# Connections among *q*-Binomial Theorems

by Alan Mehlenbacher

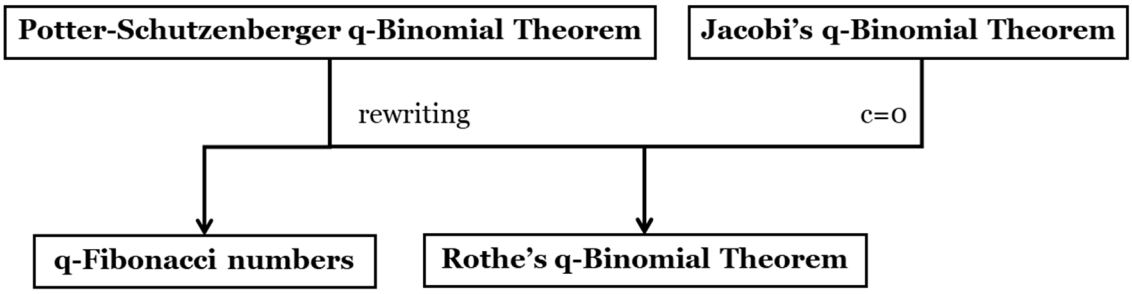
This article explains three well-known *q*-binomial theorems and the connections among them.

The concepts are illustrated using examples from Python programs that use the symbolic programming features of Sympy. For detailed explanations, derivations, and proofs, see Chapter 2 of *An Introduction to q-analysis* by Warren P. Johnson.

## 1 Theorem Overview

The foundation for the *q*-binomial theorems there is a "non-*q*" binomial theorem that deals with non-commutative variables *x* and *y*. For the subsequent *q*-binomial theorems and the *q*-Fibonacci numbers, *q* is introduced as a commutativity mechanism so that $yx = qxy$.

Among the three *q*-binomial theorems and *q*-Fibonacci numbers, there are connections as illustrated in the figure below and described on the following pages.




The three *q*-binomial theorems are listed together in the table below.

**Table 1: *q*-Theorems**

| | |
|---|---|
| Potter-Schutzenberger *q*-binomial theorem | $(x+y)^n = \sum_{k=0}^{n} \binom{n}{k}_q x^k y^{n-k}$ |
| Rothe's *q*-binomial theorem | $(z+a)(zq+a)(zq^2+a)+\ldots+(zq^{n-1}+a) = \sum_{k=0}^{n} \binom{n}{k}_q q^{\binom{k}{2}} x^k a^{n-k}$ |
| Jacobi's *q*-binomial theorem | $(b-a)(b-aq)\ldots(b-aq^{(n-1)}) =$ $\sum_{k=0}^{n} \binom{n}{k}_q (b-c)(b-cq)\ldots(b-aq^{(n-k-1)}) \times (c-a)(c-aq)\ldots(c-aq^{(k-1)})$ |

## 2 Noncommutative Binomial Theorem

The noncommutative binomial theorem demonstrates that $(x+y)^n = \sum_{k=0}^{n} V_{k,n-k}$, where $x$ and $y$ are non-commuting variables and $V_{k,n-k}$ is the sum of all the products of k $x$'s and n-k $y$'s.

For example, using Python code for both sides of the theorem, we have for n = 2:

(x+y)**n: x*y + x**2 + y*x + y**2

SumVq: x*y + x**2 + y*x + y**2

For n = 3, we have:

(x + y)**n: x*y*x + x*y**2 + x**2*y + x**3 + y*x*y + y*x**2 + y**2*x + y**3

SumV: x*y*x + x*y**2 + x**2*y + x**3 + y*x*y + y*x**2 + y**2*x + y**3

These expressions can be simplified to x**3 + 3*x**2*y + 3*x*y**2 + y**3.

## 3 Potter-Schutzenberger *q*-Binomial Theorem

Now we modify the noncommutative binomial theorem by letting $q$ commute with $x$ and $y$ so that $yx = qxy$. This results in the Potter-Schutzenberger $q$-binomial theorem $(x+y)^n = \sum_{k=0}^{n} Vq_{k,n-k}$, where $Vq_{k,n-k}$ is the sum of all the different products of k $x$'s and n-k $y$'s when $yx = qxy$. It can be shown that $Vq_{k,n-k} = \binom{n}{k}_q x^k y^{n-k}$.

For example, using Python code for both sides of the theorem, we have for n = 2:

(x+y)**n: q*x*y + x**2 + x*y + y**2

SumVq : q*x*y + x*y + x**2 + y**2

This is obviously a $q$-analogue of the noncommutative binomial theorem since setting q = 1 results in x*y + x**2 + y*x + y**2.

For n = 3, the Python output is:

(x+y)**n: q**2*(x**2*y + x*y**2) + q*(x**2*y + x*y**2) + x**3 + x**2*y + x*y**2 + y**3

SumVq : q**2*(x*y**2 + x**2*y) + q*(x*y**2 + x**2*y) + x*y**2 + x**2*y + x**3 + y**3

Again, setting q = 1 results x*y**2 + x**2*y + x*y**2 + x**2*y + x*y**2 + x**2*y + x**3 + y**3, which can be rearranged to the noncommutative binomial theorem.

# 4 *q*-Fibonacci Numbers

Since the non-q Fibonacci sequence is $F_{n+1} = F_n + F_{n-1}$, it makes sense to define the *q*-Fibonacci sequence with $F_{n+1}(q) = F_n(q) + q^n F_{n-1}(q)$.

