MaRTIn – Massive Recursive Tensor Integration

Tom Steudtner

in collaboration with
Joachim Brod, Lorenz Hüdepohl,
Emmanuel Stamou

Comput.Phys.Commun. 306 (2025) 109372 [2401.04033]

FORM Developer's Workshop 2025

13.06.2025 U Liverpool

technische universität
dortmund

Do we need another loop code?

» tools exist for many tasks

diagram generation, expansions, partial fractioning,
tensor reduction, IBP reductions, …

Do we need another loop code?

» tools exist for many tasks

diagram generation, expansions, partial fractioning,
tensor reduction, IBP reductions, …

» few completely public pipelines

» all-in-one package, geared towards complicated problems

Introducing MaRTIn

» initial release: Comput.Phys.Commun. 306 (2025) 109372 [2401.04033]

» try it (GPL v3): https://gitlab.com/manstam/martin

add-on (nicer graphs): https://gitlab.com/manstam/richard_draw

» here: current development status

Scope

» any relativistic QFT: renormalisable & non-renormalisable

» any n-point functions

Scope

» any relativistic QFT: renormalisable & non-renormalisable

» any n-point functions

» all-in-one package

» flexible, extensible

Scope

» any relativistic QFT: renormalisable & non-renormalisable

» any n-point functions

» all-in-one package

» flexible, extensible

» parallel: currently each diagram

» user-defined models

Current main modes

external momentum expansion

$$\frac{1}{(p+q)^2 - M^2} = \frac{1}{p^2 - M^2} - \frac{2p \cdot q + q^2}{[p^2 - M^2]^2} + \dots$$

» e.g. matching, eff. potentials

» 2-loop, any mass / massless

[Davydychev, Tausk (1993)] [Bobeth et al. (1999)]

Current main modes

external momentum expansion

$$\frac{1}{(p+q)^2 - M^2} = \frac{1}{p^2 - M^2} - \frac{2p\cdot q + q^2}{[p^2 - M^2]^2} + \dots$$

» e.g. matching, eff. potentials

» 2-loop, any mass / massless

[Davydychev, Tausk (1993)] [Bobeth et al. (1999)]

IR rearrangement with mass [Chetyrkin et al. (1997)]

$$\frac{1}{(p+q)^2} = \frac{1}{p^2 - m^2} - \frac{2p\cdot q + q^2 + m^2}{p^2 - m^2}\frac{1}{(p+q)^2}$$

» e.g. RGEs

» 3-loop, single IRA mass

[Broadhurst (1992)] [Broadhurst (1998)]
[Lee (2012-13)]

Special Design Goals

» careful treatment of γ_5

NDR, semi-NDR, Larin scheme, ‘t Hooft-Veltman

» both $d = 4$ and $d = 3$

» polyratfun for masses

» systematic ε- expansion

Technical Requirements

» prerequisits:

- `QGRAF`
- `FORM`
- `make`
- `python3` wrapper

```
$ martin problem-file.dat target
```

Technical Requirements

» prerequisits:

- `QGRAF`
- `FORM`
- `make`
- `python3` wrapper

```
$ martin problem-file.dat target
```

» input ⟶ output

- model files
- problem file

- `FORM` results
- `Mathematica` results
- graphic diagrams

Installation

» MaRTIn and QGRAF source directories

» config file `~/.martin.conf` (paths, CPUs, ...)

» working directory `martin-user/`

`models/`	
`qgraf/`	
`form/`	model files
`rdraw/`	
`prc/`	user-defined routines
`problems/`	problem files `loop.`*`process`*`.dat`
`results/`	results, populated by MaRTIn

Problem File

```
*--#[ QGRAF :

  model = "qmodel.prop.lag";
  model = "qmodel.vrtx.lag";

  in    = field1[q1];
  out   = field2[q1];

  loops = 2;
  loop_momentum = p;
  options       = onepi;

*--#] QGRAF :
```

Problem File

```
*--#[ QGRAF :

  model = "qmodel.prop.lag";
  model = "qmodel.vrtx.lag";

  in    = field1[q1];
  out   = field2[q1];

  loops = 2;
  loop_momentum = p;
  options       = onepi;

*--#] QGRAF :
```

```
*--#[ MAIN :

  #define NM "1"
  #define MODEL1 "form-model"

  #define IRA "2"
* or EXPDENO "2"

  #define DSCHEME "HV"
* "NDR","sNDR","HV","LARIN"

  #define FINALEPLIM "1"

*  #define PRINT
*--#] MAIN :
```

