IBP Reduction in Form

Kay Schönwald

in collaboration with Joshua Davies

Funded by
the European Union

June 22, 2026

Outline

- Introduction
- General Idea
- Conclusions and Outlook

Introduction

Introduction

- Long history of IBP reduction in form:
  - `mincer` [Larin, Tkachov, Vermaseren '91]
  - `forcer` [Ueda, Rujil, Vermaseren '16]
  - `matad` [Steinhauser '00]
  - `fmft` [Pikelner '17]
  - ...
- Specialized codes, mostly hand written and optimized for a specific kind of topology (propagators or tadpoles).
- Use cases: Mellin moments, asymptotic expansions, renormalization constants, ...

$\Rightarrow$ Usually high number of dots and scalar products, i.e. a real challenge for the Laporta algorithm

General Idea

General Idea

- If we want to be flexible, we cannot rely on hand written code.
- Utilize the capabilities of `LiteRed(2)` to generate recursive reduction rules. [Lee '12-]

Pros

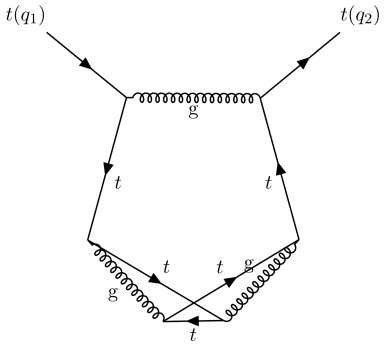
- `LiteRed(2)` is automated and can find reduction rules for many topologies.
- No hand-written intervention necessary.
- ...

Cons

- We rely on `LiteRed(2)` (and therefore `Mathematica` to find the reduction rules.
- Reduction rules close, but are (probably) not optimized.
- Cannot proceed if `LiteRed(2)` fails.

Example: going beyond `mincer`

- `mincer` implements massless, off-shell two-point functions.
- What happens if we add an internal massive particle?



f3x1111111001 with $q^2 = -Q^2$

$$k_1^2 - m^2, \quad (k_1+q)^2 - m^2$$
$$k_2^2 - m^2, \quad (k_2+q)^2 - m^2$$
$$k_3^2 - m^2, \quad (k_3+q)^2 - m^2$$
$$(k_1-k_2)^2, \quad (k_2-k_3)^2$$
$$(k_1-k_2+k_3)^2 - m^2$$

Implementation

- Run `LiteRed(2)` to obtain:
  1. zero sector (176)
- Translate to form routines:

```
id f3x111111001(x1?neg0_,x2?neg0_,x3?neg0_,x4?neg0_,x5?neg0_,x6?neg0_,x7?neg0_,x8?neg0_,x9?neg0_) = 0 ;
id f3x111111001(x1?neg0_,x2?neg0_,x3?neg0_,x4?neg0_,x5?neg0_,x6?neg0_,x7?neg0_,x8?neg0_,x9?pos_) = 0 ;
id f3x111111001(x1?neg0_,x2?neg0_,x3?neg0_,x4?neg0_,x5?neg0_,x6?neg0_,x7?neg0_,x8?pos_,x9?neg0_) = 0 ;
id f3x111111001(x1?neg0_,x2?neg0_,x3?neg0_,x4?neg0_,x5?neg0_,x6?neg0_,x7?pos_,x8?neg0_,x9?neg0_) = 0 ;
id f3x111111001(x1?neg0_,x2?neg0_,x3?neg0_,x4?neg0_,x5?neg0_,x6?pos_,x7?neg0_,x8?neg0_,x9?neg0_) = 0 ;
id f3x111111001(x1?neg0_,x2?neg0_,x3?neg0_,x4?neg0_,x5?pos_,x6?neg0_,x7?neg0_,x8?neg0_,x9?neg0_) = 0 ;
```

```
In[8]:= ZeroSectors[f3x111111001]

Out[8]= {js[f3x111111001, 0, 0, 0, 0, 0, 0, 0, 0, 0],
         js[f3x111111001, 0, 0, 0, 0, 0, 0, 0, 0, 1],
         js[f3x111111001, 0, 0, 0, 0, 0, 0, 0, 1, 0],
         js[f3x111111001, 0, 0, 0, 0, 0, 0, 1, 0, 0],
         js[f3x111111001, 0, 0, 0, 0, 0, 1, 0, 0, 0],
         js[f3x111111001, 0, 0, 0, 0, 1, 0, 0, 0, 0],
         js[f3x111111001, 0, 0, 0, 1, 0, 0, 0, 0, 0],
         js[f3x111111001, 0, 0, 1, 0, 0, 0, 0, 0, 0],
         js[f3x111111001, 0, 1, 0, 0, 0, 0, 0, 0, 0],
         js[f3x111111001, 1, 0, 0, 0, 0, 0, 0, 0, 0],
```

