Calculation of protein expectation value from peptide expectation values in X! Tandem:

Assume that an experiment has generated \( s \) mass spectra. If a protein sequence is being inferred from the observation of \( n \) unique peptide sequences, each of which has been assigned an expectation value \( e_j \), then the expectation value for the protein, \( E_{\text{pro}} \), is given by:

\[
E_{\text{pro}} = \left( \frac{\beta^n (1 - \beta)^{s-n}}{sN^{n-1}} \right) \times \left( \prod_{j=1}^{n} e_j \right) \times \left( \prod_{i=0}^{n-1} \frac{(s-i)}{(n-i)} \right)
\]

where

\( N = \) peptide sequences scored in to find the \( n \) unique peptides
\( \beta = N/(\text{total number of peptides in the proteome considered}) \)

In the exceptional case that only one peptide has been observed, \( E_{\text{pro}} = e_1 \)

The following page shows the code used by X! Tandem to implement this equation.
/*
 * expect_protein is used to assign the expectation value for a protein, if more than one peptide has been found for that protein. The expectation values for the peptides are combined with a simple Bayesian model for the probability of having two peptides from the same protein having the best score in different spectra.
 */

double mprocess::expect_protein(const unsigned long _c, // number of peptides found
                               const unsigned long _t, // number of total spectra
                               const unsigned long _n, // number of peptides considered
                               const double _d // sum of log peptide expectation values
)
{
    double dValue = _d+log10((double)m_tProteinCount);
    if(_c == 1 && _d < 0.0) {
        return _d;
    }
    else if(_c == 1) {
        return 1.0;
    }
    if(_c == 0) {
        return 1.0;
    }
    double dN = _n;
    double dK = _c;
    double dV = _t;
    unsigned long a = 0;
    while(a < _c) {
        dValue += log10((dV - a)/(dK - a));
        a++;
    }
    dValue -= log10(dV);
    dValue -= (dK-1.0)*log10(dN);
    double dP = dN/(double)m_tPeptideCount;
    if(dP >= 1.0)
        dP = 0.9999999;
    double dLog = dK*log10(dP)+(dV-dK)*log10(1.0-dP);
    dValue += dLog;
    return dValue;
}