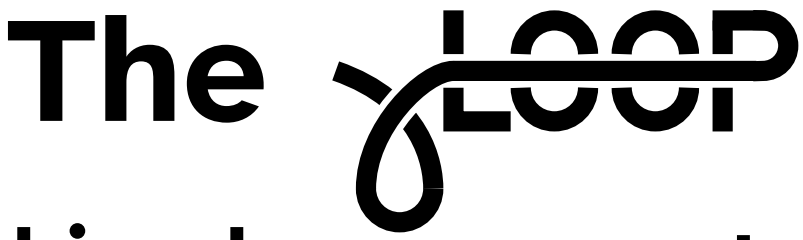
The  ecosystem

Making loop computations easier with `rust`

10.6.26 - GDR QCD Workshop Orsay

Lucien Huber

What is LOOP[1]?

rust based framework for differential cross sections using Local Unitarity

Design principle: reusable, generic components → a small ecosystem of Rust crates

- `gammaloop` → orchestration: UFO, Feynman rules, CFF integrands, threshold subtraction, UV R-operation, Local Unitarity
- `spenso` → tensor networks
- `idenso` → symbolic tensor simplification
- `linnet` → graph manipulation and drawing
- `feyngen`[2] → Feynman graph generation
- `momtrop` → tropical sampling in momentum space

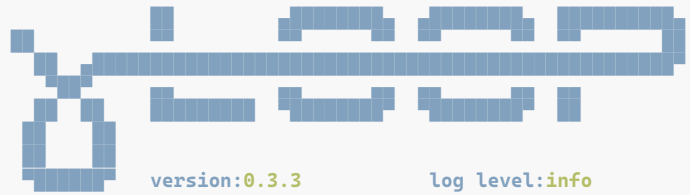
[1] Built by Valentin Hirschi, Zeno Cappati, Mathijs Fraaije, Ben Ruijl, Nic Fink, Kaapo Seppänen and Cedric Sigrist

[2] Part of `gammaloop`.

