Manual for

IBIS

Version 1.0

Paul van Hoegaerden, Coenraad Marinissen, Wouter Waalewijn

March 6, 2025

Quick reference guide for IBIS 1.0

- **Step 0** Install FORM (chapter 2)
- **Step 1 (optional)** Prepare your inverse binomial sum (chapter 4)
- **Step 2** Run IBIS (chapter 3)
- **Step 3 (optional)** Process the output (chapter 6)
- **Step 4 (if necessary)** Commonly encountered problems (chapter 7)

Useful procedures

- Running IBIS `IBIS`
- More than two harmonic sums `summer(i)`
- Unsynchronized harmonic sums `Prepsync` and `summer(i)`
- Wrong sum boundaries `Boundaryfix`
- $n - j$ as the denominator instead of j `Denflip`
- Euler-Zagier sums `ZtoS(Z,S)`
- 0 as an index `summer(i)`

Contents

Chapter 1

Introduction

This document serves as a manual for IBIS, which solves inverse binomial sums in terms of S-sums. It is intended for first-time users, but structured in the hope that repeat users can quickly find the solutions to their problems. We include a section on commonly encountered problems, if you encounter an unusual or particularly interesting one, please get in touch.

Before diving into the manual, we first introduce the notation for harmonic and Euler-Zagier sums as implemented in FORM [1, 2]:

$$\begin{aligned} S_{p_1,\ldots,p_k}(n) &\to \texttt{S(R(p1,...,pk),n)}, \\ Z_{p_1,\ldots,p_k}(n) &\to \texttt{Z(R(p1,...,pk),n)}. \end{aligned} \tag{1.1}$$

For the rest of the notation and an introduction to S- and Z-sums, we refer the reader to the main paper [3]. It is assumed that the definitions and conventions presented there are already familiar to the reader of this manual.

Chapter 2

Initial installation and setup

IBIS uses the FORM symbolic manipulation system, which can be found at `https://www.nikhef.nl/~form/`. No particular optimisations for parallelisation (i.e. PARFORM or TFORM) are implemented in IBIS, but parallelisation may still speed up the calculations somewhat. IBIS itself consists of one file: IBIS.h, which contains the procedures, functions, and definitions that are used for solving the inverse binomial sums. To use IBIS, a user must include this header file at the beginning of their `.frm` file, call IBIS after writing their inverse binomial sum as input, and run it using FORM[1]. This `.frm` file can be named freely, and edited as desired, but IBIS.h should be left untouched.

[1]i.e. for a file called `invbinosum.frm` containing an inverse binomial sum, the command to run IBIS is `form invbinosum.frm`

Chapter 3

How to use IBIS

With IBIS, users can solve inverse binomial sums in terms of S-sums. The sums that can be solved are all of the following form:

$$\sim \sum_{j=1}^{n-1} \frac{1}{\binom{n}{j}} \cdots \tag{3.1}$$

In Tab. 3.1, we specify the additional terms (represented by the dots) that can appear in the summand and classify the inverse binomial sums according to their sign behavior and the number of harmonic sums they contain.

Fixed sign	
0 harmonic sums	$\sum_{j=\{1,-c+1\}}^{n-1} \frac{j!(n-j)!}{n!} \frac{1}{(j+c)^k}$
1 harmonic sum	$\sum_{j=\max\{1,-c+1\}}^{n-1} \frac{j!(n-j)!}{n!} \frac{1}{(j+c)^k} S_{p_1,\ldots,p_s}(n-j)$ and $\sum_{j=\max\{1,-c+1\}}^{n-1} \frac{j!(n-j)!}{n!} \frac{1}{(j+c)^k} S_{q_1,\ldots,q_r}(j)$
2 harmonic sums	$\sum_{j=\max\{1,-c+1\}}^{n-1} \frac{j!(n-j)!}{n!} \frac{1}{(j+c)^k} S_{p_1,\ldots,p_s}(n-j) S_{q_1,\ldots,q_r}(j)$
Sign altering	
0 harmonic sums	$\sum_{j=\max\{1,-c+1\}}^{n-1} \frac{j!(n-j)!}{n!} \frac{(-1)^j}{(j+c)^k}$
1 harmonic sum	$\sum_{j=\max\{1,-c+1\}}^{n-1} \frac{j!(n-j)!}{n!} \frac{(-1)^j}{(j+c)^k} S_{p_1,\ldots,p_s}(n-j)$ and $\sum_{j=\max\{1,-c+1\}}^{n-1} \frac{j!(n-j)!}{n!} \frac{(-1)^j}{(j+c)^k} S_{q_1,\ldots,q_r}(j)$
2 harmonic sums	$\sum_{j=\max\{1,-c+1\}}^{n-1} \frac{j!(n-j)!}{n!} \frac{(-1)^j}{(j+c)^k} S_{p_1,\ldots,p_s}(n-j) S_{q_1,\ldots,q_r}(j)$

Table 3.1: The basis set of inverse binomial sums solvable using IBIS. Using procedures provided by IBIS and SUMMER a larger set can be solved, as discussed in the IBIS manual.

Variable	Set of possible values
c	$\mathbb{Z}$
k	$\mathbb{N}_0$
p_i, q_j $i \in \{1,2,...,s\}$ and $j \in \{1,2,...,r\}$	$\mathbb{Z}_{\neq 0}$
s,r	$\mathbb{N}_0$
n	Symbolic: n Numerical: $\mathbb{N}_{>1}$

Table 3.2: Possible values for the parameters specified in Tab. 3.1 that are supported by IBIS.

The max function returns the largest value among all its input arguments. This ensures that division by zero is avoided in the inverse binomial sum. In Tab. 3.2, we outline the available options for the variables that appear in Tab. 3.1, which the user can modify as needed.

To simplify the remainder of this manual, we define the set of all possible combinations of indices for harmonic sums, denoted as D, as follows:

$$D = \left\{ \{p_1, ..., p_s\} \middle| s \in \mathbb{N}_{>0}, p_i \in \mathbb{Z} \setminus 0 \text{ for } i \in \{1, ..., s\} \right\}. \tag{3.2}$$

From this point forward, we can represent harmonic sums that appear in inverse binomial sums using this notation, such as $S_A(j)$, where $A \in D$.