Implementation

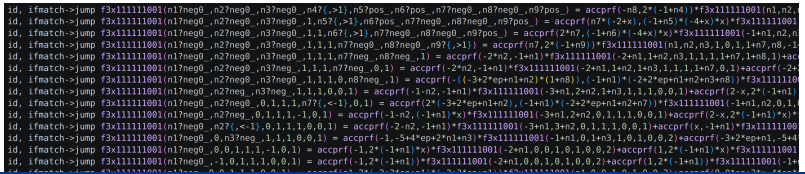
- Run `LiteRed(2)` to obtain:
  2. mapped sectors (278)
- Translate to form routines:

```
id, ifmatch->mapped f3x111111001(n1?neg0_,n2?pos_,n3?neg0_,n4?pos_,n5?neg0_,n6?neg0_,n7?pos_,n8?pos_,n9?pos_) = f3x111111001(n5,n6,n3,n4,n1,n2,n8,n7,n9);

,n9?pos_) = f3x111111001(0,0,n4,n3,n5,n6,n7,0,n9)*Pow(-f3x111111001(-1,0,0,0,0,0,0,0)+2*f3x111111001(0,0,-1,0,0,0,0,0)+2*f3x111111001(0,0,0,0,0,0,-1,0,0),-n2)*Po
```

Implementation

- Run `LiteRed(2)` to obtain:
  3. reduction rules for unique sectors (57) $\Rightarrow$ multiple rules per sector (overall 1436)
- Translate to form routines:



Implementation

- Some reduction rules have complicated pattern for the validity.
- Use $-variables to resolve these:

```
Identify f3x111111001(n1?$n1,n2?$n2,n3?$n3,n4?$n4,n5?$n5,n6?$n6,n7?$n7,n8?$n8,n9?$n9) = f3x111111001(n1,n2,n3,n4,n5,n6,n7,n8,n9)
```

```
$tmp1 = -$n1 + $n3 ;
if( $tmp1 !=  0 );
    id, ifmatch->jump f3x111111001(n1?neg0_,1,n3?neg0_,1,0,1,n7?neg_,0,1) = accprf(-1-n7,n1-n3)*f3x111111001(-1+n1,0,n3,1,0,1,2+n7,0,1)+accprf(1+n7,n1
endif;
id, ifmatch->jump f3x111111001(n1?neg_,1,0,1,n5?neg0_,1,0,n8?neg_,1) = accprf(-1-n8,1+n1+n8)*f3x111111001(-1+n1,1,-1,1,n5,1,0,2+n8,1)+accprf(n5,1+n1+n
id, ifmatch->jump f3x111111001(n1?neg0_,1,0,1,n5?neg0_,1,0,n8?{,<-1},1) = accprf(-1-n8,1+n1+n8)*f3x111111001(-1+n1,1,-1,1,n5,1,0,2+n8,1)+accprf(n5,1+n
$tmp1 = -$n1 + $n3 ;
$tmp2 = 1 - $n3 + $n5 ;
if( $tmp1 ==  0 );
if( $tmp2 !=  0 );
    id, ifmatch->jump f3x111111001(n1?neg0_,1,n3?neg_,1,n5?{,<-1},1,0,0,1) = accprf(-2*(-2+2*ep+n3),(-1+n3-n5)*(-1+2*ep+n5))*f3x111111001(-1+n3,0,n3,1
endif;
endif;
$tmp1 = -$n1 + $n3 ;
if( $tmp1 ==  0 );
    id, ifmatch->jump f3x111111001(n1?neg0_,1,n3?neg0_,1,0,1,n7?{,<-1},0,1) = accprf(-3*n3,-2+3*ep+n3+n7)*f3x111111001(-2+n3,1,1+n3,1,0,1,2+n7,0,1)+ac
endif;
id, ifmatch->jump f3x111111001(0,n2?pos_,0,n4?{,>2},0,2,0,0,1) = accprf(3-ep-n4,-1+n4)*f3x111111001(0,n2,0,-1+n4,0,2,0,0,1)+accprf(n2*(-1+ep+n2),(-2+n
id, ifmatch->jump f3x111111001(0,n2?pos_,0,n4?{,>2},0,1,0,0,2) = accprf(-n2,2*(-1+n4))*f3x111111001(-1,1+n2,0,-1+n4,0,2,0,0,0)+accprf(3-6*ep-3*n2+2*n4
id, ifmatch->jump f3x111111001(0,n2?{,>1},0,1,0,n6?{,>1},0,0,2) = accprf(-n2,2+x)*f3x111111001(-1,1+n2,0,0,0,n6,0,0,1)+accprf(3*n2,2*(2+x))*f3x1111110
id, ifmatch->jump f3x111111001(0,n2?{,>1},0,2,n5?neg_,1,0,0,1) = accprf(-3+2*ep+n2,-1+n2)*f3x111111001(0,-1+n2,0,2,1+n5,1,0,0,1)+accprf(2,-1+n2)*f3x11
```

