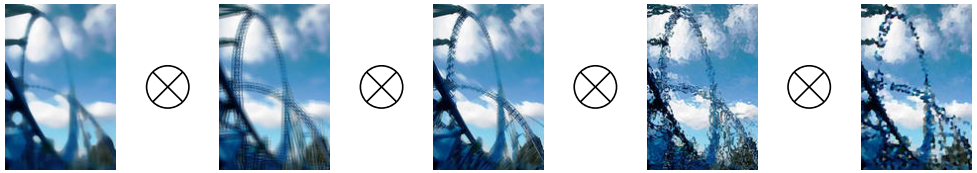
Applications of symbols and graph matroids and Reduze 2

Andreas v. Manteuffel

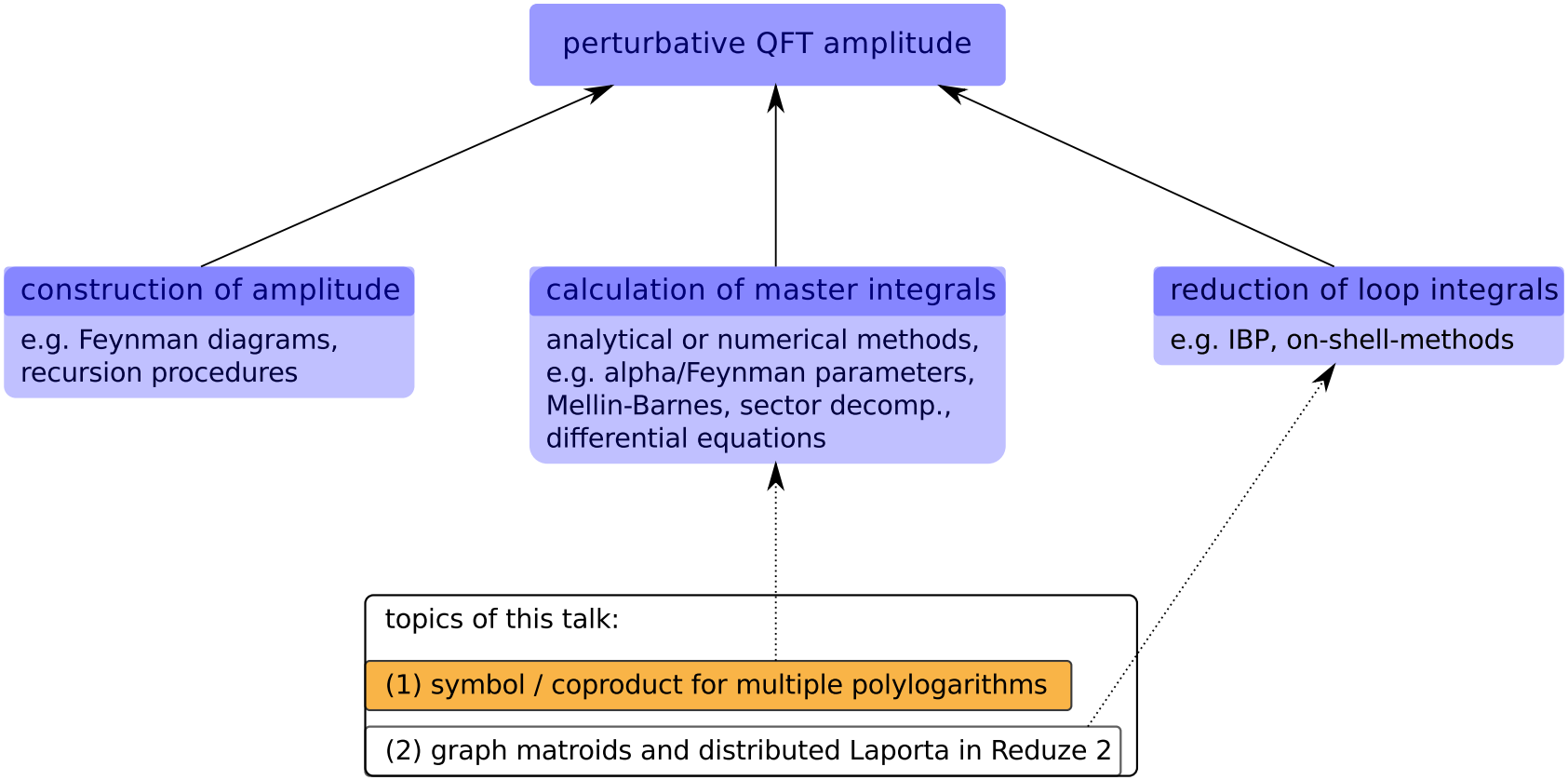
JOHANNES GUTENBERG UNIVERSITÄT MAINZ

PRiSMA

Frontiers in Perturbative Quantum Field Theory



10-12 September 2012, Bielefeld


perturbative QFT amplitude
construction of amplitude
e.g. Feynman diagrams, recursion procedures
calculation of master integrals
analytical or numerical methods, e.g. alpha/Feynman parameters, Mellin-Barnes, sector decomp., differential equations
reduction of loop integrals
e.g. IBP, on-shell-methods
topics of this talk:
(1) symbol / coproduct for multiple polylogarithms
(2) graph matroids and distributed Laporta in Reduze 2