Diagram Generation

feyngen

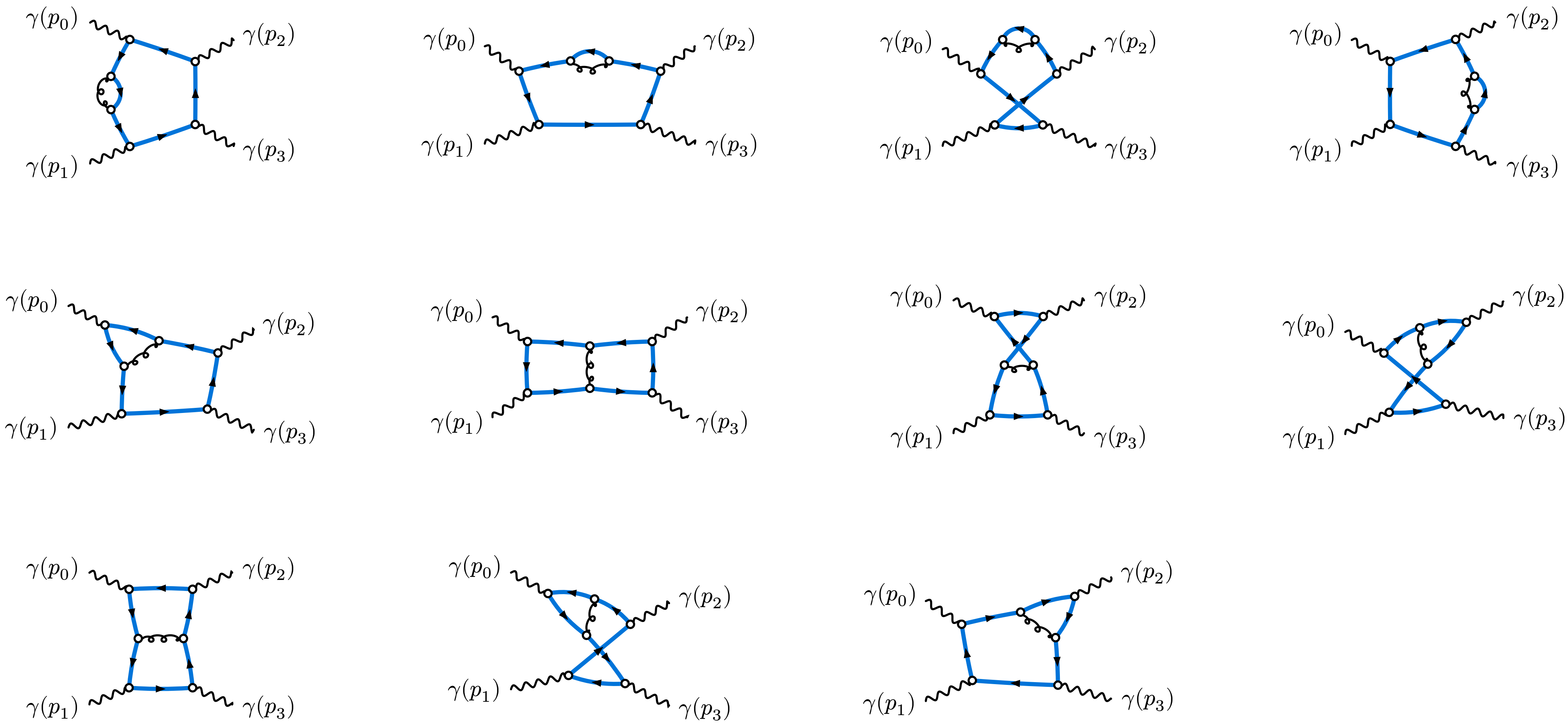
UFO model aware generation



```
version:0.3.3          log level:info

aa_aa_2L | γloop > import model sm
Running command import model sm
Loading built-in JSON model sm-default
The following 11 external parameters were forced to zero by the restriction card:
lamWS, AWS, rhoWS, etaWS, ymc, yme, ymm, MC, Me, MM, WTau
The following 34 vertex rules were removed by the restriction card:
V_82 → (c~, d, G+) | V_83 → (t~, d, G+) | V_84 → (c~, s, G+) | V_85 → (t~, s, G+) | V_86 → (u~, b, G+) | V_87 → (c~, b, G+) | V_90 → (c~, d, W+) | V_91 → (t~, d,
W+) | V_92 → (u~, s, W+) | V_94 → (t~, s, W+) | V_95 → (u~, b, W+) | V_96 → (c~, b, W+) | V_101 → (e+, e-, G0) | V_102 → (mu+, mu-, G0) | V_104 → (e+, e-, H) |
V_105 → (mu+, mu-, H) | V_110 → (ve~, e-, G+) | V_111 → (vm~, mu-, G+) | V_116 → (b~, u, G-) | V_117 → (d~, c, G-) | V_118 → (s~, c, G-) | V_119 → (b~, c, G-) |
V_120 → (d~, t, G-) | V_121 → (s~, t, G-) | V_124 → (s~, u, W-) | V_125 → (b~, u, W-) | V_126 → (d~, c, W-) | V_128 → (b~, c, W-) | V_129 → (d~, t, W-) | V_130 →
(s~, t, W-) | V_138 → (c~, c, G0) | V_140 → (c~, c, H) | V_145 → (e+, ve, G-) | V_146 → (mu+, vm, G-)
The following 36 couplings were removed by the restriction card:
GC_13, GC_14, GC_16, GC_17, GC_18, GC_19, GC_20, GC_22, GC_23, GC_24, GC_25, GC_26, GC_28, GC_29, GC_42, GC_43, GC_44, GC_46, GC_47, GC_48, GC_84, GC_85, GC_86, GC_87,
GC_88, GC_89, GC_90, GC_91, GC_92, GC_93, GC_101, GC_102, GC_103, GC_105, GC_106, GC_107
aa_aa_2L | γloop >
generate amp a a > a a | a t t~ g ghG ghG~ QCD==2 QED==4 [{2}] --symmetrize-initial-states=true --symmetrize-final-states=true --numerator-grouping no_grouping --only-
diagrams
Running command
generate amp a a > a a | a t t~ g ghG ghG~ QCD==2 QED==4 [{2}] --symmetrize-initial-states=true --symmetrize-final-states=true --numerator-grouping no_grouping --only-
diagrams
0s      | Δ=0s      | Starting Feynman graph generation with Symbolica ...
0s      | Δ=0s      | Number of graphs resulting from Symbolica generation:                                    192
0s      | Δ=0s      | Number of graphs retained after removal of vetoed topologies:                           144
0s      | Δ=0s      | Number of graphs after vertex info assignment:                                           144
0s      | Δ=0s      | Number of graphs after all complete graphs filters are applied:                          72
0s      | Δ=0s      | Number of graphs after closed fermion chains analysis:                                   72
0s      | Δ=0s      | Number of graphs after symmetrization of external states:                                20
0s      | Δ=0s      | Number of graphs after canonization of vertices, edges and flows (graph labels assigned here): 20
0s      | Δ=0s      | Number of graphs after numerator-aware grouping with strategy 'no grouping':            20 (13 isomorphically unique graphs, 0 color zero, 0
 lorentz zero, 0 grouped and 0 cancellation)
( Sum of the symmetry factors from each graph generated = -48 )
Generated amplitude 20 graphs.
Only diagram generation was requested, skipping integrand generation. You can generate integrands later using the 'generate existing <options>' command.
```