Problem File

```
*--#[ QGRAF :

  model = "qmodel.prop.lag";
  model = "qmodel.vrtx.lag";

  in    = field1[q1];
  out   = field2[q1];

  loops = 2;
  loop_momentum = p;
  options       = onepi;

*--#] QGRAF :
```

```
*--#[ MAIN :

  #define NM "1"
  #define MODEL1 "form-model"

  #define IRA "2"
* or EXPDENO "2"

  #define DSCHEME "HV"
* "NDR","sNDR","HV","LARIN"

  #define FINALEPLIM "1"

*  #define PRINT
*--#] MAIN :
```

code injection folds

```
USERDEF, TEST,
FOLD1, …, FOLD5
```

Problem File

```
*--#[ QGRAF :

  model = "qmodel.prop.lag";
  model = "qmodel.vrtx.lag";

  in    = field1[q1];
  out   = field2[q1];

  loops = 2;
  loop_momentum = p;
  options       = onepi;

*--#] QGRAF :
```

```
*--#[ MAIN :

  #define NM "1"
  #define MODEL1 "form-model"

  #define IRA "2"
* or EXPDENO "2"

  #define DSCHEME "HV"
* "NDR","sNDR","HV","LARIN"

  #define FINALEPLIM "1"

*  #define PRINT
*--#] MAIN :
```

code injection folds

```
USERDEF, TEST,
FOLD1, …, FOLD5
```

mass ordering

```
*--#[ MASSES :
  M1,M2,M3,

*--#] MASSES :
```

Problem File

```
*--#[ QGRAF :

  model = "qmodel.prop.lag";
  model = "qmodel.vrtx.lag";

  in    = field1[q1];
  out   = field2[q1];

  loops = 2;
  loop_momentum = p;
  options       = onepi;

*--#] QGRAF :
```

```
*--#[ MAIN :

  #define NM "1"
  #define MODEL1 "form-model"

  #define IRA "2"
* or EXPDENO "2"

  #define DSCHEME "HV"
* "NDR","sNDR","HV","LARIN"

  #define FINALEPLIM "1"

*  #define PRINT
*--#] MAIN :
```

code injection folds

```
USERDEF, TEST,
FOLD1, …, FOLD5
```

mass ordering

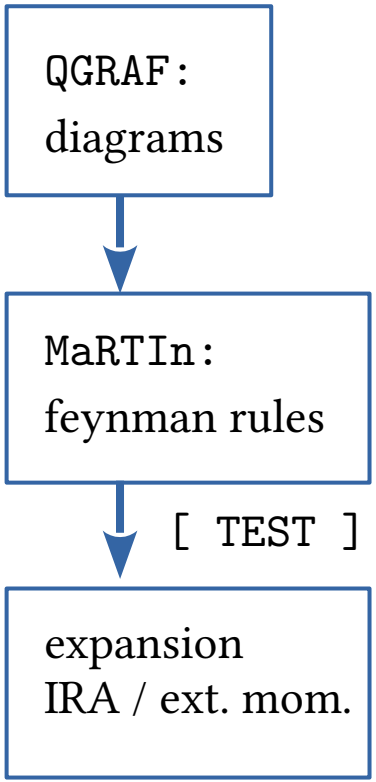
```
*--#[ MASSES :
  M1,M2,M3,

*--#] MASSES :
```

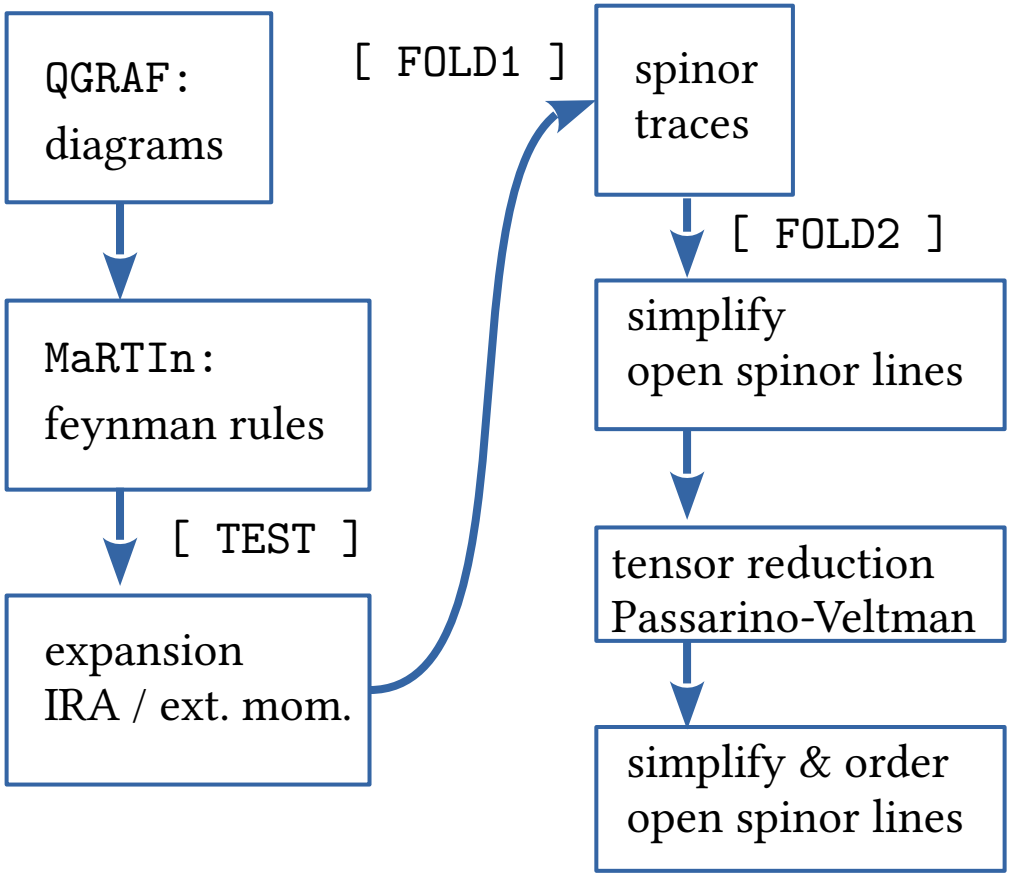
terminal output

```
*--#[ PRINT :
  bracket ep;
  print +s;
*--#] PRINT :
```

