ICE — the IBP Chooser of Equations

Philipp Kant

2016-01-17 Sun

This is the manual of ICE, a program to choose a maximal linearly independent subset from a given set of Integration-by-Parts and/or Lorentz Invariance equations. The algorithm it implements is described in [1], which is also the reference that should be cited when ICE is used in a calculation leading to a scientific publication.

1 Installation

The easiest way to compile the program from source is to use the stack build tool, avaialble at `https://github.com/commercialhaskell/stack`. Executing

```
stack install
```

in the directory containing the sources of ice will install the ice executable to `$HOME/.local/bin/ice`. You can either add that directory to your PATH, or use stack to run ice, as in

```
stack exec -- ice -id -im example/se1l.in
```

The stack tool will automatically download the GHC compiler and all dependencies.

2 Usage

ICE is run as

```
ice [OPTIONS] [FILE]
```

where the following options are available:

`-d --dumpfile=FILE` In addition to the output on stdout, print a list of newline-separated equation numbers to `FILE`. Note that the equations are zero-indexed.

`-l --logfile=ITEM` Specify where to write the logfile. Default is `ice.log`.

`--intname=NAME` This is used to control the name of the function representing integrals in the input file. The default is `Int`.

-i --invariants=x Add the symbol `x` to the list of variables that appear in the polynomials.

--sortList Sort the list of linearly independent equations. Otherwise, prints a permutation that brings the matrix as close to upper triangular form as possible.

-b --backsub After forward elimination, perform backward elimination in order to determine which master integrals appear in the result for each integral.

-r --rmax=n, **-s --smax=n** Only relevant if `--backsub` is given. Do not try to find a representation for integrals with more than `rmax` dots or more than `smax` scalar products. A system of IBP equations will typically contain some integrals with many dots and/or scalar products (more than the integrals used as seeds in the generation of the system) that can not be determined by the system, but are not master integrals and could be reduced if the system was enlarged. Discarding those before the backward elimination saves some time.

-v --visualize Draw images of the sparsity pattern of original, reduced, and solved matrices (only works for moderately sized systems).

-f --failbound=NUM Repeat forward elimination to decrease probability of failure below NUM.

-p --pipes use stdin and stdout for communication instead of files.

-? --help Display help message

-V --version Print version information

The input file `FILE` should have the following syntax:

- Each line gives one term in an equation in the form
  `Int[<indices>]*(<sum of terms>)` \ The brackets are mandatory.
- Each term consits of
  - a sign, + or -. If the first term is positive, its sign can be omitted.
  - a (positive) integer coefficient. If the coefficient is one, this can be omitted.
  - a multiplication operator * (unless the integer coefficient is omitted), followed by one of the strings defined as an invariant using the `-i` option, possibly followed by ^ and a positive exponent. This can be repeated, separated by the multiplication operator *.
- Equations are terminated and separated by a line consisting of only a semicolon.

For an example, see the following section.

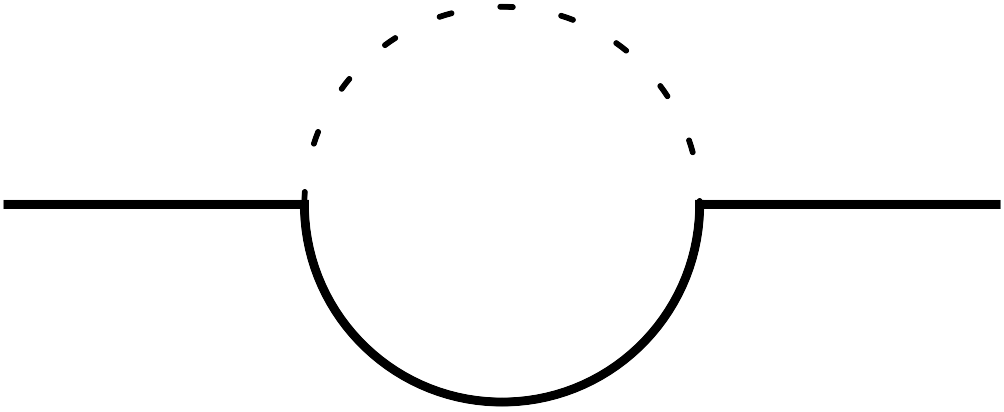

Figure 1: One-Loop massive self-energy

3 Example: One-Loop Massive Self-Energy

In order to illustrate the usage of the program, we give a simple example input file for the reduction of the diagram shown in Figure **??**. The input file with the equations is found in `example/se1l.in`. It contains equations to reduce one scalar product and one dot, with invariants `d` and `m`. The command line to run ICE on this file is

```
./dist/build/ice/ice -id -im example/se1l.in
```