Motivation for symbol calculus

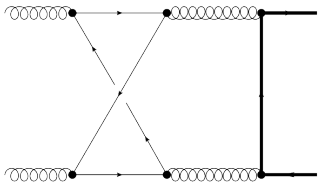
symbol calculus:

- tool to exploit functional relations between polylogarithms in a systematic way
  - works: simplify and transform functions; Zagier ('91), Goncharov ('95), Gangl ('02)
  - wish: perform integrations, see talk by C. Bogner

first application in theoretical physics:

- $N = 4$ supersymmetric Yang-Mills two-loop six-point remainder function
  Del Duca, Duhr, Smirnov (09): $\mathcal{O}(10^3)$ multi-dimensional polylogs
- simplifies to few-line expression with few Li_4, Li_2 and ln !
  found with symbols by Goncharov, Spradlin, Vergu, Volovich ('10)

symbols also very welcome for **QCD**, e.g.:

- e.g. 
- corner integral alone gives $\mathcal{O}(10^3)$ GPLs, many two-dimensional
- numerical evaluation slow, stability problematic
- switching basis functions, expansions, analytical continuation difficult
- simplifications guided by symbols possible; Ferroglia, AvM, Studerus

crucial progress very recently:

- construction of functions for given symbol; Duhr, Gangl, Rhodes ('11)
  - applied to HPLs; Bühler, Duhr ('11)
- extended symbol calculus from coproduct: **constants**;
  Goncharov ('02), Brown ('11), Duhr ('12)
  demonstrated for QCD amplitude; Duhr ('12)

Definition of symbol map

Let G be a generalized polylogarithm with

$$\mathrm{d}G = \sum_i \hat{G}_i \, \mathrm{d}\ln(R_i)$$

where R_i is a rational function of the polylog arguments. The **symbol map $\mathcal{S}$**

$$\mathcal{S}(G) = \sum_i \mathcal{S}(\hat{G}_i) \otimes R_i \,,$$

associates a tensor with the polylogarithm.

examples:

- $\mathcal{S}(\ln x) = x$
- $\mathcal{S}(\mathrm{Li}_3\, x) = -((1-x) \otimes x \otimes x)$
- $\mathcal{S}(G(1,0,-1,-1,x)) = (1+x) \otimes (1+x) \otimes x \otimes (1-x)$

Rules for symbols

- $R_1 \cdots \otimes (R_a R_b) \otimes \cdots R_k = R_1 \cdots \otimes R_a \otimes \cdots R_k + R_1 \cdots \otimes R_b \otimes \cdots R_k$ (log law)
- $R_1 \cdots \otimes (c R_a) \otimes \cdots R_k = R_1 \cdots \otimes R_a \otimes \cdots R_k$ for constant c
- preserves shuffle product

Example for symbol calculus

goal: derive "simplification formula" for $\mathrm{Li}_2(1/x)$ with $0 < x < 1$, $\mathrm{Im}\, x = \varepsilon$

$$\begin{aligned}\mathcal{S}(\mathrm{Li}_2(1/x)) &= -(-1 + 1/x) \otimes (1/x) \\ &= (1 - x) \otimes x - x \otimes x \\ &= \mathcal{S}(-\mathrm{Li}_2(x) - (1/2)\ln^2 x)\end{aligned}$$

reproduces the highest degree part of the full answer

$$\mathrm{Li}_2(1/x) = -\mathrm{Li}_2(x) - (1/2)\ln^2 x + i\pi \ln x - (2/3)\pi^2$$

note: works at **highest degree** only

- $\mathcal{S}(\ln(-x)) = \mathcal{S}(\ln(x))$: no info on discontinuity
- $\mathcal{S}(\pi) = \mathcal{S}(\zeta_3) = 0$: no constants

Systematic reduction to basis function with symbols

"Integrating the symbol"

Duhr, Gangl, Rhodes ('11)

1. compute symbol of expression to match and normalize symbol
2. pick functions: $\ln(R_1), \mathrm{Li}_2(R_1), \mathrm{Li}_3(R_1), \mathrm{Li}_4(R_1), \mathrm{Li}_{22}(R_1, R_2)$, ...
3. pick arguments: algorithmic procedure guided by symbol to match
4. build basis functions
5. iterative procedure
   - project symbol onto some subspace:
     elimination of shuffle products and (anti)symmetrizations
   - match against a specific class of basis function, substract this contribution

QCD application

$gg \to t\bar{t}$ **at two-loops**, ferm. contrib. (incl. non-planar part)

- pick specific coefficient in finite part: linear combination of polylogs
- pick uniform weight 4 part
- representation not unique (due to crossings etc.), imaginary polylogs
- here: 182 GPLs in total, 84 weight 4 non-Li_k GPLs
- symbol contains slots (weight have unit roots):

$$\{-1+x, x, 1+x, y, 1+y, x+y, 1+xy, 1+x^2+xy, 1-x+x^2+xy\}$$

algorithmic conversion to new set of real basis function:

1. partition (4) (no shuffles), antisymmetric part: $\mathrm{Li}_{22}(R_1, R_2)$ using $a(a(w,x), a(y,z))$ with $a(y,z) = y \otimes z - z \otimes y$
2. partition (4) (no shuffles), rest: $\mathrm{Li}_4(R_1)$
3. partition (3,1): $\mathrm{Li}_3(R_1)\ln(R_2)$
4. partition (2,2): $\mathrm{Li}_2(R_1)\mathrm{Li}_2(R_2)$
5. partition (2,1,1): $\mathrm{Li}_2(R_1)\ln(R_2)\ln(R_3)$
6. partition (1,1,1,1) (rest): $\ln(R_1)\ln(R_2)\ln(R_3)\ln(R_4)$

result:

- 28 Li_{22}, 48 Li_4, 58 Li_3, 18 Li_2, 11 ln
- all real

What about **subleading degree** *terms (const $\times$ polylog) ?*

- accessible by **coproduct**: Goncharov ('02), Brown ('11)
- extended symbol calculus based on coproduct by Duhr ('12):
  - systematic approach
  - applied to two-loop QCD amplitude

Definition of the coproduct

For a multiple polylogarithm

$$I(a_0; a_1, \ldots, a_n; a_{n+1}) = \int_{a_0}^{a_{n+1}} \frac{\mathrm{d}t}{t - a_n} I(a_0; a_1, \ldots, a_{n-1}; t)$$

the coproduct Δ is defined according to Goncharov ('02):

$$\Delta\left(I(a_0; a_1, \ldots, a_n; a_{n+1})\right) = \sum_{0=i_1<\ldots<i_{k+1}=n} I(a_0; a_{i_1}, \ldots, a_{i_k}; a_{n+1}) \otimes \prod_{p=0}^{k} I(a_{i_p}; a_{i_p+1}, \ldots, a_{i_{p+1}-1}$$

examples:

- $\Delta(\ln(x)) = 1 \otimes \ln(x) + \ln(x) \otimes 1$
- $\Delta(\mathrm{Li}_2(x)) = 1 \otimes \mathrm{Li}_2(x) - \ln(1-x) \otimes \ln(x) + \mathrm{Li}_2(x) \otimes 1$
- $\Delta(\ln(x)\ln(y)) = 1 \otimes (\ln(x)\ln(y)) + \ln(x) \otimes \ln(y) + \ln(y) \otimes \ln(x) + (\ln(x)\ln(y)) \otimes 1$

Rules for the coproduct

- coassociativity $(\mathrm{id} \otimes \Delta)\,\Delta = (\Delta \otimes \mathrm{id})\,\Delta$
- compatible with product: $\Delta(a \cdot b) = \Delta(a) \cdot \Delta(b)$ where $(a_1 \otimes a_2) \cdot (b_1 \otimes b_2) \equiv (a_1 \cdot b_1) \otimes (a_2 \cdot b_2)$

note: coproduct means "decomposition"

Graded decomposition with the coproduct

- Hopf algebra of multiple polylogs **graded by weight**:

$$\mathcal{H} = \bigoplus_{n=0}^{\infty} \mathcal{H}_n$$

since coproduct preserves weight we may decompose

$$\mathcal{H}_n \xrightarrow{\Delta} \bigoplus_{p+q=n} \mathcal{H}_p \otimes \mathcal{H}_q$$

and define $\Delta_{p,q}$ to be the part with values in $\mathcal{H}_p \otimes \mathcal{H}_q$

- iterated coproduct:

$$\mathcal{H} \xrightarrow{\Delta} \mathcal{H} \otimes \mathcal{H} \xrightarrow{\Delta \otimes \mathrm{id}} \mathcal{H} \otimes \mathcal{H} \otimes \mathcal{H} \xrightarrow{\Delta \otimes \mathrm{id} \otimes \mathrm{id}} \mathcal{H} \otimes \mathcal{H} \otimes \mathcal{H} \otimes \mathcal{H}$$

and corresponding parts $\Delta_{p,q,\ldots,r}$

- $\boxed{\text{symbol } \mathcal{S} = \text{maximally iterated coproduct } \Delta_{1,\ldots,1} \mod \pi}$

calculus can be extended to contain constants:

$$\Delta(\pi) = \pi \otimes 1$$
$$\Delta(\zeta_k) = \zeta_k \otimes 1 + 1 \otimes \zeta_k \qquad \text{for } k \text{ (odd)}$$

example:

$$\begin{aligned}\Delta_{1,1}\left(\mathrm{Li}_2(1/x)\right) &= -\ln(1-1/x) \otimes \ln(1/x) \\ &= \ln(1-x) \otimes \ln(x) - \ln(x) \otimes \ln(x) + i\pi \otimes \ln(x) \\ &= \Delta_{1,1}\left(-\mathrm{Li}_2(x) - (1/2)\ln^2(x) + i\pi \ln(x)\right)\end{aligned}$$

reproduces identity up to pure constant of maximal transcendality, here: π^2
fix constant by numerical evaluation + PSQL (or limit):

$$\mathrm{Li}_2(1/x) - \left(-\mathrm{Li}_2(x) - (1/2)\ln^2(x) + i\pi \ln(x)\right) = -6.5797362673929\cdots = -(2/3)\pi^2$$

systematic procedure: back to $gg \to t\bar{t}$ **at two-loops** application:

1. fix maximum degree 4 part with symbol, substract from original expression
2. take $\Delta_{1,1,1,1}$, use unique set of real ln, gives

$$i\pi \otimes \Delta_{1,1,1}(\text{something})$$

   match "something" at degree 3 to basis functions using symbols
3. take $\Delta_{2,1,1}$, use unique set of real ln, Li_2, gives

$$\pi^2 \otimes \Delta_{1,1}(\text{something})$$

   match "something" at degree 2 to basis functions using symbols
4. take $\Delta_{3,1}$, use unique set of real ln, Li_2, Li_3, gives

$$\zeta(3) \otimes \text{something1} + i\pi^3 \otimes \text{something2}$$

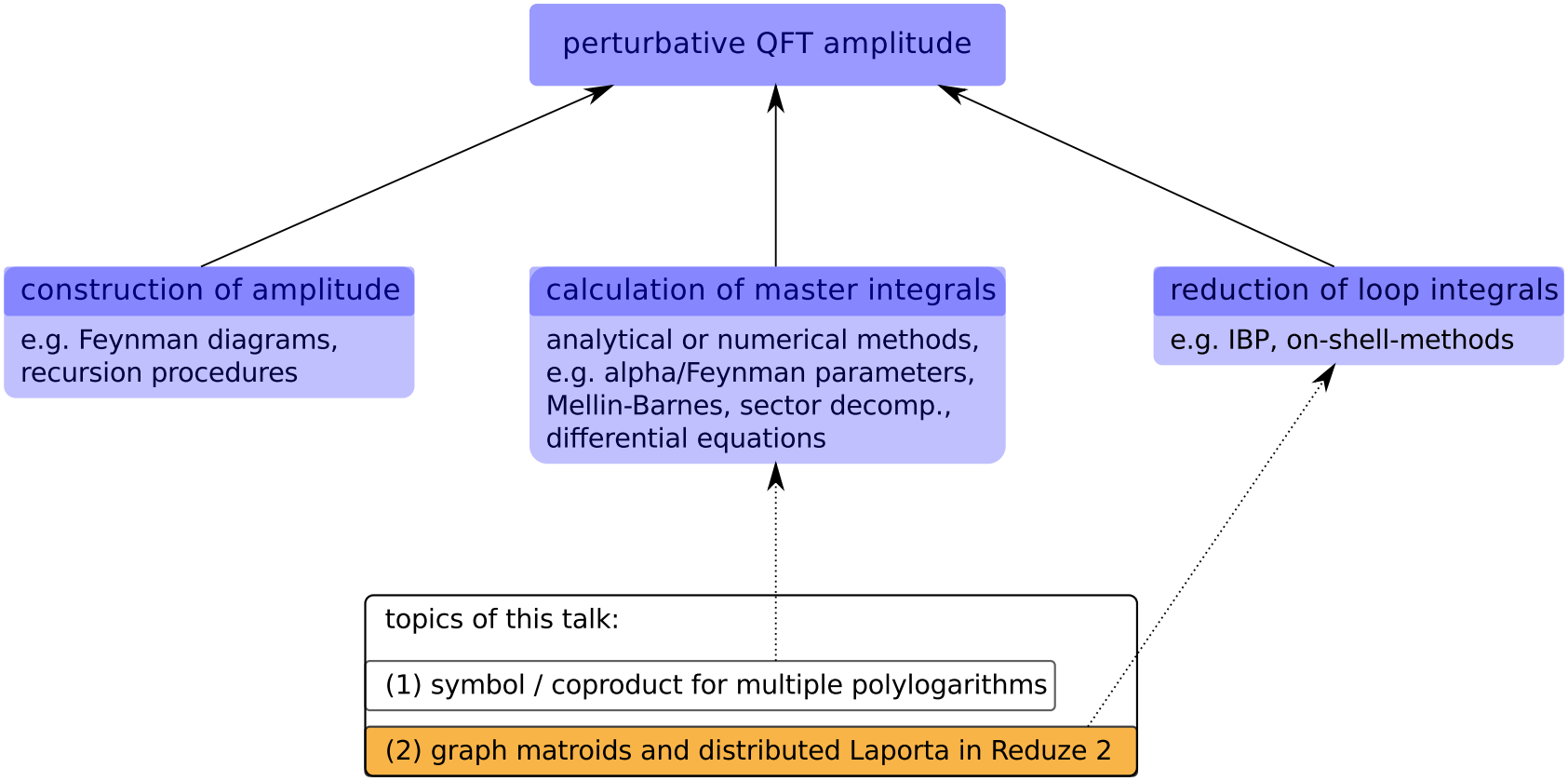
   read off "somethingi" (just logs) at degree 1
5. numerical evaluation, match against