feyngen



Graph input

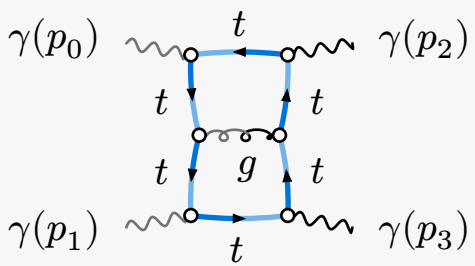
Graphviz dot format is standard, we should use it!

```
digraph GL16{
  projector = "ϵ(0,mink(4,hedge(0))) *ϵ(1,mink(4,hedge(1)))
*ϵbar(2,mink(4,hedge(2))) *ϵbar(3,mink(4,hedge(3)))";

  ext  [style=invis];
  ext -> 3  [dir=none particle="a"];
  ext -> 2  [dir=none particle="a"];
  5 -> ext  [dir=none particle="a"];
  4 -> ext  [dir=none particle="a"];
  0 -> 1    [dir=none lmb_id=0 particle="g"];
  0 -> 4    [lmb_id=1 particle="t"];
  5 -> 0    [particle="t"];
  1 -> 2    [particle="t"];
  3 -> 1    [particle="t"];
  2 -> 5    [particle="t"];
  4 -> 3    [particle="t"];
}
```

Graph input

Graphviz dot format is standard, we should use it!



$\Rightarrow$ `linnet` crate:

- allows graph parsing and mannipulation from python, rust and typst
- works with subgraphs natively

Generating Numerators

A graph + a UFO model gives you all the information needed to dress the graph with ***localized*** numerators.

```
digraph {
  0 [num="UFO::GC_11 *gamma(bis(4,hedge(6)),bis(4,hedge(9)),mink(4,hedge(4)))
*t(coad(8,hedge(4)),cof(3,hedge(9)),dind(cof(3,hedge(6))))"];
  ...
  5 [num="UFO::GC_2 *g(cof(3,hedge(15)),dind(cof(3,hedge(8))))
*gamma(bis(4,hedge(8)),bis(4,hedge(15)),mink(4,hedge(2)))"];
  ...
  0 -> 1   [... num="-1𝑖 *g(coad(8,hedge(4)),coad(8,hedge(5)))
*g(mink(4,hedge(4)),mink(4,hedge(5)))" particle="g"];
  ...
  5 -> 0   [num="(Q(6,mink(4,edge(6,1)))
*gamma(bis(4,hedge(9)),bis(4,hedge(8)),mink(4,edge(6,1)))+UFO::MT
*g(bis(4,hedge(8)),bis(4,hedge(9))))*1𝑖
*g(cof(3,hedge(8)),dind(cof(3,hedge(9))))" particle="t"];
 ...
}
```

