開, Gauss's binomial formula はそれぞれ q_analogue.v の 1417 行目, 1512 行目にある.

q-類似

q-類似は, 学部 4 年次に卒業研究のテーマとして扱ったものである. 初めは Euler の分割に登場し, その後 Gauss や Heine らによって超幾何関数の研究の中で進展していった. q-類似を系統的に発展させたのは Jackson であり, また q-積分の概念も彼によって導入された.

q-類似とは, 実数パラメータ q, 実数上の関数 f に対して

$$D_q f(x) \coloneqq \frac{f(qx) - f(x)}{(q-1)x}$$

で定義される q-微分を出発点とし, この q-微分に対してうまく振る舞い, かつ q を極限で 1 に近づけると通常の定義に一致するように数学の諸概念を一般化するものである. 例えば, 自然数 n について x^n を定義に沿って q-微分すると,

$$D_q x^n = \frac{(qx)^n - x^n}{(q-1)x} = \frac{q^n - 1}{q-1} x^{n-1}$$

となる. 通常の微分では, $(x^n)' = nx^{n-1}$ となることと比較して, n の q-類似 $[n]$ を

$$[n] = \frac{q^n - 1}{q-1}$$

と定める. また, $(x-a)^n$ の q-類似は,

$$(x-a)_q^n \coloneqq \begin{cases} 1 & (n=0) \\ (x-a)(x-qa)\cdots(x-q^{n-1}a) & (n \geq 1) \end{cases}$$

と定義することで, q-微分と自然数の q-類似に対してうまく振る舞う, つまり

$$D_q (x-a)_q^n = [n](x-a)_q^{n-1}$$

が成り立つ. 更に, 階乗と二項係数の q-類似をそれぞれ

$$[n]! \coloneqq \begin{cases} 1 & (n=0) \\ [n] \times [n-1] \cdots [1] & (n \geq 1) \end{cases}$$

$$\begin{bmatrix} n \\ j \end{bmatrix} \coloneqq \frac{[n]!}{[j]![n-j]!}$$

で定めれば, q-Taylor 展開, Gauss's binomial formula は以下のように書ける.

q-Taylor 展開

$f(x)$ を N 次の実係数多項式とする. 任意の $c \in \mathbb{R}$ について

$$f(x) = \sum_{j=0}^{N} (Dq^j f)(c) \frac{(x-c)_q^j}{[j]!}$$

が成り立つ.

Gauss's binomial formula

$x, a \in \mathbb{R}, n \in \mathbb{Z}_{>0}$ について

$$(x+a)_q^n = \sum_{j=0}^{n} \begin{bmatrix} n \\ j \end{bmatrix} q^{\frac{j(j-1)}{2}} a^j x^{n-j}$$

が成り立つ.

この 2 つを形式化することが本論文の主目的である.

Coq

Coq とは, 前述のとおり定理証明支援系の 1 つであり, 人間がチェックすることが難しい複雑な証明でも正しさが保証され, また証明付きプログラミングにも応用される. Mizar, Isabelle/HOL 等他にも定理証明支援系は存在するが, 修士 1 年次後期に履修した授業で Coq の使い方を学んだため, 今回の形式化に利用した.

Coq は型付き λ 計算という理論に基づいている. λ 計算とは, 大まかにいえば計算の実行をモデル化したもので, Church によって考案された. その後, もともとは Russell のパラドクスを回避するために Russell 自身によって導入された型理論と結びついて型付き λ 計算 (こちらも Church によって考案された) として発展し, Martin-Löf によって直観主義型理論が開発され, 構成主義的な数学の基礎付けがなされた. 更により表現力の高い λ 計算の研究が続けられ, Coquand と Huet の開発した CoC(Calculus of Constructions) と呼ばれる 2 階述語論理よりも表現力の高い型付き λ 計算を用いて Coq が開発された. この CoC を用いていることから, Coq には補題とその証明を同じ言語で記述できるという特徴がある. また, 現在の Coq には Paulin-Mohring によって帰納的型を導入した λ 計算である CIC (Calculus of Inductive Constructions) が用いられている.

λ 計算については修士 1 年次に少人数クラスで学習した内容であるため, Coq との関係にも触れつつ本論文でも説明を加える. 更に, 型と形式化に関する数学の理論として, Homotopy Type Theory がある. これについては修士 2 年次に少人数クラスで学んだため, q-類似の形式化とは章を分け, 少人数クラスのまとめとして概要を述べる. 今回の証明に関しては, Coq の標準ライブラリ [2] に加えて, 数学の証明のために整備されたライブラリ群である mathcomp [5] も用いている. Coq が用いられた有名な例として, 四色定理や Feit-Thompson の定理 (奇数位数定理) などがあり, それらも mathcomp に基づいている.