Now, let us provide a specific example of a `.frm` file that uses IBIS to solve an inverse binomial sum in terms of S-sums. For instance, suppose we want to solve the following inverse binomial sum:

$$\sum_{j=1}^{n-1} \frac{j!(n-j)!}{n!} \frac{(-1)^j}{j^2} S_1(j) S_2(n-j) \tag{3.3}$$

This sum can be solved using the following Form code:

```
#include- IBIS.h

Local F1 = sum(j,1,n-1)*invbino(n,j)*sign(j)
           *den(j)^2*S(R(1),j)*S(R(2),n-j);

#call IBIS

Print;

.end
```

Executing this code yields the result:

```
- 4*S(R(-4,1),X(1,1),n) - 6*S(R(-2,-3),X(1,1),n)
+ 4*S(R(-2,1,-2),X(1,1,1),n) + 2*S(R(-2,1,2),X(1,1,1),n)
+ 2*S(R(-2,2,1),X(1,1,1),n) - 6*S(R(-2,3),X(1,1),n)
+ 4*S(R(1,-4),X(1,1),n) - 4*S(R(1,-2,-2),X(1,1,1),n)
- 2*S(R(1,-2,2),X(1,1,1),n) - 2*S(R(1,2,-2),X(1,1,1),n)
- 2*S(R(1,2,2),X(1,1,1),n) + 6*S(R(1,4),X(1,1),n)
+ 2*S(R(2,-2,1),X(1,1,1),n) - 2*S(R(2,1,-2),X(1,1,1),n)
- S(R(2,1,2),X(1,1,1),n) + 6*S(R(3,-2),X(1,1),n)
+ 3*S(R(3,2),X(1,1),n);                                    (3.4)
```

This serves as a straightforward example of solving an inverse binomial sum in terms of S-sums using IBIS. For more complex examples, we refer the reader to Chapter 5.

Chapter 4

How to prepare your inverse binomial sum as input for IBIS

Users may encounter inverse binomial sums that resemble the forms in Tab. 3.1 but do not match them exactly. In this section, we extend the range of inverse binomial sums that IBIS can solve by outlining sums that deviate from the forms listed in Tab. 3.1 yet can be transformed to conform to these structures. Each subsection focuses on a specific type of deviation from the standard form and includes an example demonstrating how to solve an inverse binomial sum with that particular deviation. In the next chapter, we also present examples that incorporate multiple deviations simultaneously.

4.1 More than 2 harmonic sums

The first deviation from the forms listed in Tab. 3.1 that we will deal with is the presence of more than two harmonic sums in an inverse binomial sum. To specify the type of products we can transform into a solvable form in IBIS, let $A_1, ..., A_m, B_1, ..., B_l \in D$, where $m, l \in \mathbb{N}_{>0}$ and $a_1, ..., a_m, b_1, ..., b_l \in \mathbb{Z}$. The general product of harmonic sums that can be reduced to a form solvable by IBIS is given by:

$$S_{A_1}(j + a_1) \cdots S_{A_m}(j + a_m) \cdot S_{B_1}(n - j + b_1) \cdots S_{B_l}(n - j + b_l)\,. \tag{4.1}$$

The correct summation boundaries for the inverse binomial sum containing this general product of harmonic sums are given by:

- Upper limit: $\min\{n - 1, n + b_{\min} - 1\}$,
- Lower limit: $\max\{1, -a_{\min} + 1\}$,

where $a_{\min} = \min(a_1, ..., a_m)$ and $b_{\min} = \min(b_1, ..., b_l)$. These boundaries ensure that none of the harmonic sums within the inverse binomial sum evaluate to zero. The product of harmonic sums in Eq. (4.1) can be rewritten as a sum of products containing at most two harmonic sums using the following recursion relation:

$$\begin{aligned} S_{p_1,\dots,p_k}(n)\cdot S_{p'_1,\dots,p'_l}(n) = &\sum_{i=1}^{n}\frac{\text{sign}(p_1)^i}{i^{|p_1|}}S_{p_2,\dots,p_k}(i)S_{p'_1,\dots,p'_l}(i)\\ &+\sum_{i=1}^{n}\frac{\text{sign}(p'_1)^i}{i^{|p'_1|}}S_{p_1,\dots,p_k}(i)S_{p'_2,\dots,p'_l}(i)\\ &-\sum_{i=1}^{n}\frac{\text{sign}(p_1\cdot p'_1)^i}{i^{|p_1|+|p'_1|}}S_{p_2,\dots,p_k}(i)S_{p'_2,\dots,p'_l}(i)\,. \end{aligned} \tag{4.2}$$

In this recursion, the terms on the right-hand side involve products of harmonic sums with fewer indices than the product on the left-hand side. Consequently, the recursion will ultimately terminate successfully, resulting in a sum consisting solely of individual harmonic sums.

This recursion has been implemented into SUMMER [4] under the name `basis`. For this algorithm to work, the harmonic sums must share the same argument. To synchronize the arguments, SUMMER uses the `synchss` procedure, which adjusts all arguments to the largest value among them using the method described below in Eq. (4.12). For a more detailed explanation on the synchronization of arguments, we refer the reader to the main paper [3]. Using SUMMER, the general product in Eq. (4.1) can be transformed into a sum of terms in the following simplified form:

$$C\cdot S_{A_{\max}}(j+a_{\max}) * S_{B_{\max}}(n-j+b_{\max})\,, \tag{4.3}$$

where $a_{\max}=\max(a_1,\dots,a_m)$, $b_{\min}=\max(b_1,\dots,b_l)$ and C is a prefactor.

The prefactor C may include a combination of either a `delta` or `deltap` function (For a detailed explanation of why these functions appear, see [4]) together with a term of the form $\frac{1}{(j+a)^x}$ or $\frac{1}{(n-j+b)^{x'}}$, where $x, x'\in\mathbb{N}_0$, a an integer between the minimum and maximum value of $\{a_1,\dots,a_m\}$ and b an integer between the minimum and maximum of $\{b_1,\dots,b_l\}$. If the user assigns the correct limits that we specified to the inverse binomial sum, the `deltap` function can be set to 1 and the `delta` function can be set to 0. Although a detailed understanding of these settings is not necessary for using IBIS, an in-depth explanation is provided in the appendix for interested readers.

The `densplit` procedure within SUMMER ensures that each term contains only a single denominator involving j. It works by repeatedly applying the relation:

$$\begin{aligned} \frac{1}{(i+x)^m(i+y)^k} = &\sum_{a=1}^{m}\binom{m+k-1-a}{k-1}\frac{(-1)^{m-a}}{(y-x)^{m+k-a}(i+x)^a}\\ &+\sum_{a=1}^{k}\binom{m+k-1-a}{m-1}\frac{(-1)^{m}}{(y-x)^{m+k-a}(i+y)^a}\,, \end{aligned} \tag{4.4}$$

This ensures that C only contains terms with a single denominator.

To allow IBIS to solve an inverse binomial sum, the arguments of the harmonic sums within the sum must first be synchronized. Whether this can be done depends on the values of $a_{\max}$ and $b_{\max}$. In many cases the arguments of the harmonic sums need to be synchronized before IBIS can process the inverse binomial sum. The next subsection delves into the details of synchronizing the arguments of harmonic sums. Additionally, the following chapter includes an example demonstrating how to handle an inverse binomial sum containing a product of harmonic sums in the form of Eq. (4.1) where the arguments of the harmonic sums have to be synchronized before they can be substituted back into the inverse binomial sum.