Generating Numerators

Using a generic format for graphs mean we can use existing tools to visualize, parse and manipulate the graphs from different tools.

`python`, `mathematica`, `rust` and `c++` all support/ have libraries for `dot`

⇒ create a numerator from a graph by just multiplying the `num = "..."` fields

⇒ Need a CAS, we use `symbolica`

⇒ Need a tensor “vocabulary” → use `spenso`

What to do with numerators

Spenso

Spenso is a generic tensor network computation orchestrator, that defines the tensor vocabulary of `symbolica`

- Given a syntax like `gamma(bis(4,1),bis(4,2),mink(4,3))` `spenso` can understand that this ***represents*** a tensor
- And valid tensorial expression can be parsed into a network
- Values can be “filled-in”/ concretized

Try it!

Spenso for our example

```
1𝑖·GC_2⁴·GC_11²
    ·(MT·g^(b₄ b₅)+q₄(αᵉ⁴.¹)·γ(αᵉ⁴.¹,b₅ b₄))
    ·(MT·g^(b₆ b₁₃)
        +q₅(αᵉ⁵.¹)·γ(αᵉ⁵.¹,b₁₃ b₆)
    )·(MT·g^(b₇ b₁₄)
        +q₉(αᵉ⁹.¹)·γ(αᵉ⁹.¹,b₇ b₁₄)
    )·(MT·g^(b₈ b₁₂)
        +q₈(αᵉ⁸.¹)·γ(αᵉ⁸.¹,b₁₂ b₈)
    )·(MT·g^(b₉ b₁₇)
        +q₆(αᵉ⁶.¹)·γ(αᵉ⁶.¹,b₉ b₁₇)
    )·(MT·g^(b₁₀ b₁₁)
        +q₇(αᵉ⁷.¹)·γ(αᵉ⁷.¹,b₁₁ b₁₀)
    )·ϵ₀(α₀)·ϵ₁(α₁)·ϵ̄₂(α₂)·ϵ̄₃(α₃)
    ·γ(α₂,b₄ b₇)·γ(α₀,b₆ b₁₁)
    ·γ(α₃,b₈ b₅)·γ(α₁,b₁₀ b₉)
    ·γ(α₁₅,b₁₄ b₁₃)
    ·γ(α₁₆,b₁₇ b₁₂)
    ·t(c₁₅ Γ₁₃)(Γ₁₄)
    ·t(c₁₆ Γ₁₂)(Γ₁₇)·g^(α₁₅ α₁₆)
    ·g^(c₁₅ c₁₆)·δ_(Γ₅)^(Γ₄)
    ·δ_(Γ₈)^(Γ₅)·δ_(Γ₁₃)^(Γ₆)
    ·δ_(Γ₄)^(Γ₇)·δ_(Γ₁₂)^(Γ₈)
    ·δ_(Γ₁₀)^(Γ₉)·δ_(Γ₁₁)^(Γ₁₀)
    ·δ_(Γ₆)^(Γ₁₁)·δ_(Γ₇)^(Γ₁₄)
    ·δ_(Γ₉)^(Γ₁₇)
```

Idenso

The more common way of dealing with numerators is algebraically.

For example

$$\gamma^{\mu}_{ij}\gamma^{\nu}_{jk} + \gamma^{\nu}_{ij}\gamma^{\mu}_{jk} = 2\eta^{\mu\nu}\eta_{ik},$$

for gamma algebra, allows reducing

$$\mathrm{tr}(\gamma ... \gamma) \to \sum\prod \eta \to \sum\prod p_i \cdot p_j$$

`idenso` essentially implements the same replacement rules as `tracen` and `color.h` in form.