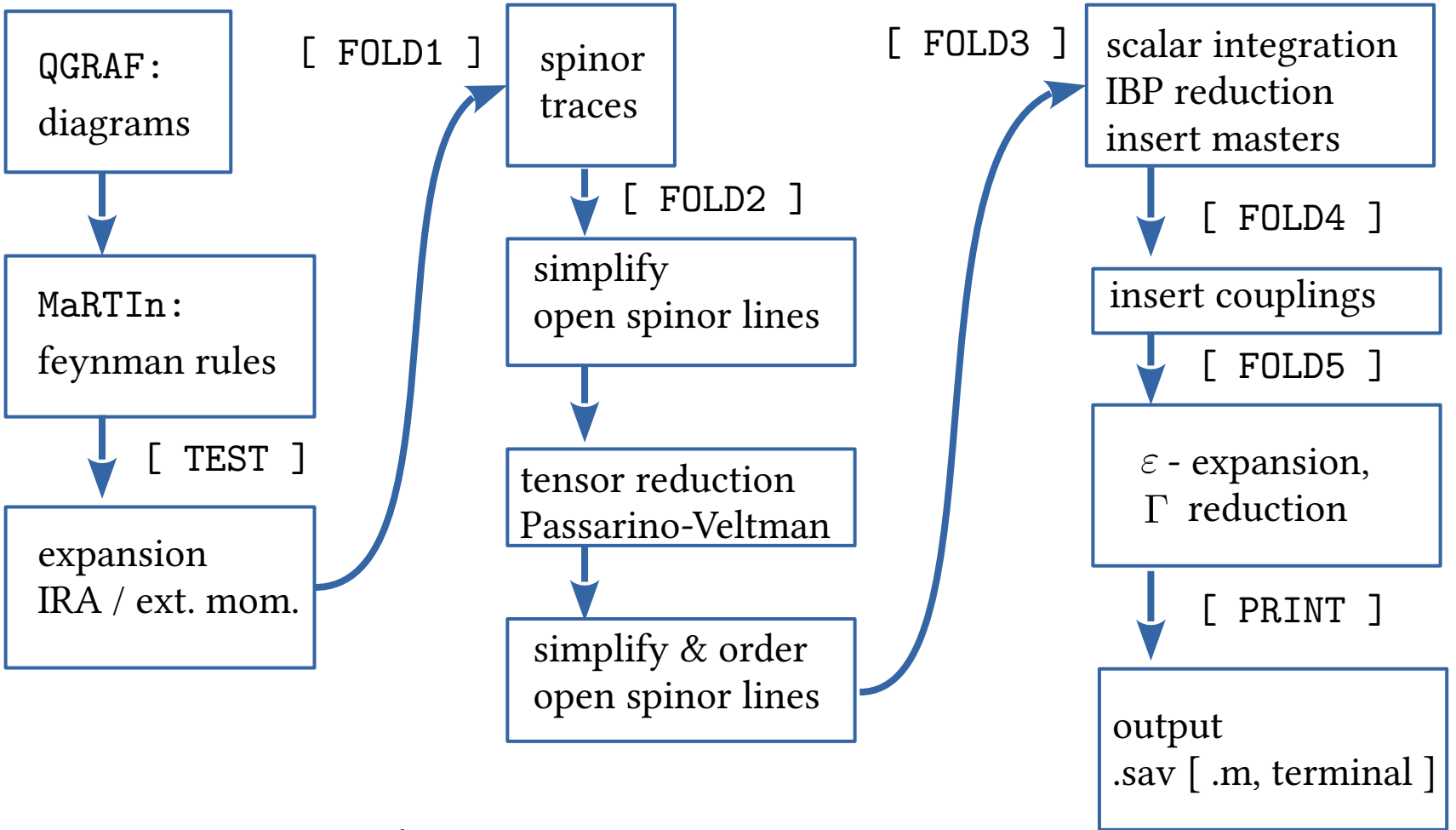
MaRTIn run


QGRAF:
diagrams
MaRTIn:
feynman rules
[TEST]
expansion
IRA / ext. mom.