$$i\pi\zeta_3, \quad \pi^4$$

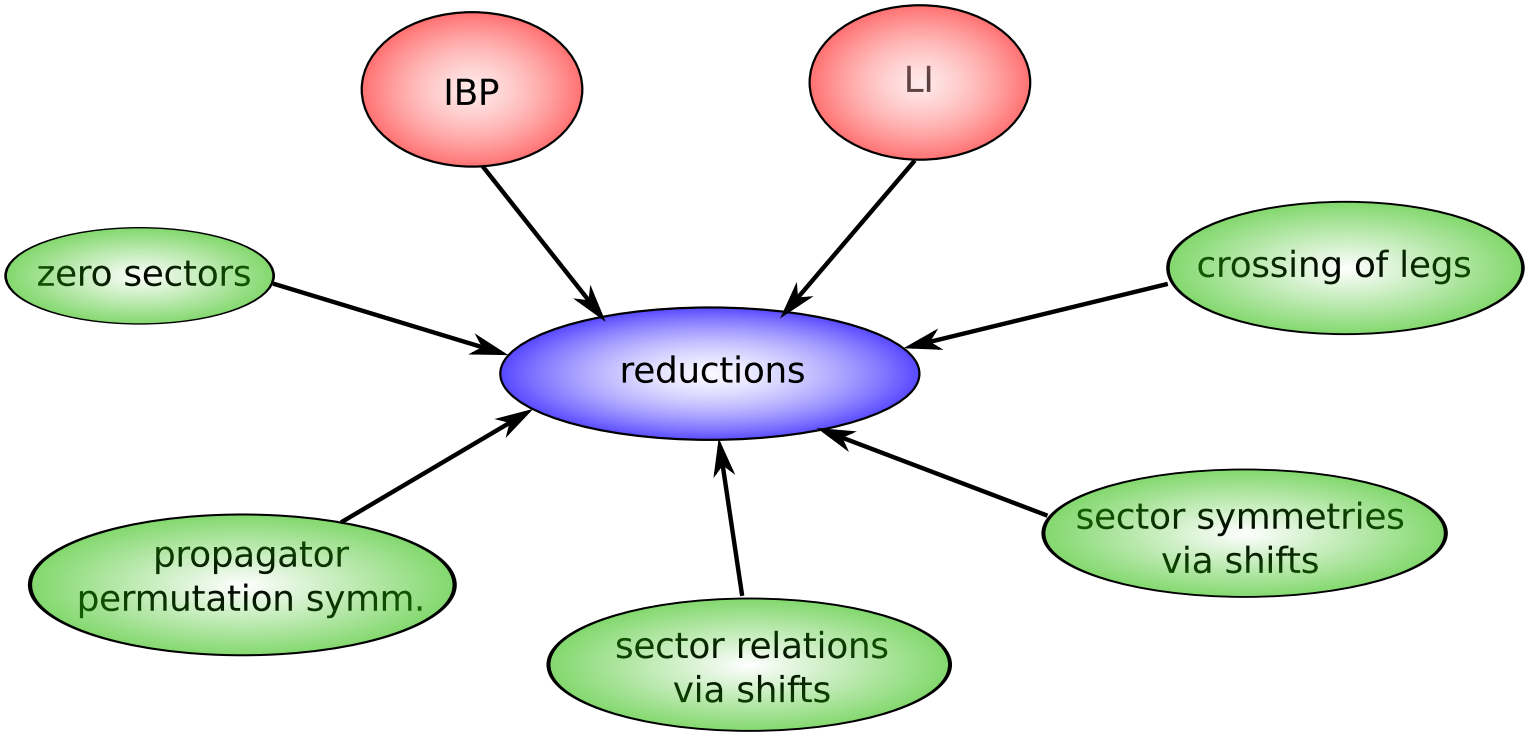
(requires matching of degree 3 and lower, obtained by recursion)

results for $gg \to t\bar{t}$:

- considerable simplifications for poles
- finite parts: heavy improvement in access wrt traditional techniques


perturbative QFT amplitude
construction of amplitude
e.g. Feynman diagrams, recursion procedures
calculation of master integrals
analytical or numerical methods, e.g. alpha/Feynman parameters, Mellin-Barnes, sector decomp., differential equations
reduction of loop integrals
e.g. IBP, on-shell-methods
topics of this talk:
(1) symbol / coproduct for multiple polylogarithms
(2) graph matroids and distributed Laporta in Reduze 2

Removing ambiguities for integrals



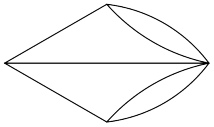
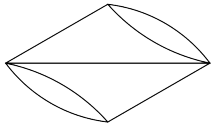
Reduze 2: sectors, graphs and matroids

$$
\begin{tikzcd}
\boxed{\textbf{integral A}\ (\text{propagators})} \arrow[d, "\textit{construct}"] & \exists\text{shift ?} \arrow[l] \arrow[r] & \boxed{\textbf{integral B}\ (\text{propagators})} \arrow[d, "\textit{construct}"] \\
\boxed{\textbf{graph A}} & \text{isomorphic ?} \arrow[l] \arrow[r] & \boxed{\textbf{graph B}}
\end{tikzcd}
$$

Reduze 2: sectors, graphs and matroids

$$\begin{tikzcd}
\boxed{\textbf{integral A}\ (\text{propagators})} \arrow[d, "\textit{construct}"] & \exists\text{shift ?} \arrow[l] \arrow[r] & \boxed{\textbf{integral B}\ (\text{propagators})} \arrow[d, "\textit{construct}"] \\
\boxed{\textbf{graph A}} & \text{isomorphic ?} \arrow[l] \arrow[r] & \boxed{\textbf{graph B}}
\end{tikzcd}$$