There is only one case in which the sum of the products in Eq. (4.3) can directly be substituted back into the inverse binomial sum for IBIS to solve. This occurs when $a_{\max} = b_{\max} = 0$. In this scenario, all products of harmonic sums take the following form:

$$C \cdot S_{A_{\max}}(j) * S_{B_{\max}}(n-j)\,, \tag{4.5}$$

which allows the sum of products to be directly substituted back into the inverse binomial sum and solved in terms of S-sums using IBIS. To illustrate, let us consider solving the following inverse binomial sum:

$$\sum_{j=1}^{n-1} \frac{j!(n-j)!}{n!} \frac{(-1)^j}{j^2} S_{1,2}(j) S_2(j) S_1(n-j)\,. \tag{4.6}$$

We begin by using the following FORM code to simplify the product of harmonic sums:

```
#include- summer.h

Local F1 = den(j1)^2*S(R(1,2),j1)
              *S(R(2),j1)*S(R(1),n-j1);

#call summer(1)

Print;

.end
```

In this example, we use the `summer(i)` procedure from the SUMMER package, which serves as its primary summation routine. This procedure systematically evaluates all sums in the expression, starting with the highest-level sum indicated by the parameter `i`. During its execution, the routine performs several key tasks: splitting fractions, synchronizing the arguments of harmonic sums, and applying the algebraic rules governing these sums. It is important to note that the individual components of this routine cannot be called separately, so the user must carefully prepare the input to ensure correctness. For instance, adding an unnecessary term like `sum(j1,1,n)` in front of the expression would cause `summer(i)` to evaluate and rewrite the whole expression in terms of harmonic sums, which is not the intended use of the procedure here. To avoid this, make sure your expression includes only the necessary components before calling the `summer(i)` procedure. In our example, we called `summer(1)` after renaming the variable j to $j1$. While it is possible to call `summer(i)` with values of $i > 1$, users must be cautious in such cases to ensure that no irrelevant terms are present in the expression, as the procedure will also evaluate expressions for j_i where $i > 1$. If the variable is not renamed, running `summer(1)` will have no effect.

The output generated by `summer(1)` is as follows:

```
2*den(j1)^2*S(R(1),-j1+n)*S(R(1,2,2),j1)
- den(j1)^2*S(R(1),-j1+n)*S(R(1,4),j1)
+ den(j1)^2*S(R(1),-j1+n)*S(R(2,1,2),j1)
- den(j1)^2*S(R(1),-j1+n)*S(R(3,2),j1);                    (4.7)
```

Next, we revert the variable back to j and substitute this result into the inverse binomial sum. The following FORM code demonstrates how to proceed with IBIS:

```
#include- IBIS.h

L F1 = sum(j,1,n-1)*invbino(n,j)*sign(j)*
       (2*den(j)^2*S(R(1),-j+n)*S(R(1,2,2),j)
       - den(j)^2*S(R(1),-j+n)*S(R(1,4),j)
       + den(j)^2*S(R(1),-j+n)*S(R(2,1,2),j)
       - den(j)^2*S(R(1),-j+n)*S(R(3,2),j));

#call IBIS;

Print;

.end
```

By running this code, the inverse binomial sum is solved in terms of S-sums.

4.2 Unsynchronized harmonic sums

Unsynchronized harmonic sums refer to harmonic sums where the arguments differ from the standard form specified in Tab. 3.1. Specifically, they take the form $S_{p_1,\ldots,p_s}(n-j+a)$ or $S_{p_1,\ldots,p_s}(j+a)$ with $a \in \mathbb{Z} \setminus \{0\}$. Such harmonic sums can be transformed to a synchronized form that allows the inverse binomial sums to be solved using IBIS. Two cases need to be considered based on the value of a:

- $a > 0$
- $a < 0$

4.2.1 Case 1: $a > 0$

To synchronize harmonic sums with $a > 0$, we use the following identity:

$$S_{p_1,\ldots,p_s}(j+x) = S_{p_1,\ldots,p_s}(j) + \sum_{i=1}^{x} \frac{\text{sign}(p_1)^{j+i}}{(j+i)^{|p_1|}} S_{p_2,\ldots,p_s}(j+i)\,, \tag{4.8}$$

where $x \in \mathbb{N}_{>0}$. Note that the harmonic sums appearing in the sum from $i = 1$ to x on the right-hand side have one fewer index than the harmonic sum on the left-hand side. By iteratively applying this property and using:

$$S(n) = \begin{cases} 1, & \text{if } n > 0 \\ 0, & \text{if } n \leq 0 \end{cases}, \tag{4.9}$$

we can rewrite all unsynchronized harmonic sums with $a > 0$ into a synchronized form. We have implemented this recursion in the procedure `Prepsync`. However, the synchronized harmonic sums produced by the `Prepsync` procedure are not immediately suitable for substitution back into the inverse binomial sum to be solved by IBIS. These sums are often accompanied by multiple denominators involving the variable j, rather than a single denominator. Before substituting them back into the inverse binomial sum, we thus use the `densplit` procedure from SUMMER to split the fractions. The resulting expression can then be solved with IBIS.

As an example of using the `Prepsync` procedure to solve an inverse binomial sum containing harmonic sums with $a > 0$, consider the following inverse binomial sum that we want to solve:

$$\sum_{j=1}^{n-1} \frac{j!(n-j)!}{n!} \frac{(-1)^j}{j^2} S_2(j+2)\,. \tag{4.10}$$

To solve this, we first make the change $j \to j1$ and then use the following FORM code:

```
#include- IBIS.h

#include- summer.h

L F1 = den(j1)^2*S(R(2),j1+2);

#call Prepsync

#call summer(1)

Print;

.end
```

For this example, it is again essential that the FORM code does not include any unnecessary or unrelated terms, such as `sum(j1,1,n)`. Our focus here is solely on utilizing the `densplit` feature of the `summer(i)` routine, which is designed to handle fraction splitting. Running the FORM code yields:

```
2*den(1+j1) + den(1+j1)^2 + 1/4*den(2+j1) + 1/4*den(2+j1)^2
- 9/4*den(j1) + 5/4*den(j1)^2 + den(j1)^2*S(R(2),j1);          (4.11)
```

Substituting this result back into the inverse binomial sum after changing back $j1 \to j$, we can solve it using IBIS with the following code:

```
#include- IBIS.h

L F1 = sum(j,1,n-1)*invbino(n,j)*sign(j)*(2*den(1+j)
       + den(1+j)^2 + 1/4*den(2+j) + 1/4*den(2+j)^2
       - 9/4*den(j) + 5/4*den(j)^2
       + den(j)^2*S(R(2),j));

#call IBIS

Print;

.end
```

4.2.2 Case 2: $a < 0$

To handle unsynchronized harmonic sums with $a < 0$, we use the following property of harmonic sums:

$$S_{p_1,\ldots,p_s}(j-1) = S_{p_1,\ldots,p_s}(j) - \frac{\text{sign}(p_1)^j}{j^{|p_1|}} S_{p_2,\ldots,p_s}(j)\,. \tag{4.12}$$