or

```
./dist/build/ice/ice -id -im -r1 -s1 --backsub example/se1l.in
```

In the latter case, ICE also determines which master integrals are needed to express each integral with at most one dot and/or one scalar product. This will create the file `ice.log` with contents similar to

```
ICE -- Integration-By-Parts Chooser of Equations
Command line arguments: Config {inputFile = "example/se1l.in"
  , dumpFile = "", logFile = "ice.log", intName = "Int"
  , invariants = ["d","m"], sortList = False, backsub = True
  , rMax = 1, sMax = 1, visualize = False, failBound = 1.0
  , pipes = False}
Number of equations: 8
Number of integrals: 8
Number of integrals within r=1, s=1: 4
Probing for p = 3036998401
Random points: [2034626856,325408928]
The probability that too many equations were discarded
  is less than 9.219629504286786e-9
Number of linearly independent equations: 7
```

```
Indices of linearly independent equations (starting at 0):
5
4
6
1
0
2
3
Integrals that can be reduced with these equations:
Int[2,-1]
Int[2,0]
Int[1,-1]
Possible Master Integrals:
Int[1,0]
Performing backward elimination.
Final representations of the integrals will look like:
Int[2,-1] -> {Int[1,0]}
Int[2,0] -> {Int[1,0]}
Int[1,-1] -> {Int[1,0]}
Timings (wall time):
Parsing and preparing equations: 0.000289s
Solving Equations: 0.000386s
```

First, ICE reports the values of the command line arguments given. Next, the number of equations and integrals, as well as the number of integrals that lie within the region given by the values of `rmax` and `smax` is listed.

After that, the actual algorithm starts. The prime number and evaluation point is given, followed by the number of linearly independent equations and their positions in the input file (starting with zero). This is the main information of interest for a subsequent run of Laporta's Algorithm. With the option `--dumpfile`, it is possible to write this list to a separate file.

Finally, we get information on which integrals were reduced and which are considered master integrals. In case the `--backsub` option is given, ICE also gives a lists of which master integrals appear in the expression for each integral that could be reduced.

In addition to creating the logfile, ICE will also print the numbers of the linearly independent equations to `stdout`.

4 Details on the Implementation

Internally, a run of ICE consists of the following steps:

- Parsing of the input file
- Ordering of the integrals and bringing the system to matrix form
- Processing the matrix with the algorithm of [1]

- Optionally, performing backward elimination in order to determine which master integrals are needed to express each integral

In the following, we give some remarks about each step.

4.1 Parsing of the Input File

The syntax of the input file has been described above, and an example is distributed along with ICE. As ICE will typically be used on large problems in an automated toolchain, the parser has been designed for speed, not for helpful error messages. In particular, unless all invariants appearing in the equations are declared via the `-i` command line option, the program will crash.

4.2 Ordering of the Integrals

In order to express complicated integrals in terms of easier ones, we perform an ordering on the integrals, in decreasing order, following [2]. By inserting the integrals into a binary search tree, we assign a number to each integral. This number corresponds to the column number in the matrix.

The ordering determines which integrals are considered master integrals by ICE. Should one wish to change it (for instance, to prefer scalar products over dots in the master integrals), the definition of the ordering is found in `./Ice/Types.hs` and can easily be modified.

4.3 Main Algorithm

The prime p defining the field $\mathbb{F}_p$, and the evaluation point, is chosen randomly. For efficiency, the program contains a list of 100 pre-calculated large prime numbers. Large in this context means that they are as large as possible under the constraint that their square can be represented as a 64bit integer.

For a detailed description of the main algorithm, see [1].

4.4 Optional Backward Elimination

Sometimes, it can be desirable to know which master integrals appear in the expression for a certain integral (for example, one could drop certain coefficients known to be zero at an earlier stage of the reduction). This knowledge is easily obtained by performing a backward elimination and noting which entries of the resulting matrix are non-zero. Ice performs this step if the command line argument `--backsub` is provided.

In a given system of IBP identities, there will be some integrals with more dots and/or scalar products than in the integrals used to generate the system. Some of these integrals can not be reduced to master integrals without enlarging the system, so ICE will drop (after the forward elimination) any equations that still contain integrals with more dots (scalar products) than allowed by the option `--rmax` (`--smax`).

References

[1] Philipp Kant. Finding Linear Dependencies in Integration-By-Parts Equations: A Monte Carlo Approach. *Comput. Phys. Commun.*, 185:1473–1476, 2014.

[2] S. Laporta. High precision calculation of multiloop Feynman integrals by difference equations. *Int. J. Mod. Phys.*, A15:5087–5159, 2000.