Implementation

- Iterate `id` statements as long as reduction is not complete.
- Need to hide master integrals from the reduction rules:

```
#procedure reducef3x111111001
#do red=1,1
    #message f3x111111001: reduce level `$globalmaxlevel' ...
```

```
*   Hide MIs:
*--#[
    id f3x111111001(0,0,0,1,0,1,0,0,1) = MIf3x111111001(0,0,0,1,0,1,0,0,1) ;
    id f3x111111001(0,0,0,1,0,0,1,1,1) = MIf3x111111001(0,0,0,1,0,0,1,1,1) ;
    id f3x111111001(0,0,0,2,0,0,1,1,1) = MIf3x111111001(0,0,0,2,0,0,1,1,1) ;
    id f3x111111001(0,0,0,1,0,0,2,1,1) = MIf3x111111001(0,0,0,1,0,0,2,1,1) ;
    id f3x111111001(0,0,0,1,0,1,1,0,1) = MIf3x111111001(0,0,0,1,0,1,1,0,1) ;
    id f3x111111001(0,0,0,1,0,2,1,0,1) = MIf3x111111001(0,0,0,1,0,2,1,0,1) ;
    id f3x111111001(0,0,0,1,1,1,0,0,1) = MIf3x111111001(0,0,0,1,1,1,0,0,1) ;
    id f3x111111001(0,0,1,0,0,0,1,1,1) = MIf3x111111001(0,0,1,0,0,0,1,1,1) ;
```

```
*--#]

*   Continue the loop if integrals remain:
    If( occurs(f3x111111001) );
        Redefine red "0";
*       Recompute term integral level:
        Identify f3x111111001(?a) = f3x111111001(?a)*level(?a);
```

Optimizations – I

- We need a lot of `id` statements for the reductions:

$$176 + 278 + 1436 = 1890$$

- Avoid (redundant) pattern matching by calculating jumping to the right sector:

```
*--#[
*   Store indices, compute sector and level, init temporaries:
    Identify f3x111111001(n1?$n1,n2?$n2,n3?$n3,n4?$n4,n5?$n5,n6?$n6,n7?$n7,n8?$n8,n9?$n9) = f3x111111001(n1,n2,n3,n4,n5,n6,n7,n8,n9)*sectorid(n1,n2,n3,n4,n5,n6,n7,n8,n9)*leve
*--#]
```

```
*   Sector jump into Mapped Sectors or Reduction Rules:
*--#[
    Switch $sectorid;
        Case 49; Goto 49;
        Case 73; Goto 73;
        Case 81; Goto 81;
        Case 137; Goto 137;
        Case 145; Goto 145;
        Case 161; Goto 161;
        Case 168; Goto 168;
```

Optimizations – II

- Replace `id` statements for the vanishing sectors into a tablebase replacement
  $\Rightarrow$ Can save a lot of time!
- Determine the level of the integrals and only reduce the ones with the highest level
  $\Rightarrow$ potentially more terms can merge and we need to do less
  ??? usefulness seems to be very dependent on topology, but should be good for amplitude reductions
- Ideas from the audience?