By iteratively applying this identity, any harmonic sum with $a < 0$ can be rewritten in terms of harmonic sums with $a = 0$. While this identity is also implemented in the `synchss` procedure in SUMMER, SUMMER uses it specifically to synchronize the arguments of products of harmonic sums. In contrast, we use it solely to align the arguments of individual harmonic sums to $a = 0$. As a result, we implemented this property differently in the `Prepsync` procedure. Just like in the case where $a > 0$, the result after applying `Prepsync` to the unsynchronized harmonic sums is synchronized harmonic sums multiplied by multiple denominators containing the variable j. Before substituting these expressions back into the inverse binomial sum, we must again use the `densplit` procedure in SUMMER to handle the denominators. To illustrate this process, let us solve the following inverse binomial sum:

$$\sum_{j=3}^{n-1} \frac{j!(n-j)!}{n!} \frac{1}{j^2} S_2(j-2)\,. \tag{4.13}$$

The lower limit here is 3 instead of 1 because $S_{p_1,\ldots,p_s}(n) = 0$ if $n \leq 0$. To synchronize the harmonic sum appearing in this inverse binomial sum, we apply the following FORM code:

```
#include- IBIS.h

#include- summer.h

L F1 = den(j1)^2*S(R(2),j1-2);

#call Prepsync

```

```
#call summer(1)

Print;

.end
```

Here, our goal is to use only the `densplit` feature, so we deliberately exclude any parts of the sum that could interfere with the execution of `summer(i)`. Additionally, in this code, we once again make the substitution $j \to j1$ to ensure proper functionality. Running this FORM code produces the following output:

```
2*den(-1+j1) - den(-1+j1)^2 - 2*den(j1) - den(j1)^2
- den(j1)^4 + den(j1)^2*S(R(2),j1);                        (4.14)
```

Substituting this back into the inverse binomial sum, we are left with the following:

$$\sum_{j=3}^{n-1} \frac{j!(n-j)!}{n!} \left(\frac{2}{j-1} - \frac{1}{(j-1)^2} - \frac{2}{j} - \frac{1}{j^2} - \frac{1}{j^4} + \frac{S_2(j)}{j^2} \right). \tag{4.15}$$

For the first two denominators involving j, the lower limit of the sum must be 2, whereas for the remaining denominators containing j, the lower limit should be 1. Since the lower limit of the sum in Eq. (4.15) is 3, the expression must first be reformulated before solving it with IBIS. A detailed explanation of this process is provided in the next section.

4.3 Wrong sum boundaries

As mentioned in the example at the end of Sec. 4.2, users may encounter inverse binomial sums where the upper limit is not given by $n-1$ or the lower limit is not given by $\max(1, -c+1)$. To address this issue, IBIS includes a procedure called `Boundaryfix`, designed to rewrite such sums into a form with proper boundaries. To use this procedure, one has to make sure that all the harmonic sums appearing in the inverse binomial sum are synchronized. Therefore this procedure is typically used after synchronizing the harmonic sums in the inverse binomial sum with the `Prepsync` procedure. The `Boundaryfix` procedure works by decomposing the inverse binomial sum into a sum with the correct summation boundaries and additional corrective terms. For example, consider the following inverse binomial sum:

$$\sum_{j=3}^{n-3} \frac{j!(n-j)!}{n!} \frac{S_2(j)}{j^2} \tag{4.16}$$

By applying the following FORM code:

```
#include- IBIS.h

L F1 = sum(j,3,n-3)*invbino(n,j)*den(j)^2*S(R(2),j);

```

```
#call Boundaryfix

Print;

.end
```

The output of this code is:

```
den(j)^2*S(R(2),j)*sum(j,1,-1+n)*invbino(n,j)
- S(R(2),1)*invbino(n,1) - 1/4*S(R(2),2)*invbino(n,2)
- sum(j,-2+n,-1+n,den(j)^2*S(R(2),j)*invbino(n,j));
```
(4.17)

This output corresponds to the following expression:

$$\sum_{j=3}^{n-3} \frac{j!(n-j)!}{n!} \frac{S_2(j)}{j^2} = \sum_{j=1}^{n-1} \frac{j!(n-j)!}{n!} \frac{S_2(j)}{j^2} - S_2(1) \frac{1!(n-1)!}{n!} - \frac{1}{4} S_2(2) \frac{2!(n-2)!}{n!} - \sum_{j=n-2}^{n-1} \frac{j!(n-j)!}{n!} \frac{S_2(j)}{j^2} . \tag{4.18}$$

As shown, the first term now has proper boundaries and can be directly solved using IBIS by running:

```
#include- IBIS.h

L F1 = sum(j,1,n-1)*invbino(n,j)*den(j)^2*S(R(2),j);

#call IBIS

Print;

.end
```

Keep in mind the following constraints when working with inverse binomial sums:

- Upper limit constraints:
  - The binomial coefficient $\binom{n}{j}$ is undefined for $j > n$. Therefore, the upper limit of the inverse binomial sum should not exceed n.
  - If the inverse binomial sum contains a harmonic sum of the form $S_{p_1,\ldots,p_s}(n-j)$, this harmonic sum will be equal to zero for $j \geq n$. Therefore, defining the upper summation variable for these inverse binomial sums to be greater than $n-1$ is not appropriate.
- Lower limit constraints:
  - The binomial coefficient $\binom{n}{j}$ is also undefined for $j < 0$. Therefore, the lower limit of the inverse binomial sum cannot be less than 0.

– If the initial exponent of the inverse binomial sum is non-zero or if a harmonic sum of the form $S_{p_1,\ldots,p_s}(j)$ appears, the lower limit must be at least 1. This ensures that division by zero is avoided and also accounts for the fact that $S_{p_1,\ldots,p_s}(n) = 0$ if $n \leq 0$.

To assist users, these constraints are enforced through checks within the `Boundaryfix` procedure. If a violation occurs, the FORM program is terminated, and an error message is displayed, explaining the issue.

The `Boundaryfix` procedure does have a limitation. It does not handle terms like $\frac{1}{(n-j+a)^x}$, where $a \in \mathbb{Z}$ and $x \in \mathbb{N}$. Before using the procedure, users must first transform the sum by substituting $j \to n - j$. Detailed instructions for performing this transformation can be found in the next section.

4.4 $n - j$ as the denominator instead of j

When synchronizing harmonic sums of the form $S_{p_1,\ldots,p_s}(n-j+a)$ with $a \in \mathbb{N}$ using Eq. (4.8), terms like $\frac{1}{(n-j+a)^x}$, where $x \in \mathbb{N}$, may appear. Inverse binomial sums containing these kind of terms cannot be directly solved by IBIS. However, by performing a simple change of the summation variable $j \to n - j$, the inverse binomial sum can be transformed into a solvable form.