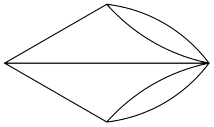
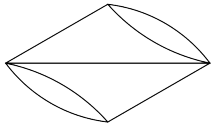
- **problem**: graphs not unique !

e.g.:  ⟵ same propagators but not isomorphic ⟶ 

Reduze 2: sectors, graphs and matroids

$$\begin{tikzcd}
\boxed{\textbf{integral A} \text{ (propagators)}} \arrow[d, "\textit{construct}"] & \exists\text{shift ?} \arrow[l] \arrow[r] & \boxed{\textbf{integral B} \text{ (propagators)}} \arrow[d, "\textit{construct}"] \\
\boxed{\textbf{graph A}} & \text{isomorphic ?} \arrow[l] \arrow[r] & \boxed{\textbf{graph B}}
\end{tikzcd}$$

- **problem**: graphs not unique !

e.g.:  ⟵ same propagators but not isomorphic ⟶ 

- **solution**: select unique representative by allowing for **twists**

idea based on **theorem** by Bogner, Weinzierl (2010):

first Symanzik polynomials $\mathcal{U}$ isomorphic $\Leftrightarrow$ graphs isomorphic **up to twists**

proof based on Whitney's theorem for isomorphisms of graph matroids

Definition of matroid

A matroid is a pair $(E, \mathcal{I})$ where E finite ground set, $\mathcal{I}$ collection of subsets of E, the **"independent sets"**, and

- $\emptyset \in \mathcal{I}$
- if $I \in \mathcal{I}$ and $I' \subset I$ then also $I' \in \mathcal{I}$
- if $I_1, I_2 \in \mathcal{I}$, $|I_1| < |I_2|$ then $\exists e \in I_2 - I_1$ with $I_1 \cup \{e\} \in \mathcal{I}$

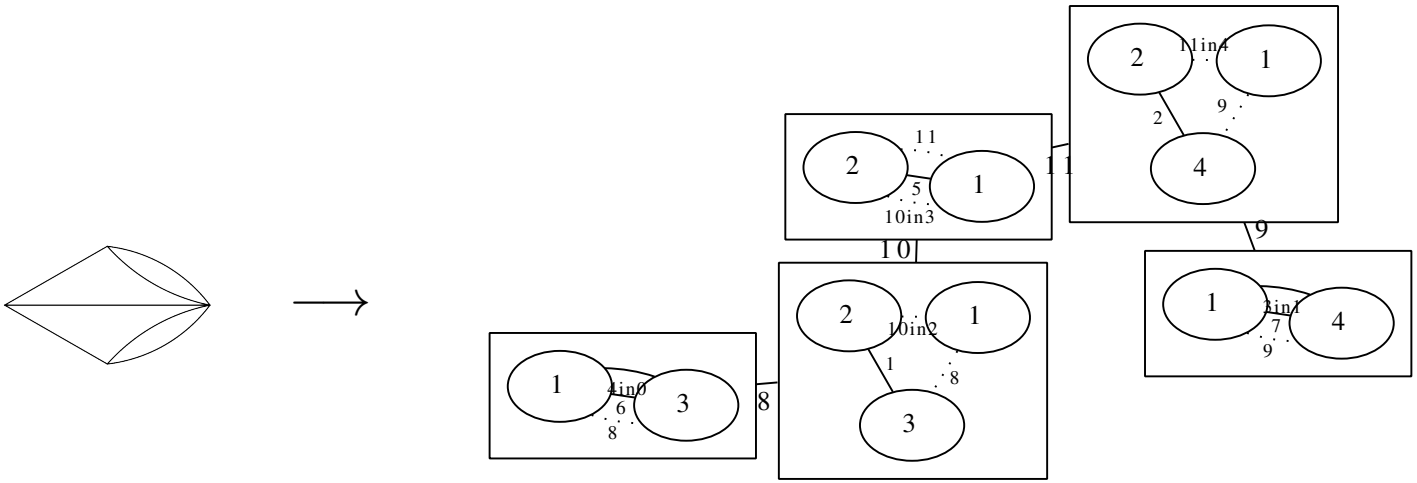
- generalizes notion of **linear dependency**
- graph matroid: dependencies of **edges**
- application to Feynman graph: propagators relevant, no reference to vertices

propagators of two vacuum diagrams related by shift $\Leftrightarrow$ graph matroids isomorphic

Algorithm: shift finder

1. generate graph for sector
2. colour edges according to masses
3. connect external legs with a new vertex
4. decompose into triconnected components (Hopcroft, Tarjan '73; Gutwenger, Mutzel '01)
5. minimize graph by twists
6. check for graph isomorphism (McKay '81)

example:



tree of triconnected components
(dashed "virtual edges" mark positions for Tutte twists)

tree of triconnected components:

all available in open source program:

```
_______    ________
___  __ \_________  /___  __________
__  /_/ /  _ \  __  /_  / / /__  /  _ \
_  _, _//  __/ /_/ / / /_/ /__  /_  __/
/_/ |_| \___/\__,_/  \__,_/ ____\___/
```