Defining the first two *q*-Fibonacci numbers $F_0(q) = F_1(q) = 1$, then the next four *q*-Fibonacci numbers defined by the *q*-Fibonacci sequence are:

$F_2(q) = F_1(q) + q^1 F_0(q) = 1 + q$,

$F_3(q) = F_2(q) + q^2 F_1(q) = 1 + q + q^2$,

$F_4(q) = F_3(q) + q^3 F_2(q) = 1 + q + q^2 + q^3(1 + q) = 1 + q + q^2 + q^3 + q^4$,

$F_5(q) = F_4(q) + q^4 F_3(q) = 1 + q + q^2 + q^3 + q^4 + q^4(1 + q + q^2) = 1 + q + q^2 + q^3 + 2q^4 + q^5 + q^6$, and so on.

Setting q = 1 results in the regular Fibonacci numbers 1, 1, 2, 3, 5, 8, ….

Using the Potter-Schutzenberger *q*-binomial theorem, it can be shown that $F_n(q) = \sum_{k=0}^{n} \binom{n-k}{k}_q q^{k^2}$.

For the left side of this identity, the Python code uses the recursion to yield the first column of Table 2. For the right side of this identity, the Python code uses the formula derived from the Potter-Schutzenberger *q*-binomial theorem to yield the second column of Table 2. The results demonstrate the validity of the identity.

**Table 2: Demonstration of *q*-Fibonacci Sequences**

| $F_{n+1}(q) = F_n(q) + q^n F_{n-1}(q)$ | $\sum_{k=0}^{n} \binom{n-k}{k}_q q^{k^2}$ |
|---|---|
| F(1): 1 | F(1): 1 |
| F(2): q + 1 | F(2): q + 1 |
| F(3): q**2 + q + 1 | F(3): q**2 + q + 1 |
| F(4): q**4 + q**3 + q**2 + q + 1 | F(4): q**4 + q**3 + q**2 + q + 1 |
| F(5): q**6 + q**5 + 2*q**4 + q**3 + q**2 + q + 1 | F(5): q**6 + q**5 + 2*q**4 + q**3 + q**2 + q + 1 |

# 4 Rothe's *q*-Binomial theorem

Rothe's *q*-binomial theorem states that $(z\ +\ a)(zq\ +\ a)\left(zq^2+a\right)+\ldots+\left(zq^{n-1}+a\right)=\sum_{k=0}^{n}Wq(n,k)$,

where $Wq(n,k)=\begin{pmatrix} n \\ k \end{pmatrix}_q q^{\binom{k}{2}}x^k a^{n-k}$. The proof starts with the Potter-Schutzenberger theorem

replacing *x* with *xy* and *y* with *ay*, still using *yx* = *qxy*, but letting *a* commute with *x, y,* and *q*. Eventually the *y* terms cancel and we are left with variables *x, q,* and *a* that commute.

For example, for n = 3 the Python code demonstrates the theorem:

LHS: a**3 + a**2*q**2*z + a**2*q*z + a**2*z + a*q**3*z**2 + a*q**2*z**2 + a*q*z**2 + q**3*z**3

RHS: a**3 + a**2*q**2*z + a**2*q*z + a**2*z + a*q**3*z**2 + a*q**2*z**2 + a*q*z**2 + q**3*z**3

Setting q = 1 and replacing a with x and z with y results in x**3 + x**2*y + x**2*y + x**2*y + x*y**2 + x*y**2 + x*y**2 + y**3, which can be rearranged to the noncommutative binomial theorem.

# 5 Jacobi's *q*-Binomial Theorem

Jacobi *q*-binomial theorem contains three commuting variables *a*, *b*, and *c* in addition to *q*. Define three products: *Prodba*, *Prodbc*, and *Prodca* as follows:

$Prodba = (b-a)(b-aq)...(b-aq^{(n-1)})$

$Prodbc = (b-c)(b-cq)...(b-aq^{(n-k-1)})$

$Prodca = (c-a)(c-aq)...(c-aq^{(k-1)})$

Then the Jacobi *q*-binomial theorem is $Prodba = \sum_{k=0}^{n} \binom{n}{k}_q Prodbc \times Prodca$.

## 5.1 Validity

Python code demonstrates the validity of the theorem. For example, for n = 3:

LHS: -a*b**2 + b**3 + q**3*(-a**3 + a**2*b) + q**2*(a**2*b - a*b**2) + q*(a**2*b - a*b**2)

RHS: -a*b**2 + b**3 + q**3*(-a**3 + a**2*b) + q**2*(a**2*b - a*b**2) + q*(a**2*b - a*b**2)

Setting q = 1, results in -a**3 + 3*a**2*b - 3*a*b**2 + b**3. Then replacing –a with x, and replacing b with y results in the noncommutative binomial theorem.

## 5.2 Connection to Rothe's *q*-Binomial Theorem

Setting c = 0 and substituting z for b, and a for –a results in a**3*q**3 + a**2*q**3*z + a**2*q**2*z + a**2*q*z + a*q**2*z**2 + a*q*z**2 + a*z**2 + z**3.

Swapping a and z results in z**3*q**3 + z**2*q**3*a + z**2*q**2*a + z**2*q*a + z*q**2*a**2 + z*q*a**2 + z*a**2 + a**3

Table 3 shows the results of rearranging this expression and comparing it term by term with Rothe's *q*-binomial theorem.

**Table 3**

| **Jacobi** | **Rothe** |
|---|---|
| a**3 | a**3 |
| z*q**2*a**2 | a**2*q**2*z |
| z*q*a**2 | a**2*q*z |
| z*a**2 | a**2*z |
| z**2*q**3*a | a*q**3*z**2 |
| z**2*q**2*a | a*q**2*z**2 |
| z**2*q*a | a*q*z**2 |
| z**3*q**3 | q**3*z**3 |