To facilitate this transformation, we implemented the procedure `Denflip`. Below, we illustrate its usage with an example. Suppose we aim to solve the following inverse binomial sum:

$$\sum_{j=1}^{n-1} \frac{j!(n-j)!}{n!} \frac{1}{(n-j)^2} S_{2,3}(n-j) S_1(j) \tag{4.19}$$

To prepare this inverse binomial sum for IBIS, we apply the following FORM code:

```
#include- IBIS.h

L F1 = sum(j,1,n-1)*invbino(n,j)*den(n-j)^2
         *S(R(2,3),n-j)*S(R(1),j);

#call Denflip

Print;

.end
```

By applying the code, we get:

```
den(j)^2*S(R(1),-j+n)*S(R(2,3),j)*sum(j,1,-1+n)*invbino(n,j);        (4.20)
```

The inverse binomial sum is now in a form that IBIS can solve.

This transformation is effective because the binomial coefficient $\binom{n}{j}$ remains invariant under the substitution $j \to n - j$, ensuring that the structure of the inverse binomial sum is preserved during the transformation. Similar to the `Boundaryfix` procedure, it is expected that the harmonic sums within the inverse binomial sum have already been synchronized prior to applying this method. Additionally, users must ensure that the upper limit of the inverse binomial sum does not exceed $n-1$. Just like in the `Boundaryfix` procedure, we have included a built-in validation check to enforce this condition. If the upper limit exceeds $n-1$, the FORM program will terminate and display an error message indicating the issue. This ensures that the procedure is applied only under appropriate conditions.

4.5 Euler-Zagier sums

Another deviation from the forms listed in Tab. 3.1 is the presence of Euler-Zagier sums within the inverse binomial sum. To rewrite these inverse binomial sums into a form solvable by IBIS, the `ZtoS` procedure, implemented in SUMMER, can be used. This procedure iteratively applies the following recursion until all Euler-Zagier sums are expressed in terms of harmonic sums:

$$Z_{p_1,\ldots,p_k}(n) = \sum_{i_1=1}^{n} \frac{\mathrm{sign}(p_1)^{i_1}}{i_1^{|p_1|}} \sum_{i_2=1}^{i_1} \frac{\mathrm{sign}(p_2)^{i_2}}{i_2^{|p_2|}} Z_{p_3,\ldots,p_k}(i_2-1) - Z_{\mathrm{sign}(p_1\cdot p_2)(|p_1|+|p_2|),p_3,\ldots,p_k}(n)\,. \tag{4.21}$$

Let us now provide an example on how to use this procedure. Suppose we want to solve the following inverse binomial sum:

$$\sum_{j=1}^{n-1} \frac{j!(n-j)!}{n!} \frac{1}{j^2} Z_{1,3,2,1,4}(n)\,, \tag{4.22}$$

we then apply the following FORM code to convert the Euler-Zagier sum into harmonic sums:

```
#include- summer.h

Local F1 = Z(R(1,3,2,1,4),n);

#call ZtoS(Z,S)

.sort
Print;

.end
```

This produces:

```
S(R(1,3,2,1,4),n) - S(R(1,3,2,5),n) - S(R(1,3,3,4),n) + S(R(1,3,7),n)
- S(R(1,5,1,4),n) + S(R(1,5,5),n) + S(R(1,6,4),n) - S(R(1,10),n)
- S(R(4,2,1,4),n) + S(R(4,2,5),n) + S(R(4,3,4),n) -S(R(4,7),n)
+ S(R(6,1,4),n) - S(R(6,5),n) - S(R(7,4),n) + S(R(11),n);        (4.23)
```

After substituting Eq. (4.23) into the original inverse binomial sum, we can solve Eq. (4.22) using IBIS. The following FORM code demonstrates this:

```
#include- IBIS.h

Local F1 = sum(j,1,n-1)*invbino(n,j)*den(j)^2*
           (S(R(1,3,2,1,4),n)-S(R(1,3,2,5),n)
           -S(R(1,3,3,4),n)+S(R(1,3,7),n)
           -S(R(1,5,1,4),n)+S(R(1,5,5),n)
           +S(R(1,6,4),n)-S(R(1,10),n)-S(R(4,2,1,4),n)
           +S(R(4,2,5),n)+S(R(4,3,4),n)-S(R(4,7),n)
           +S(R(6,1,4),n)-S(R(6,5),n)
           -S(R(7,4),n)+S(R(11),n));

#call IBIS

Print;

.end
```

By following this process, we are thus able to solve inverse binomial sums containing Euler-Zagier sums in terms of S-sums using IBIS.

4.6 0 as an index

In our paper, we explored methods for deriving a recursion that reduces the complexity of the inverse binomial sum by either lowering the initial exponent (see the paper for its definition) or reducing the number of indices. However, when attempting to construct a recursion in the special case where the initial exponent of the inverse binomial sum is zero, we encountered certain challenges. Since zero is the lowest possible initial exponent, there are no terms with the same number of indices and lower complexity. As a result, the recursion is restricted to contain only elements with fewer indices. This additional restriction made it impossible to synchronize all elements in the recursion. To address this issue, we allow the initial exponent to be equal to 0, but we prohibit 0 as an index for the harmonic sums. This adjustment enabled us to sufficiently synchronize all the elements in the recursion.
Prohibiting 0 as an index for the harmonic sums should not pose a problem when solving the inverse binomial sums. Specifically, if an inverse binomial sum with 0 as an index in an harmonic sum arises, we can rewrite the harmonic sum as a sum of other harmonic sums without 0 as an index. To achieve this we use SUMMER. SUMMER has a built-in feature capable of rewriting harmonic sums containing an index of 0 into harmonic sums without 0 as an index. While the exact implementation details of SUMMER's approach to rewriting harmonic sums with zero indices are not explicitly documented, the method appears to rely

on transformations along the following lines. It may repeatedly apply an identity such as:

$$\begin{aligned} S_{0,p_2,\dots,p_s}(n) &= \sum_{i=1}^{n}\sum_{j=1}^{i}\frac{1}{j^{p_2}}S_{p_3,\dots,p_s}(j) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n}\frac{1}{j^{p_2}}S_{p_3,\dots,p_s}(j) - \sum_{i=1}^{n}\sum_{j=i+1}^{n}\frac{1}{j^{p_2}}S_{p_3,\dots,p_s}(j) \\ &= n\sum_{j=1}^{n}\frac{1}{j^{p_2}}S_{p_3,\dots,p_s}(j) - \sum_{j=2}^{n}\sum_{i=1}^{j-1}\frac{1}{j^{p_2}}S_{p_3,\dots,p_s}(j) \\ &= (n+1)S_{p_2,\dots,p_s}(n) - S_{(p_2-1),p_3,\dots,p_s}(n)\,. \end{aligned} \tag{4.24}$$

This example assumes $p_2 \geq 0$ for simplicity, but similar transformations work for $p_2 < 0$. In addition, SUMMER likely uses identities for nested sums, such as:

$$\begin{aligned} \sum_{i=1}^{n} i \sum_{j=1}^{i}\frac{1}{j^k}S_{p_1\dots p_s}(j) &= \sum_{j=1}^{n}\frac{1}{j^k}S_{p_1\dots p_s}(j)\sum_{i=j}^{n} i \\ &= \sum_{j=1}^{n}\frac{1}{j^k}S_{p_1\dots p_s}(j)\left(\frac{n(n+1)}{2} - \frac{j(j-1)}{2}\right), \end{aligned} \tag{4.25}$$

which help SUMMER work around using 0 as an index.

