Automated Proof of Divisibility Identities Involving Sums of Exponentials

Tuomas Laakkonen, Simran Chowdhry

September 10, 2017

1 Abstract

We give a general method for proving (or determining the falsehood of) identities of the form

$$\sum_{i}^{n} a_i {b_i}^{c_i x + d_i} \equiv 0 \pmod{p}$$

for $x \geq 1$, by repeated application of mathematical induction, as well as several algorithms to perform this method automatically.

2 Exponential Normal Form

Any function of the form

$$f(x) = \sum_{i}^{n} a_i {b_i}^{c_i x + d_i}$$

can be expressed as a function of the form

$$f(x) = \sum_{i}^{n} a'_i {b'_i}^{x}$$

by the usual index laws, using the transformations

$$a'_i = a_i {b_i}^{d_i}$$

$$b'_i = {b_i}^{c_i}$$

This form is known as *exponential normal form*.

3 Proof Method

Given a function $f(x)$ in exponential normal form, let the $P(x)$ be the proposition that:

$$f(x) \equiv 0 \pmod{p}$$

We prove this identity by mathematical induction. First we verify that $P(1)$ is true. If this is not so, then the identity is false. If $f(x)$ only has one term, then this is enough to prove $P(x)$ for all $x \geq 1$, otherwise we proceed to the induction step:

$$\begin{aligned}
f(x+1) &= \sum_i^n a_i {b_i}^{x+1} \\
&= \sum_i^n a_i b_i b_i^x \\
&= \sum_i^n a_i \left(b_n + b_i - b_n\right) b_i^x \\
&= \sum_i^n a_i b_n b_i^x + \sum_i^n a_i \left(b_i - b_n\right) {b_i}^x \\
&= b_n \left(\sum_i^n a_i {b_i}^x \right) + \sum_i^n a_i \left(b_i - b_n\right) {b_i}^x \\
&= b_n \left(\sum_i^n a_i {b_i}^x \right) + \sum_i^{n-1} a_i \left(b_i - b_n\right) {b_i}^x \\
&= b_n f(x) + \sum_i^{n-1} a_i \left(b_i - b_n\right) {b_i}^x
\end{aligned}$$

Letting $g(x) = \sum_i^{n-1} a_i \left(b_i - b_n\right) {b_i}^x$,

$$f(x+1) = b_n f(x) + g(x)$$

Thus to prove $P(x+1)$ we need only assume $P(x)$ and then prove that:

$$g(x) \equiv 0 \pmod{p}$$

However, $g(x)$ is in exponential normal form, so we can recursively use this method of proof to prove this. Thus we see that $P(1)$ and $P(x) \rightarrow P(x+1)$, giving $P(x)$ for all $x \geq 1$ by induction.

4 Automation

A naïve translation of the above proof method yields a recursive procedure:

```
function PROVE(a, b, p)
    n ← |a|
    if ∑_i^n a_i b_i ≡ 0 (mod p) then
        if n = 1 then
            return Proven
        else
            a′ ← {a_i(b_i − b_n) | 0 ≤ i ≤ n − 1}
            return PROVE(a′, b, d)
        end if
    else
        return False
    end if
end function
```

Because in every recursive call, the size of the argument decreases, termination is ensured. Exploiting this yields an iterative procedure:

```
function PROVE(a, b, p)
    k ← |a|
    while k ≥ 1 do
        if ∑_i^k a_i b_i ≡ 0 (mod p) then
            a ← {a_i(b_i − b_k) | 0 ≤ i ≤ k − 1}
            k ← k − 1
        else
            return False
        end if
    end while
    return Proven
end function
```

This algorithm shows that to prove such an identity amounts to checking that

$$\sum_i^k a_i b_i \prod_j^{n-k} (b_i - b_{n-j}) \equiv 0 \pmod{p}$$

is true for $1 \leq k \leq n$.

5 Conclusions

While this method is valid for a large set of identities, it could be extended in the future to support identities of the form

$$\sum_i^n a_i {b_i}^{f_i(x)} + w \equiv q \pmod{p}$$

for forms of $f_i(x)$ other than linear functions.