Comparisons - f3x111111001

	(1, 1, 0, 1, 1, 1, 1, 1, 2)	(1, 1, -3, 1, 2, 1, 2, 1, 2)
`mathematica`	58s/330MB	21:41/732MB
`mathematica + fermat`	59s/330MB	16:58/729MB
`tform4.3 -w8`	1:24/1.1GB	33:40/1.7GB*
`tform5.0 -w8`	51s/1.0GB	12:20/5.0GB
`kira3`	44s/1.7GB	6:00/4.1GB(2:03/3.7GB)

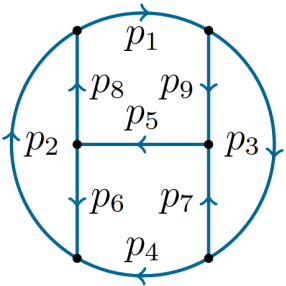
Given are time/memory as used on my laptop:
`AMD Ryzen 7 PRO 6850U (16 cores)/32GB RAM`

* Division by 0 if not `On Highfirst;`

Comparisons - four loop tadpoles

	tad4l(5,4,3,2,1,1,1,1,1,-5)	tad4l(9,8,7,6,5,4,3,2,1,0)
mathematica	20:58/750MB	–
mathematica + fermat	22:55/1.3GB	–
tform4.3 -w8	2:08/540MB	–
tform5.0 -w8	53s/800MB	1:46:29/5.6GB*
kira3	–	–

Given are time/memory as used on my laptop:
AMD Ryzen 7 PRO 6850U (16 cores)/32GB RAM



* Numerator degree up to 100!

Beneficial features

- More flexible integer pattern matching?
  $\Rightarrow$ Can we avoid the $\$$-variables with a more flexible pattern matching on integers?

Beneficial features

- More flexible integer pattern matching?
  $\Rightarrow$ Can we avoid the $-variables with a more flexible pattern matching on integers?
- faster rational arithmetic (`flint` probably the best we can do)
  $\Rightarrow$ When do we simplify the rational coefficients?
  Is it better to just simplify in the end or after each step?

Beneficial features

- More flexible integer pattern matching?
  $\Rightarrow$ Can we avoid the $-variables with a more flexible pattern matching on integers?
- faster rational arithmetic (`flint` probably the best we can do)
  $\Rightarrow$ When do we simplify the rational coefficients?
  Is it better to just simplify in the end or after each step?
- `MaxTermSize` (like always)
  $\Rightarrow$ reductions with many variables run into `MaxTermSize` issues rather fast
  ??? Can we do a reduction over finite fields in form?
  Rational reconstruction in form or link to external package?

Beneficial features

- More flexible integer pattern matching?
  $\Rightarrow$ Can we avoid the $-variables with a more flexible pattern matching on integers?
- faster rational arithmetic (`flint` probably the best we can do)
  $\Rightarrow$ When do we simplify the rational coefficients?
  Is it better to just simplify in the end or after each step?
- `MaxTermSize` (like always)
  $\Rightarrow$ reductions with many variables run into `MaxTermSize` issues rather fast
  ??? Can we do a reduction over finite fields in form?
  Rational reconstruction in form or link to external package?
- Can we get rid of `mathematica` completely?
  $\Rightarrow$ `LiteRed` in open source packages/form itself?

Conclusions and Outlook

Conclusions and Outlook

Conclusion:

- Automize IBP reduction in `Form` by utilizing recurrence relations provides by `LiteRed`.
- Potential to reduce integrals with a large number of dots and numerators without the memory hit in usual Laporta algorithm.

Outlook:

- Provide a package which automatizes `LiteRed` $\rightarrow$ `Form`.
- Optimization of `LiteRed` rules?

Thank You!