To see how this feature works, consider the following inverse binomial sum:

$$\sum_{j=1}^{n-1}\frac{j!(n-j)!}{n!}\frac{1}{j^2}S_{1,2,0,0,1}(j) \tag{4.26}$$

The harmonic sum $S_{1,2,0,0,1}(j)$ must first be rewritten in terms of harmonic sums without zero indices. This can be achieved by expanding the harmonic sum and applying the following FORM code:

```
#include- summer.h

Local F1 = sum1(j1,1,j)*sum2(j2,1,j1)*sum3(j3,1,j2)*
             sum4(j4,1,j3)*sum5(j5,1,j4)*den(j1)*
             den(j2)^2*den(j5);
#call summer(5)

Print;

.end
```

Here, we call `summer(5)` because the harmonic sum contains five indices. Running this FORM code yields the following result:

```
- 5/4*theta(-1+j)*j - 3/4*theta(-1+j)*S(R(1,1),j)
+ 3/2*theta(-1+j)*S(R(1,1,1),j) + theta(-1+j)*S(R(1,2,1),j)
- 1/2*theta(j)*j + 1/2*theta(j)*S(R(1),j) + 1/2*theta(j)*S(R(1),j)*j; (4.27)
```

where `theta` represents the Heaviside step function. In the context of inverse binomial sums with correct summation boundaries (see Sec. 4.3), j will always be greater than 1. Therefore, the `theta` function can be set to 1 using the `id theta(?a) = 1;` statement. After simplifying, the rewritten sum can be solved using the following FORM code:

```
#include- IBIS.h

L F1 = sum(j,1,n-1)*invbino(n,j)*(- 7/4*den(j)
       + 1/2*S(R(1),j)*den(j)^2 + 1/2*S(R(1),j)*den(j)
       - 3/4*S(R(1,1),j)*den(j)^2
       + 3/2*S(R(1,1,1),j)*den(j)^2
       + S(R(1,2,1),j)*den(j)^2);

#call IBIS;

Print;

.end
```

As one can see, we have to be cautious of the powers of j appearing when rewriting the harmonic sum. If the power of j generated by the `summer(i)` procedure exceeds the initial exponent of $\frac{1}{j}$ appearing in the inverse binomial sum, the sum is no longer solvable by IBIS. This restriction arises because the initial exponent of an inverse binomial sum must be greater than or equal to zero for IBIS to solve it. For instance, in our example, the initial inverse binomial sum containing the harmonic sum $S_{1,2,0,0,1}(j)$ must have had an initial exponent of at least 1 to be solvable by IBIS. This requirement is due to the fact that the first, fifth, and seventh term in Eq. (4.27) contain j raised to the power of 1.

Chapter 5

Examples

In this chapter, we will handle some more interesting inverse binomial sums that have several deviations from the standard forms in Tab. 3.1.

5.1 Example 1: More than two unsynchronized harmonic sums

The first inverse binomial sum that we will take a look at is the following:

$$\sum_{j=1}^{n-2} \frac{j!(n-j)!}{n!} \frac{(-1)^j}{j^2} S_1(n-j) S_2(n-j+1) S_1(j) \tag{5.1}$$

This inverse binomial sum contains more than two harmonic sums and $S_2(n-j+1)$ has an unsynchronized argument. Before we can solve this inverse binomial sum with IBIS we thus have to do some preparation using the procedures explained in Chap. 4. The following FORM code is applied:

```
#include- IBIS.h

#include- summer.h

L F1 = den(j1)^2*S(R(1),n-j1)*S(R(2),n-j1+1)*S(R(1),j1);

#call summer(1)
#call Prepsync
#call summer(1)

id deltap(a?) = 1;
id delta(a?) = 0;

Print;

```

```
.end
```

Note that we have set all the deltap functions to 1 and the delta functions equal to 0, just like we mentioned in Sec. 4.1. The above code outputs the following expression:

```
den(-1-n)^2*den(-1+j1-n)^2*S(R(1),j1)*S(R(1),-j1+n)
- 2*den(-1-n)^3*den(-1+j1-n)^2*S(R(1),j1)
+ 2*den( - 1 - n)^3*den(-1+j1-n)*S(R(1),j1)*S(R(1),-j1+n)
- 3*den(-1-n)^4*den(-1+j1-n)*S(R(1),j1)
- 2*den(-1-n)^3*den(1+n)*den(-1+j1-n)*S(R(1),j1)
- den(-1-n)^2*den(1+n)*den(-1+j1-n)^2*S(R(1),j1)
+ den(-1-n)^2*den(1+n)^2*den(-1+j1-n)*S(R(1),j1)
+ den(-1-n)*den(1+n)^2*den(-1+j1-n)^2*S(R(1),j1)
+ 2*den(j1)*den(1+n)^3*S(R(1),j1)*S(R(1),-j1+n)
+ den(j1)^2*den(1+n)^2*S(R(1),j1)*S(R(1),-j1+n)
+ den(j1)^2*S(R(1),j1)*S(R(1,2),-j1+n)
+ den(j1)^2*S(R(1),j1)*S(R(2,1),-j1+n)
- den(j1)^2*S(R(1),j1)*S(R(3),-j1+n);                    (5.2)
```

As one can see, the expression contains terms like $\frac{1}{(n-j+a)^x}$ and not every term satisfies the boundary requirements for IBIS. Therefore, the `Denflip` and `Boundaryfix` procedures must be applied. By changing $j1 \to j$ and running the following FORM code, we are able to solve the inverse binomial sum:

```
#include- IBIS.h

L F1= sum(j,1,n-2)*invbino(n,j)*sign(j)*
      (den(-1-n)^2*den(-1+j-n)^2*S(R(1),j)*S(R(1),-j+n)
      - 2*den(-1-n)^3*den(-1+j-n)^2*S(R(1),j)
      + 2*den(-1-n)^3*den(-1+j-n)*S(R(1),j)*S(R(1),-j+n)
      - 3*den(-1-n)^4*den(-1+j-n)*S(R(1),j)
      - 2*den(-1-n)^3*den(1+n)*den(-1+j-n)*S(R(1),j)
      - den(-1-n)^2*den(1+n)*den(-1+j-n)^2*S(R(1),j)
      + den(-1-n)^2*den(1+n)^2*den(-1+j-n)*S(R(1),j)
      + den(-1-n)*den(1+n)^2*den(-1+j-n)^2*S(R(1),j)
      + 2*den(j)*den(1+n)^3*S(R(1),j)*S(R(1),-j+n)
      + den(j)^2*den(1+n)^2*S(R(1),j)*S(R(1),-j+n)
      + den(j)^2*S(R(1),j)*S(R(1,2),-j+n)
      + den(j)^2*S(R(1),j)*S(R(2,1),-j+n)
      - den(j)^2*S(R(1),j)*S(R(3),-j+n));

#call Denflip
#call Boundaryfix