Comparison

method	byte size	number of sums	number of products	time to execute	time to optimize	time to evaluate
`spenso`	198161	413	328	2.952ms	54.32ms	1.17µs
`idenso`	1910344	2593	1834	617.2ms	866.13ms	6.75µs

Dealing with divergences

Local divergence subtraction

Direct numerical integration of feynman integrals requires addressing **all** divergences locally.

- Ultra-Violet ($k \to \infty$)
- Infra-Red ($k \to 0$)
- Threshold ($k \to \text{on-shell}$)

Local divergence subtraction

Direct numerical integration of feynman integrals requires addressing **all** divergences locally.

- Ultra-Violet ($k \to \infty$) using BPHZ/ R-Operation
- Infra-Red ($k \to 0$)
- Threshold ($k \to \text{on-shell}$)

Local divergence subtraction

Direct numerical integration of feynman integrals requires addressing **all** divergences locally.

- Ultra-Violet ($k \to \infty$) using BPHZ/ R-Operation
- Infra-Red ($k \to 0$) using KLN
- Threshold ($k \to \text{on-shell}$)

Local divergence subtraction

Direct numerical integration of feynman integrals requires addressing **all** divergences locally.

- Ultra-Violet ($k \to \infty$) using BPHZ/ R-Operation
- Infra-Red ($k \to 0$) using KLN
- Threshold ($k \to \text{on-shell}$) using Threshold subtraction

Local divergence subtraction

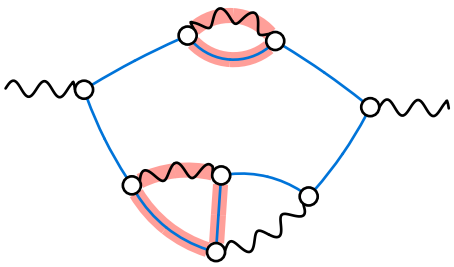
Direct numerical integration of feynman integrals requires addressing **all** divergences locally.

- Ultra-Violet ($k \to \infty$) using BPHZ/ R-Operation
- Infra-Red ($k \to 0$) using KLN
- Threshold ($k \to \text{on-shell}$) using Threshold subtraction

Lets have a look at the UV

Spinneys and Wood

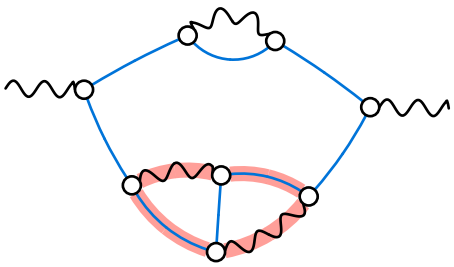
Identify divergent part of integrand/diagram/graph

$\Gamma =$ 

- many choices of loop momenta
- loop momentum = cycle = integral

Spinneys and Wood

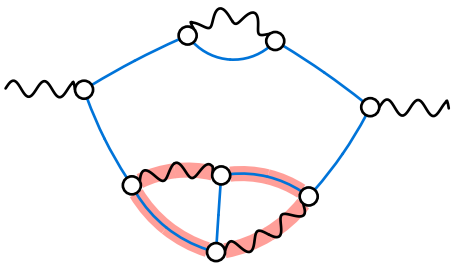
Identify divergent part of integrand/diagram/graph

$\Gamma =$ 

- many choices of loop momenta
- loop momentum = cycle = integral

Spinneys and Wood

Identify divergent part of integrand/diagram/graph

$\Gamma =$ 

- many choices of loop momenta
- loop momentum = cycle = integral
- if coherent rescaling of any such subgraph yeilds a UV singularity (DOD≥ 0) call it a ***spinney*** and collect all such subgraphs into the ***wood*** $\mathrm{W}(\Gamma)$