Reduze 2 - Distributed Feynman Integral Reduction

A.v.M., Studerus

arXiv:1201.4330

http://projects.hepforge.org/reduze

Application example

auto-generated shifts for non-planar double box family:

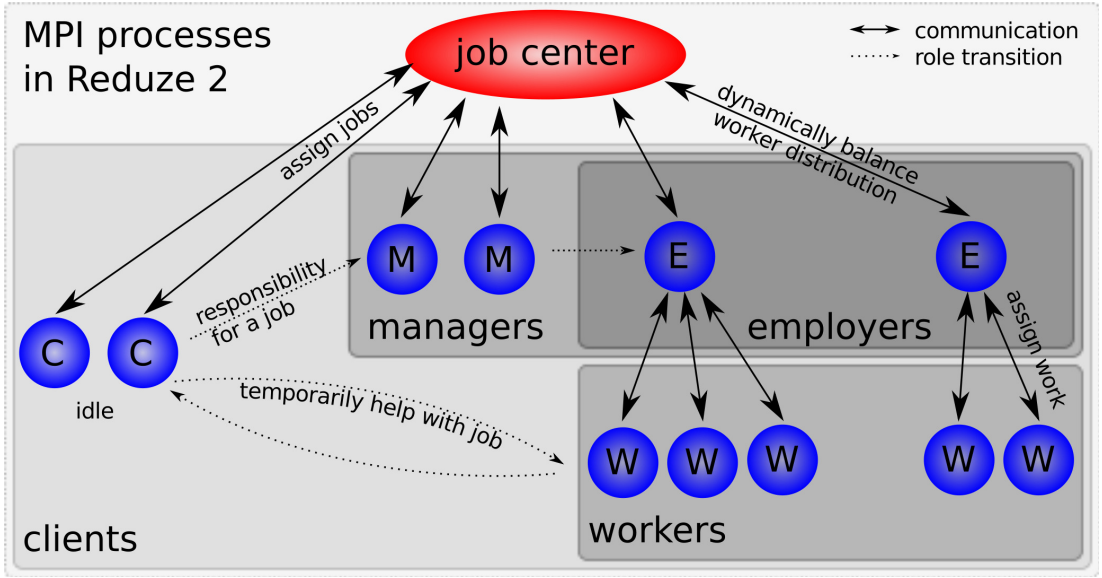
figures : less

Datei Bearbeiten Ansicht Verlauf Lesezeichen Einstellungen Hilfe

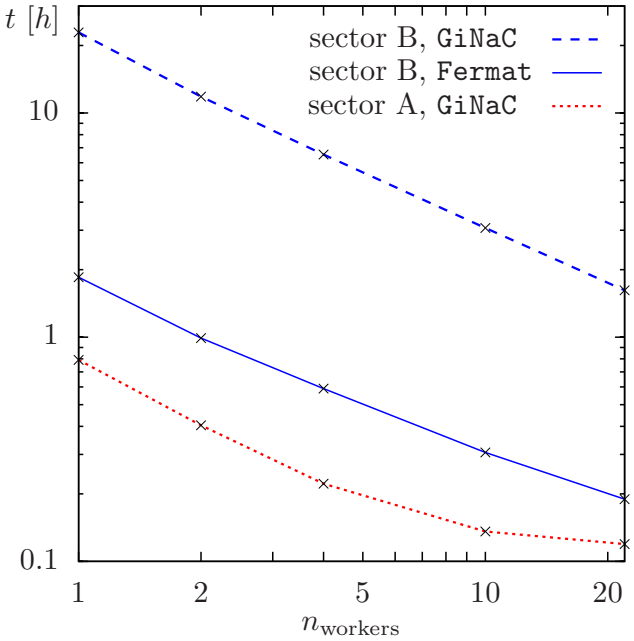
```
---
sectormappings:
  name: box2n
  zero_sectors:
    t=0: [0]
    t=1: [1, 4, 16, 64, 256]
    t=2: [3, 5, 6, 12, 17, 18, 20, 24, 48, 65, 66, 68, 72, 80, 96, 192, 257, 260, 272, 320]
    [ ... truncated ]
  sectors_without_graph:
    t=3: [22, 28, 52, 82, 84, 88, 104, 112, 276]
    t=4: [23, 29, 30, 53, 54, 60, 83, 85, 86, 89, 90, 92, 105, 106, 108, 113, 114, 116, 120, 21
    t=5: [31, 55, 61, 87, 91, 93, 94, 107, 109, 110, 115, 117, 118, 121, 122, 124, 211, 213, 21
    [ ... truncated ]
  sector_relations:
    3: [[box2p, 3], [[k1, k1], [k2, k2]]]
    6: [[box2p, 3], [[k1, -p1+k2], [k2, k1]]]
    7: [[box2p, 11], [[k1, k1-p1], [k2, k2]]]
    12: [[box2p, 3], [[k1, k1-p1], [k2, k2-p2]]]
    13: [[box2p, 11], [[k1, k1-p1], [k2, k2-p2]]]
    15: [[box2p, 58], [[k1, -p1+k2], [k2, k1-p1-p2]]]
    18: [[box2p, 3], [[k1, k2+p3], [k2, k1]]]
    19: [[box2px13x24, 11], [[k1, k1+p3], [k2, k2]]]
    24: [[box2p, 3], [[k1, k2+p3], [k2, k1-p2]]]
    25: [[box2px13x24, 11], [[k1, k1+p3], [k2, k2-p2]]]
    26: [[box2px12x34, 11], [[k1, k2+p3], [k2, k1-p2]]]
    27: [[box2px13, 58], [[k1, k2+p3], [k2, k1-p2+p3]]]
    [ ... truncated ]
  crossed_sector_relations:
    x12:
      207: [[box2n, 207], [[k1, -k2-p2], [k2, -k1-p1]]]
      335: [[box2n, 335], [[k1, -k2-p2], [k2, -k1-p1]]]
      463: [[box2n, 463], [[k1, -k2-p2], [k2, -k1-p1]]]
    x123:
      335: [[box2n, 335], [[k1, -k2-p2], [k2, k1+p1-k2]]]
    x1234:
      335: [[box2nx14x23, 335], [[k1, -k1+p1+k2-p3]]]
    x124:
      335: [[box2nx14x23, 335], [[k1, -k1+p1+k2-p3], [k2, -k1+p1+p2-p3]]]
    x1243:
      335: [[box2nx12x34, 335], [[k2, k1-k2+p2]]]
  sector_symmetries:
    207:
      - [[k1, -k1-p1], [k2, -k2-p2]]
      - [[k1, -k2-p2], [p1, p2], [k2, -k1-p1], [p2, p1], [p3, p1+p2-p3]]
      - [[k1, k2], [p1, p2], [k2, k1], [p2, p1], [p3, p1+p2-p3]]
    463:
      - [[k1, k2], [p1, p2], [k2, k1], [p2, p1], [p3, p1+p2-p3]]
(END)
```