```

```
#call IBIS

Print;

.end
```

5.2 Example 2: Unsynchronized Euler-Zagier sum

Next, consider the following inverse binomial sum:

$$\sum_{j=4}^{n-1} \frac{j!(n-j)!}{n!} \frac{1}{j^2} Z_{2,3}(j-3) \tag{5.3}$$

This inverse binomial sum includes an Euler-Zagier sum with an unsynchronized argument. Additionally, the lower boundary of the sum starts at $j = 4$. Preparation involves the following FORM code:

```
#include- IBIS.h

#include- summer.h

Local F1 = den(j1)^2*Z(R(2,3),j1-3);

#call ZtoS(Z,S)
#call Prepsync
#call summer(1)

Print;

.end
```

This code produces:

```
- den(-2+j1) + 5/32*den(-2+j1)^2 + 3/16*den(-2+j1)^3
- 1/4*den(-2+j1)^4 + 1/4*den(-2+j1)^5
- 1/4*den(-2+j1)^2*S(R(3),j1) + 1/4*den(-2+j1)*S(R(3),j1)
+ 2*den(-1+j1) - 3*den(-1+j1)^2 + 4*den(-1+j1)^3
- 2*den(-1+j1)^4 + den(-1+j1)^5 - den(-1+j1)^2*S(R(3),j1)
+ 2*den(-1+j1)*S(R(3),j1) - den(j1) + 91/32*den(j1)^2
+ 51/16*den(j1)^3 + 9/4*den(j1)^4 + 5/4*den(j1)^5
+ den(j1)^7 - den(j1)^4*S(R(3),j1) + den(j1)^2*S(R(2,3),j1)
- 5/4*den(j1)^2*S(R(3),j1) - den(j1)^2*S(R(5),j1)
- 9/4*den(j1)*S(R(3),j1);
```
(5.4)

As our original inverse binomial sum starts at $j = 4$, we do need to modify the sum boundaries using `Boundaryfix`. By again changing $j1 \to j$ and then applying the following FORM code, we are able to solve the inverse binomial sum:

```
#include- IBIS.h

L F1 = sum(j,4,n-1)*invbino(n,j)*(- den(-2+j)
       + 5/32*den(-2+j)^2 + 3/16*den(-2+j)^3
       - 1/4*den(-2+j)^4 + 1/4*den(-2+j)^5
       - 1/4*den(-2+j)^2*S(R(3),j)
       + 1/4*den(-2+j)*S(R(3),j) + 2*den(-1+j)
       - 3*den(-1+j)^2 + 4*den(-1+j)^3 - 2*den(-1+j)^4
       + den(-1+j)^5 - den(-1+j)^2*S(R(3),j)
       + 2*den(-1+j)*S(R(3),j) - den(j)
       + 91/32*den(j)^2 + 51/16*den(j)^3 + 9/4*den(j)^4
       + 5/4*den(j)^5 + den(j)^7 - den(j)^4*S(R(3),j)
       + den(j)^2*S(R(2,3),j) - 5/4*den(j)^2*S(R(3),j)
       - den(j)^2*S(R(5),j) - 9/4*den(j)*S(R(3),j));

#call Boundaryfix

#call IBIS;

Print;

.end
```

5.3 Example 3: More than two harmonic sums and a harmonic sums with a zero index

Lastly, suppose we want to solve the following inverse binomial sum:

$$\sum_{j=1}^{n-1} \frac{j!(n-j)!}{n!} \frac{(-1)^j}{j^2} S_{0,-4}(n-j) S_2(n-j) S_1(j) \tag{5.5}$$

Since this inverse binomial sum contains a harmonic sum with a zero index, we first have to rewrite this harmonic sum in terms of harmonic sums without a zero index. We can do this by applying the following FORM code:

```
#include- summer.h

Local F1 = sum1(j1,1,n-j)*sum2(j2,1,j1)*sign(j2)
           *den(j2)^4;

```

```
#call summer(2)

id theta(?a) = 1;

Print;

.end
```

This code will give us the following output:

```
S(R(-4),-j+n) - S(R(-4),-j+n)*j + S(R(-4),-j+n)*n - S(R(-3),-j+n);    (5.6)
```

To make sure that we have a maximum of two harmonic sums, we substitute this result back into our inverse binomial sum, make the change $j \to j1$, and apply the following FORM code:

```
#include- summer.h

L F1 = S(R(2),n-j1)*S(R(1),j1)*
       (S(R(-4),-j1+n)*den(j1)^2
       - S(R(-4),-j1+n)*den(j1)
       + S(R(-4),-j1+n)*n*den(j1)^2
       - S(R(-3),-j1+n)*den(j1)^2);

#call summer(1)

Print;

.end
```

This FORM code gives us the following output:

```
- den(j1)^2*S(R(-6),-j1+n)*S(R(1),j1)
- den(j1)^2*S(R(-6),-j1+n)*S(R(1),j1)*n
+ den(j1)^2*S(R(-5),-j1+n)*S(R(1),j1)
+ den(j1)^2*S(R(1),j1)*S(R(-4,2),-j1+n)
+ den(j1)^2*S(R(1),j1)*S(R(-4,2),-j1+n)*n
- den(j1)^2*S(R(1),j1)*S(R(-3,2),-j1+n)
+ den(j1)^2*S(R(1),j1)*S(R(2,-4),-j1+n)
+ den(j1)^2*S(R(1),j1)*S(R(2,-4),-j1+n)*n
- den(j1)^2*S(R(1),j1)*S(R(2,-3),-j1+n)
+ den(j1)*S(R(-6),-j1+n)*S(R(1),j1)
- den(j1)*S(R(1),j1)*S(R(-4,2),-j1+n)
- den(j1)*S(R(1),j1)*S(R(2,-4),-j1+n);                (5.7)
```

If we now change back $j1 \to j$ and plug this into our inverse binomial sum, we can solve the inverse binomial sum with the following FORM code:

```
#include- IBIS.h

L F1 = sum(j,1,n-1)*invbino(n,j)*sign(j)*
        (- den(j)^2*S(R(-6),-j+n)*S(R(1),j)
        - den(j)^2*S(R(-6),-j+n)*S(R(1),j)*n
        + den(j)^2*S(R(-5),-j+n)*S(R(1),j)
        + den(j)^2*S(R(1),j)*S(R(-4,2),-j+n)
        + den(j)^2*S(R(1),j)*S(R(-4,2),-j+n)*n
        - den(j)^2*S(R(1),j)*S(R(-3,2),-j+n)
        + den(j)^2*S(R(1),j)*S(R(2,-4),-j+n)
        + den(j)^2*S(R(1),j)*S(R(2,-4),-j+n)*n
        - den(j)^2*S(R(1),j)*S(R(2,-3),-j+n)
        + den(j)*S(R(-6),-j+n)*S(R(1),j)
        - den(j)*S(R(1),j)*S(R(-4,2),-j+n)
        - den(j)*S(R(1),j)*S(R(2,-4),-j+n));

#call IBIS;

Print;

.end
```

Chapter 6

Output

The output generated by IBIS will almost always be expressed in terms of S-sums. However, exceptions exist. If the user employs the `Boundaryfix` procedure or calculates an inverse binomial sum with a denominator of the form $\frac{1}{j+c}$ where $c < 0$, IBIS may also produce results involving the `sum` function. Sec. 4.3 provides an example of how to interpret this `sum` output. For users familiar with the `sum_` function in FORM, this notation should feel intuitive.

