q-類似の Coq による形式化

中村薫　名古屋大学

はじめに

- q-類似の初等的な結果を Coq で形式化する.
  →具体的にはTaylor展開のq-類似の形式化が目標
- q-類似の定義, 定理, 証明は[Kac]による.
- 形式化には mathcomp を用いている.

q-類似とは

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-微分

以下, qを1でない実数とする.

Definition [Kac] p1 (1.1), p2 (1.5)

関数$f : \mathbb{R} \to \mathbb{R}$に対して, $f(x)$のq-差分$d_q f(x)$とq-微分$D_q f(x)$を以下のように定める.

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

自然数のq-類似

$f(x) = x^n$を定義に沿ってq-微分すると以下の通りである.

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

通常の微分では, $(x^n)' = nx^{n-1}$となることと比較して, 自然数のq-類似を定める.

Definition [Kac] p2 (1.9)

. $n \in \mathbb{N}$について, nのq-類似$[n]$を

$$[n] = \frac{q^n - 1}{q - 1} (= 1 + q + q^2 + \cdots q^{n-1})$$

と定義する.

$(x-a)^n$のq-類似

$D_q (x-a)_q^n = [n](x-a)_q^{n-1}$をみたすように$(x-a)_q^n$の$q$-類似を定める.

Definition [Kac] p8 Definition (3.4)

$x, a \in \mathbb{R}$, $n \in \mathbb{N}$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.

q-Taylor展開

Theorem [Kac] 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,

$$f(x) = \sum_{j=0}^{N} (D_q^j f)(c) \frac{(x-c)_q^j}{[j]!} \left([n]! := \begin{cases} 1 & (n = 0) \\ [n] \times [n-1] \times \cdots \times [1] & (n \geq 1) \end{cases} \right)$$

が成り立つ.

mathcomp の構造

実数として mathcomp の ssrnum で定義されている`rcfType`を用いている.

```
Variables (R : rcfType) (q : R).
```

mathcomp の型には階層構造があり, より一般の型の構造を引き継ぐ.
`rcfType`は特に`ringType`を引き継いでいることが重要.
→用いる補題のほとんどが`ringType`に対するもの

q-微分の形式化

```
Hypothesis Hq : q - 1 ≠ 0.
Notation "f // g" := (fun x ⇒ f x / g x) (at level 40).
Definition dq (f : R → R) x := f (q * x) - f x.
Definition Dq f := dq f // dq id.
```

$[n]$の形式化

```
Definition qnat n : R := (q ^ n - 1) / (q - 1).
Lemma Dq_pow n x : x ≠ 0 →
  Dq (fun x ⇒ x ^ n) x = qnat n * x ^ (n - 1).
Proof.
  move⇒ Hx.
  rewrite /Dq /dq /qnat.
  rewrite -{4}(mul1r x) -mulrBl expfzMl -add_div; last by apply mulf_neq0.
  rewrite [in x ^ n](_ : n = (n -1) +1) //; last by rewrite subrK.
  rewrite expfzDr ?expr1z ?mulrA -?mulNr ?red_frac_r ?add_div //.
  rewrite -{2}[x ^ (n - 1)]mul1r -mulrBl mulrC mulrA.
  by rewrite [in (q - 1)^-1 * (q ^ n - 1)] mulrC.
Qed.
```

$(x-a)_q^n$の形式化

```
Fixpoint qbinom_pos a n x :=
  match n with
  | 0 ⇒ 1
  | n0.+1 ⇒ (qbinom_pos a n0 x) * (x - q ^ n0 * a)
  end.
Theorem Dq_qbinom_pos a n x : x ≠ 0 →
  Dq (qbinom_pos a n.+1) x = qnat n.+1 * qbinom_pos a n x.
```

関数から多項式へ

$x/x = 1$を計算するとき

- 実数 $\cdots$ $x \neq 0$が必要
- 多項式 $\cdots$ xは単項式なので自動的に $x \neq 0$(ゼロ多項式)

→多項式で考えれば約分した後でも$x = 0$での値が求められる.

mathcomp での多項式の構造

一般に環上の多項式全体は環を成し, 加群の構造を持つが, このことも mathcomp で形式化されている.
→`ringType`の補題がそのまま使え, スカラー倍も定義できる.

多項式での再定義

- q-差分

```
Definition scale_var (p : {poly R}):=
 \poly_(i < size p) (q ^ i * p`_i).
Definition dqp p := scale_var p - p.
```

`scale_var` $\cdots$ `(scale_var p).[x] = p.[qx]`

- q-微分

```
Definition Dqp p := dqp p %/ dqp 'X.
```

`Dqp'` $\cdots$ `Dqp`を`'X`で約分した形

- $(x-a)_q^n$

```
Fixpoint qbinom_pos_poly a n :=
  match n with
  | 0 ⇒ 1
  | n0.+1 ⇒ (qbinom_pos_poly a n0) * ('X - (q ^ n0 * a)%:P)
  end.
```

各点での定義との関係

```
Definition ap_op_poly (D : (R → R) → (R → R)) (p : {poly R}) :=
  D (fun (x : R) ⇒ p.[x]).
Notation "D # p" := (ap_op_poly D p) (at level 49).
Lemma dqp_dqE p x : (dqp p).[x] = (dq # p) x.
```

→`p`に`dqp`を適用した多項式の`x`での値と`p` ↦`p.[x]`という関数に`dq`を適用した関数の`x`での値は等しい

```
Lemma Dqp_Dqp'E p : Dqp p = Dqp' p.
Lemma Dqp'_DqE p x : x ≠ 0 → (Dqp' p).[x] = (Dq # p) x.

Lemma qbinom_posE a n x :
  qbinom_pos a n x = (qbinom_pos_poly a n).[x].

Lemma Dqp'_qbinom_poly a n :
  Dqp' (qbinom_pos_poly a n.+1) = (qnat n.+1) *: (qbinom_pos_poly a n).
```

q-Taylor展開の形式化

```
Fixpoint qfact n :=
  match n with
  | 0 ⇒ 1
  | n0.+1 ⇒ qfact n0 * qnat n0.+1
  end.
Theorem q_Taylorp n (f : {poly R}) c :
  (∀ n, qfact n ≠ 0) →
  size f = n.+1 →
  f = \sum_(0 ≤ i < n.+1)
    ((Dqp' \^ i) f).[c] *: (qbinom_pos_poly c i / (qfact i)%:P).
```

今後の展望

現在開発中のライブラリ mathcomp analysis [?] の利用

- $q \to 1$で通常の数学に戻ることの形式化
- 無限和に関する形式化
  →Gauss's binomial formulaの拡張, q-指数関数, q-三角関数

参考文献

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

形式化したコードは github で公開している.
`https://github.com/nakamurakaoru/q-analogue/tree/thesis`
また, 名古屋大学での修士論文も本ポスターの内容についてである.
`https://github.com/nakamurakaoru/masters-thesis/tree/main`