R - Operation

Given a graph Γ BPHZ[1] tells us how to subtract all UV divergences:

$$\mathrm{R}[\Gamma] = \sum_{S \in \mathrm{W}(\Gamma)} \Gamma \setminus S \star \prod_{\gamma \in S} \mathrm{Z}[\gamma]$$

all UV-limits + $\varnothing$ — singular behavior

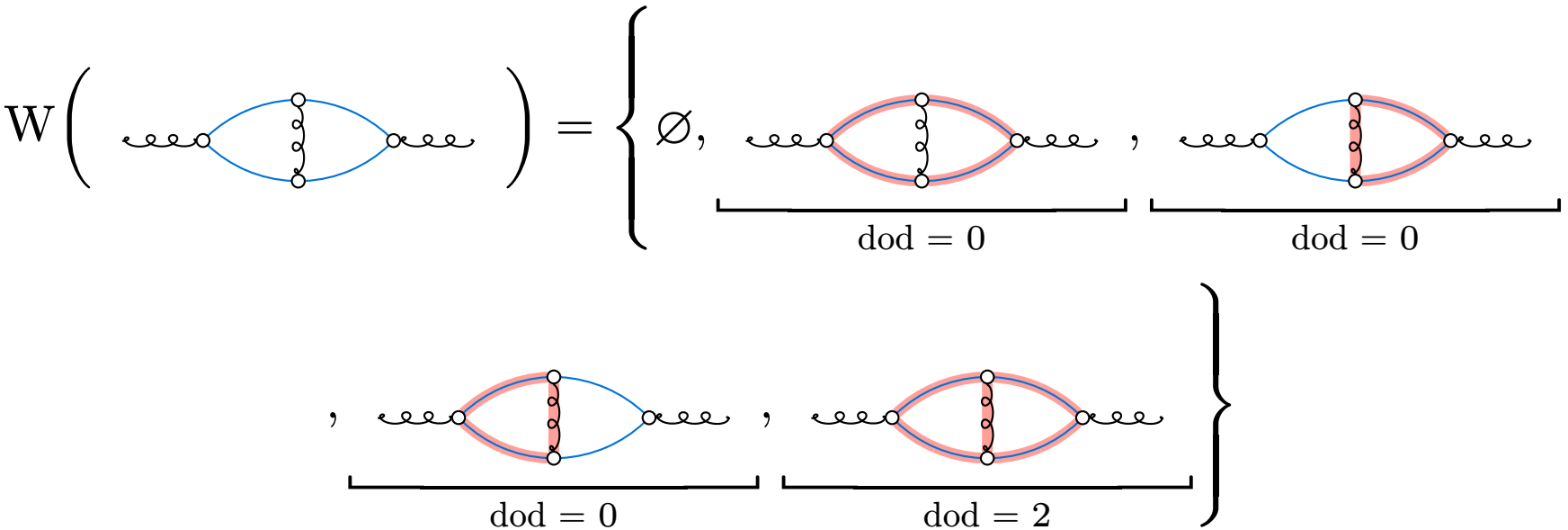
where

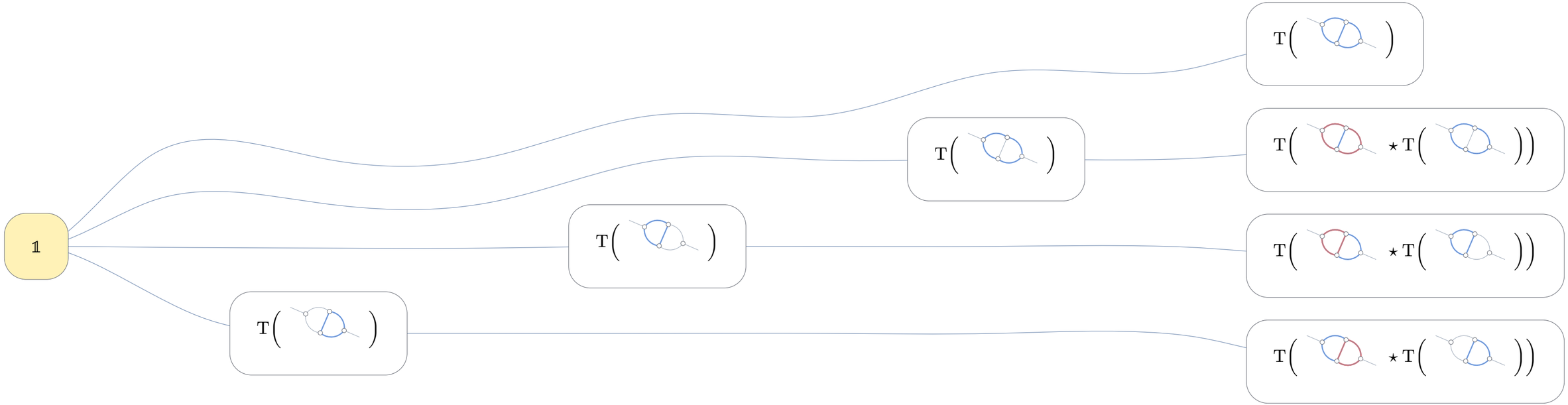
$$\gamma \text{ contribution to the renomalization CT Z} \longrightarrow \mathrm{Z}[\gamma] = -\mathrm{K}\left(\sum_{S \in \mathrm{W}(\gamma) \setminus \gamma} \gamma \setminus S \star \prod_{\tilde{\gamma} \in S} \mathrm{Z}[\tilde{\gamma}] \right)$$

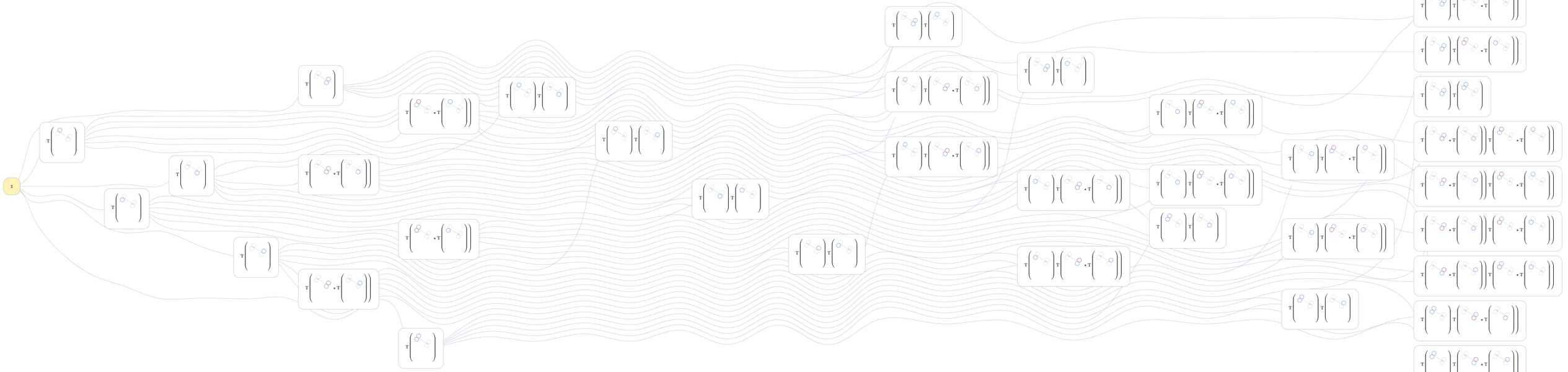
UV approximation that faithfully reproduces UV limit of its arguments

[1] Bogoliubov and Parasiuk 1957, Hepp 1966, Zimmermann 1969

R-Operation example




𝟙
T( )
T( )
T( )
T( )
T(⋆T( ))
T(⋆T( ))
T(⋆T( ))




LOOP demo