While the interpretation and application of the output are left entirely to the user's discretion, we offer some guidance for users who either calculate an inverse binomial sum with a specific numerical value of n or want to validate their numerical results using `Mathematica`. For numerical evaluations of S-sums, the `SubS` procedure from XSUMMER [5] can be particularly helpful. This procedure computes S-sums with numerical arguments and directly outputs their numerical value. To give you an example, see the following FORM code that calculates the value of $S(R(1,3,4,2),X(2,5,6,7),9)$:

```
#include- Xsummer.h

L F1 = S(R(1,3,4,2),X(2,5,6,7),9);

#call SubsS(1)

Print;

.end
```

Executing this code produces the following result:

$$S(R(1,3,4,2),X(2,5,6,7),9) = \frac{1145003952466473261191107542592781}{7469474397235200000}\,. \tag{6.1}$$

By substituting a numerical value for n, users can verify the correctness of IBIS's output. An approach to validate the output is to export it directly from FORM into `Mathematica` and compare the results. To export the result, the following FORM commands can be used:

```
Format mathematica;
#write <Speedtest.m> "{F1 -> (%E)}",F1;
```

Note that the variable must be named $F1$ for the final line to work correctly. If any discrepancies are observed between `Mathematica` and the IBIS output, users are encouraged to refer to the troubleshooting guidance in the next chapter.

The complete process of solving the inverse binomial sums is summarized in Fig 6.1, the processes with round corners are part of the IBIS program while the other processes are not.

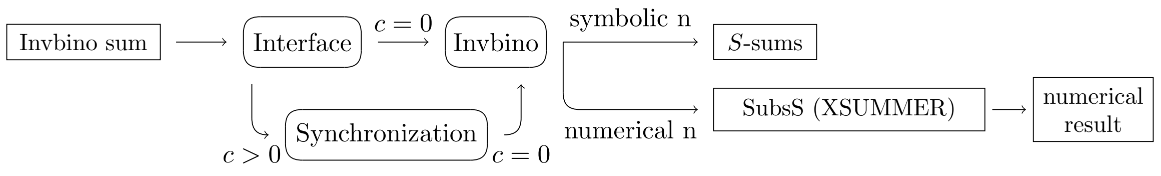

Figure 6.1: Internal structure of the package.

Chapter 7

Commonly encountered problems

Before proceeding, check the following:

- Did you make any typos in your input?
- Have you specifically named the summation variable j inside the inverse binomial sum?

A common indicator of errors is the presence of terms outside the `sum` function that still depend on j. This typically indicates that the inverse binomial sum has not been written down in a form that can be handled by IBIS.

To assist the users in avoiding common mistakes, several checks have been implemented in IBIS. Several of these validation checks have been discussed earlier in the manual. Additional examples include a warning system that alerts the user if their input contains an inverse binomial sum with a zero index or if the upper bound of the sum is not set to $n-1$. If the user makes such a mistake, the program will be terminated and an error message will show up telling the user what has gone wrong. These checks aim to prevent unintentional errors and ensure smoother usage. However, it is impossible to account for every potential input mistake. After running a computation, users are therefore encouraged to review the results carefully to ensure correctness. If you discover an unusual error or behavior that cannot be attributed to incorrect input, please reach out to us at w.j.waalewijn@uva.nl. We welcome feedback and opportunities to improve the program.

Bibliography

[1] J. A. M. Vermaseren. *New features of FORM*. Oct. 2000. arXiv: math-ph/0010025.

[2] B. Ruijl, T. Ueda, and J.A.M. Vermaseren. *FORM version 4.2*. July 2017. arXiv: 1707.06453 [hep-ph].

[3] P.A.J.W. van Hoegaerden, C.B. Marinissen, and W.J. Waalewijn. "IBIS: Inverse BInomial sum Solver". In preparation. 2025.

[4] J. A. M. Vermaseren. "Harmonic sums, Mellin transforms and integrals". In: *Int. J. Mod. Phys. A* 14 (1999), pp. 2037–2076. DOI: 10.1142/S0217751X99001032. arXiv: hep-ph/9806280.

[5] S. Moch and P. Uwer. "XSummer: Transcendental functions and symbolic summation in form". In: *Comput. Phys. Commun.* 174 (2006), pp. 759–770. DOI: 10.1016/j.cpc.2005.12.014. arXiv: math-ph/0508008.

Appendix A

Justification for setting delta to 0 and deltap to 1

When handling products of harmonic sums, the `densplit` procedure in Summer may introduce delta and deltap functions in the prefactor C (as defined in Eq. (4.3)) to handle terms like $\frac{1}{n+c}$, ensuring no division by zero occurs. The `deltap` and `delta` functions appear during the fraction-splitting process applied to terms of the form $\frac{1}{(j+a)(n-j+b)}$. These denominators are introduced by the `synchss` procedure, so we know that a is an integer between the minimum and maximum value of $\{a_1, ..., a_m\}$ and that b is an integer between the minimum and maximum of $\{b_1, ..., b_l\}$. When splitting these fractions, the $(y - x)$ term in Eq. (4.4) becomes $-(b + n + a)$. To justify setting `deltap` to 1 and `delta` to 0, we must show that, with correctly chosen summation boundaries, $b + n + a \neq 0$. For the inverse binomial sum containing the general product of harmonic sums in Eq. (4.1), we know that the correct summation boundaries are:

- Upper limit: $\min\{n - 1, n + b_{\min} - 1\}$,
- Lower limit: $\max\{1, -a_{\min} + 1\}$,

where $a_{\min} = \min(a_1, ..., a_m)$ and $b_{\min} = \min(b_1, ..., b_l)$. For the inverse binomial sum to be non-empty, the upper and lower limit must satisfy $n + b_{\min} - 1 \geq -a_{\min} + 1$. If we combine this with the fact that $-a_{\min} > -a$ and $b > b_{\min}$, we find that:

$$n + b - 1 > n + b_{\min} - 1 \geq -a_{\min} + 1 > -a + 1\,, \tag{A.1}$$

which can be rewritten into:

$$n + a + b > 2\,. \tag{A.2}$$

This inequality guarantees that $b + n + a \neq 0$, eliminating the risk of division by zero. Consequently, we can safely simplify the expression by setting `deltap` function to 1 and the `delta` function to 0.