実際に Coq で定義, 補題, 証明を表すコードは例えば以下のようなものである ([7] q_analogue.v 16-25 行目, ただし説明のためコードに変更を加えている).

```
Definition dq (f : R → R) x := f (q * x) - f x.

Lemma dq_prod f g x :
  dq (f ** g) x = (f (q * x)) * dq g x + (g x) * dq f x.
Proof.
  rewrite /dq.
  rewrite !mulrBr
  rewrite [g x * f (q * x)]mulrC
  rewrite [g x * f x]mulrC
  by rewrite subrKA.
Qed.
```

基本的には Lemma コマンドで証明したい補題の主張 (を Coq にあわせて翻訳したもの) を書き, Proof 以下にタクティックとよばれる命令を書くことで証明を進め, 証明が完了したところで Qed コマンドを書くというのが一連の流れである. Qed コマンドは Coq にその補題を登録する機能があるため, 他の補題の証明に利用することができる. また, Definition コマンドで自分で新しく関数を定義することができる.

上記の例について見てみると, まず Definition で dq という名前の関数を定義している. := の左の f : R → R と x は関数 dq の引数を表しており, := の右側は関数の定義である. これは, 通常の数学での

> 実数上の関数 f と実数 x に対して,
> $$d_q f(x) \coloneqq f(qx) - f(x)$$
> と定める.

という定義を形式化したものである.

次に Lemma で補題の主張を書いている. dq_prod は補題の名前であり, 他の証明に用いる際に使えるように名前を付けている. : の左側の f, g, x がこの補題の引数であり, : の右側が補題の主張である. この Lemma は,

> 関数 f, g と $x \in \mathbb{R}$ について,
> $$d_q(f(x)g(x)) = f(qx)d_q g(x) + g(x)d_q f(x)$$
> が成り立つ.

という補題の形式化である. 更に, Proof 以下について, rewrite とは等式変形を行う命令であり, rewrite /dq であれば dq を定義に基づいて計算していて, rewrite !mulrBr であれば積と差についての分配法則をできる限り繰り返している. mulrC は積の交換律のことで, [g x * f (q * x)]mulrC であれば g x * f (q * x) を f (q * x) * g x に, [g x * f x]mulrC であれば g x * f x を [f x * g x] に書き換えている. by rewrite subrKA は同じものの引き算は 0 になるという書き換えをし, 更にその書き換えで証明が完了することを表している. 最後に Qed と書くことで, dq_prod という補題を他の証明で使えるようになる.

構成

最後に構成について述べる. まず**??**節で q-類似の概要について説明し, **??**節で修士 1 年次に少人数クラスで学習した H.P.Barendregt の *Lambda Calculi with Types* [1] に沿って, 型付き λ 計算について述べる. **??**節は Coq の概要についてで, 特に**??**節で Coq や mathcomp の使い方を具体例を交えて説明するが, より詳細な情報については萩原 学/アフェルト・レナルドの *Coq/SSReflect/Mathcomp* [3] 等を参照のこと. これらの準備のもと, **??**節から本題の形式化に入る. [4] での定義, 定理を述べた後, その形式化を与え, 必要であれば形式化をするにあたっての注意点を述べることを繰り返すという流れである. 証明の方針等は基本的に [4] の通りであるが, **??**節では一部 [4] から離れ, 多項式として q-微分や q-二項式を定義しなおして形式化を行っている. これらの新たな定義が多項式に対してのもとの定義を適用したものと一致していることの証明も行っている. また, **??**章で修士 2 年次に少人数クラスで学習した内容についてまとめている.

謝辞

形式化の方針や Coq の使い方等, 本論文の作成において終始熱心に指導してくださった, アドバイザーである Jacques Garrigue 教授に心からの感謝を申し上げます.

今回の形式化の題材に選んだ q-類似について指導してくださった学部 4 年次の指導教員である古庄英和教授に感謝の意を表します.

多元数理科学研究科研究員の才川隆文氏から数多くの的確な助言をいただきました. 感謝申し上げます.

最後に, 研究室の皆様には常に刺激的な議論とアドバイスを頂き, 精神的にも絶えず支えられていました. 本

当にありがとうございます.

参考文献

[1] Henk Barendregt, *Lambda Calculi with Types*. In S. Abramsky, Dov M. Gabbay, S. E. Maibaum, *Handbook of logic in computer science (vol. 2): background: computational structures*, Oxford University Press, 1993.

[2] Coq Team, *The Coq Standard Library*, `https://coq.inria.fr/distrib/current/stdlib/`, 2023.

[3] 萩原 学/アフェルト・レナルド, *Coq/SSReflect/Mathcomp*, 森北出版, 2018.

[4] Victor Kac, Pokman Cheung, *Quantum Calculus*, Springer, 2001.

[5] Mathematical Components Team, *Mathematical Components*, `https://github.com/math-comp/math-comp`, 2023.

[6] Mathematical Components Team, *Mathematical Components compliant Analysis Library*, `https://github.com/math-comp/analysis`, 2023.

[7] 中村 薫, *q-analogue*, `https://github.com/nakamurakaoru/q-analogue/tree/thesis`, 2023.

[8] 梅村 浩, 『楕円関数論 楕円曲線の解析学』, 東京大学出版会, 2000.

[9] The Univalent Foundations Program, *Homotopy Type Theory: Univalent Foundations of Mathematics*, `https://homotopytypetheory.org/book`, 2013.