figures : less

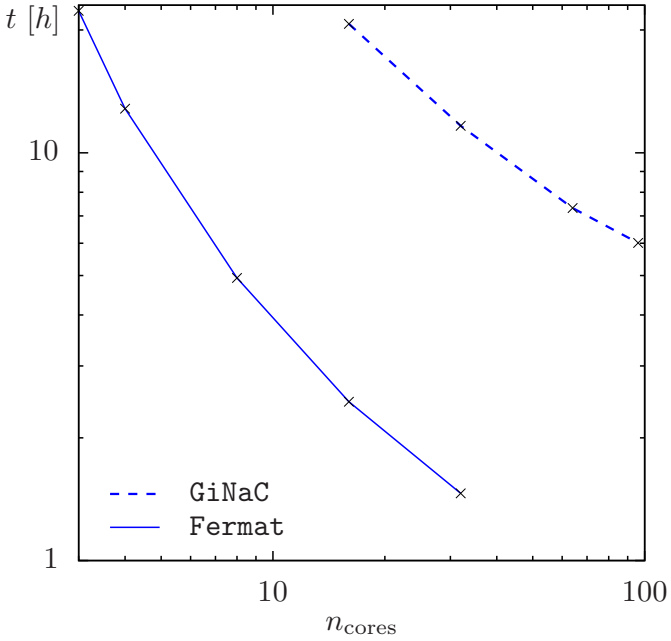
Reduze 2: distributed Laporta algorithm



reduction of single sector:



reduction of sector selection:



Available jobs in Reduze 2

andreas : bash

Datei Bearbeiten Ansicht Verlauf Lesezeichen Einstellungen Hilfe

```
andreas@chili:~$ reduze -h jobs

List of available job types:

apply_crossings:                  Generates reduction results for crossed sectors.
cat_files:                        Concatenates files.
collect_integrals:                Collects all integrals appearing in the input file.
compute_diagram_interferences:    Computes interferences of diagrams.
compute_differential_equations:   Computes derivatives of integrals wrt invariants.
export:                           Exports to FORM, Mathematica or Maple format.
find_diagram_shifts:              Matches diagrams to sectors via graphs.
find_diagram_shifts_alt:          Matches diagrams to sectors via combinatorics.
generate_identities:              Generates identities like IBPs for given seeds.
generate_seeds:                   Generates integrals from a sector.
insert_reductions:                Inserts reductions in expressions.
normalize:                        Simplifies linear combinations and equations.
print_reduction_info_file:        Analyzes reductions in a file.
print_reduction_info_sectors:     Analyzes reductions available for sectors.
print_sector_info:                Prints diagrams and other information for sectors.
reduce_files:                     Reduces identities in given files.
reduce_sectors:                   Reduces integrals from a selection of sectors.
run_reduction:                    Low-level job to run a reduction.
select_reductions:                Selects reductions for integrals.
setup_sector_mappings:            Finds shifts between sectors via graphs.
setup_sector_mappings_alt:        Finds shifts between sectors via combinatorics.
sum_terms:                        Sums terms.
test:                             Empty job template.
verify_same_terms:                Verifies two files contain the same terms.

andreas@chili:~$
```

andreas : bash

Conclusions

- **symbol + coproduct** refinement:
  - powerful algorithmic treatment of multiple polylogs
  - outlook: direct solving of differential equations ?
- **graph matroid algorithm**
  - allows to identify Feynman integrals
- Reduze 2: open source tool
  - parallelized reductions of Feynman integrals
  - finding shifts between sectors or diagrams