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

January 30, 2023

はじめに

q-類似の概要

q-類似：以下の 2 つの条件をみたす数学の諸概念の一般化

- $q \to 1$ とすると通常の数学に一致する
- 実数パラメータ q, 実数上の関数 f に対して

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

　で定義される q-微分に対してうまく振る舞う

q-類似を考える利点：あえてパラメータを増やすことで証明が簡単になる場合がある

Jacobi の三重積

$z, q \in \mathbb{R}, |q| < 1$ として,

$$\sum_{n=-\infty}^{\infty} q^{n^2} z^n = \prod_{n=1}^{\infty} (1 - q^{2n})(1 + q^{2n-1} z)(1 + q^{2n-1} z^{-1})$$

が成り立つ.

q-微分

まず q-微分の定義から始める. q を 1 でない実数とする.

Definition 2.1 ([1] p1 (1.1), p2 (1.5))

関数 $f : \mathbb{R} \to \mathbb{R}$ に対して, $f(x)$ の q 差分 $d_q f(x)$ を,

$$d_q f(x) := f(qx) - f(x)$$

と定める. 更に, $f(x)$ の q 微分を $D_q f(x)$ を,

$$D_q f(x) := \frac{d_q f(x)}{d_q x} = \frac{f(qx) - f(x)}{(q-1)x}$$

と定める.

自然数の q-類似

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.

$(x-a)^n$ の q-類似

続いて $(x-a)^n$ の q-類似を定義し, その性質を調べる.

Definition 2.2 ([1] p8 Definition (3.4))

$x, a \in \mathbb{R}, n \in \mathbb{Z}_{\geq 0}$ に対して, $(x-a)^n$ の q-類似 $(x-a)_q^n$ を,

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

と定義する.

Proposition 2.3

$n \in \mathbb{Z}_{>0}$ に対し,

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

が成り立つ.

Proof.

q-Taylor 展開

Theorem 2.4 ([1] p12 Theorem 4.1)

$f(x)$ を, N 次の実数係数多項式とする. 任意の $c \in \mathbb{R}$ に対し,

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

が成り立つ.

Proof.

$\frac{(x-a)_q^n}{[n]!}$ が, a, D_q に対して Theorem **??**の三条件をみたすことを確かめればよい. (i), (ii) は $(x-a)_q^n$ の定義から, (iii) は Proposition2.3 から分かる. □

Gauss's binomial formula

Lemma 2.5 ([1] p15 Example (5.5))

$n \in \mathbb{Z}_{>0}$ について,

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

が成り立つ. この式は Gauss's binomial formula と呼ばれる.

Proof.

$f = (x+a)_q^n$ とすると, 任意の正整数 $j \le n$ に対して,

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

であり, また

$$(x+a)_q^m = (x+a)(x+qa)\cdots(x+q^{m-1}a)$$

形式化

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.

Coq の使い方 コマンド

Coq に与える命令はコマンドとタクティックの 2 種類がある.

Variable `Variable [変数]: [型]` で, 特定の型を持つ変数を宣言できる. 例えば

`Variable n: nat`

で, 変数 n が自然数型 `nat` の要素であることを表している.

Definition 新たに関数を定義するためのコマンドで,

`Definition [関数名] ([引数]: [引数の型]): [関数の型] := [関数の定義式]`

という形で用いる.

Lemma 補題を宣言するためのコマンドで,

`Lemma [補題名] ([引数]: [引数の型]): [補題の主張]`

という形である. `Lemma` の代わりに `Theorem`, `Corollary` 等でも同じ機能をもつ.

Proof/Qed `Proof` は `Lemma` の後に書いて補題の主張と証明を分ける (実際には省略可能で, 人間の見やすさのために書いている). 証明を完了させて `Qed` を書くことで Coq に補題を登録することができ, 他の補題の証明に使えるようになる.

Coq の使い方 タクティック

タクティックは `Proof...Qed` の間に使われる．よく使われるタクティックは `move`, `apply`, `rewrite` の 3 つである．

`move` `move=> H` でゴールの前提に `H` という名前をつけてコンテクストに移動する．また `move: H` で補題 `H` もしくはコンテクストに存在する `H` をゴールの前提に移す．

`apply` 補題 *lem* が `P1 → P2` という形で，ゴールが `P2` のとき，`apply` *lem* でゴールを `P1` に変える．コンテクストの仮定 `H` が `P1` → *P2* であれば `apply H` で同じことができる．

`rewrite` *def* が定義のとき，`rewrite /`*def* でゴールに出現している *def* を展開する．
また，補題 *lem* が `A = B` という形のとき，`rewrite` *lem* でゴールに出現する `A` を `B` に書き換える (ただし，*lem* が `H → (A = B)` という形であるとき，新たなゴールとして `H` が追加される)．更に，`rewrite` *lem* `in H` で，コンテクストの `H` に出現する `A` を `B` に書き換える．`apply` と同じく，使いたい等式が仮定にある場合も同じように使える．更に，`rewrite` は繰り返し回数や適用箇所を指定できる．

Coq の使い方 3 モーダスポーネンス

モーダスポーネンスの形式化

Coq の使い方 4 代入計算

代入に関する簡単な計算を形式化

Dq

Dq

の定義

qnat

qnat

の定義, x^n の微分

qbinom_pos

$(x - a)_q^n$ の形式化

Dq qbinom_pos

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

qbinom

関数から多項式へ

Dqp

を定義

q-Taylor 展開の形式化

Gauss's binomial formula の形式化

Victor Kac, Pokman Cheung, *Quantum Calculus*, Springer, 2001.