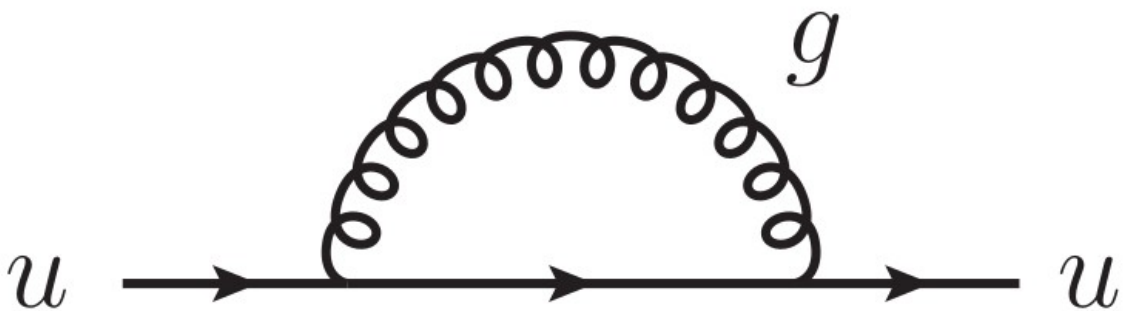
MaRTIn run


QGRAF:
diagrams
MaRTIn:
feynman rules
[TEST]
expansion
IRA / ext. mom.
[FOLD1]
spinor
traces
[FOLD2]
simplify
open spinor lines
tensor reduction
Passarino-Veltman
simplify & order
open spinor lines

MaRTIn run


QGRAF:
diagrams
MaRTIn:
feynman rules
[TEST]
expansion
IRA / ext. mom.
[FOLD1]
spinor
traces
[FOLD2]
simplify
open spinor lines
tensor reduction
Passarino-Veltman
simplify & order
open spinor lines
[FOLD3]
scalar integration
IBP reduction
insert masters
[FOLD4]
insert couplings
[FOLD5]
ε - expansion,
Γ reduction
[PRINT]
output
.sav [.m, terminal]

Example: Quark Self-Energy



Example: Quark Self-Energy

```
*--#[ QGRAF:
 model = 'sm.prop.lag';
 model = 'sm.vrtx.lag';
 in  = fu[q1];
 out = fu[q1];
 loops = 1;
 loop_momentum = p;
 options =;
 true = iprop[g, 1, 1];

*--#] QGRAF:
*--#[ MAIN:

 #define FINALEPLIM "-1"
 #define NM "1"
 #define MODEL1 "SM"
 #define GAUGEG "gaugeg"
 #define IRA "1"
 #define DSCHEME "NDR"

*--#] MAIN:
```

Example: Quark Self-Energy

```
*--#[ QGRAF:
 model = 'sm.prop.lag';
 model = 'sm.vrtx.lag';
 in  = fu[q1];
 out = fu[q1];
 loops = 1;
 loop_momentum = p;
 options =;
 true = iprop[g, 1, 1];

*--#] QGRAF:
*--#[ MAIN:

 #define FINALEPLIM "-1"
 #define NM "1"
 #define MODEL1 "SM"
 #define GAUGEG "gaugeg"
 #define IRA "1"
 #define DSCHEME "NDR"

*--#] MAIN:
```

run:

```
martin problems/SM/loop.1_uu.dat dia1
```

result:

```
FORM 5.0.0-beta.1 (Mar  7 2025, v5.0.0-beta.1-122-g638f84b)  Run: Sun Jun  8 11:16:45 2025
    #-

  dia1 =

      + ep^-1 * (
         + rat( - xiqg*Mu - 3*Mu,4)*FLine(ubarsp,fu,su3col,J2,mom,q1,dirac,
        one,usp,fu,su3col,J2,mom,q1)*i_*pi^-1*alphas*CF
         + rat(xiqg,4)*FLine(ubarsp,fu,su3col,J2,mom,q1,dirac,q1,usp,fu,
        su3col,J2,mom,q1)*i_*pi^-1*alphas*CF
         );

 0.55 sec out of 0.57 sec
```

$$\texttt{dia1} = -\frac{i\alpha_s}{4\pi\varepsilon} C_F \bar{u}(\mathbf{q}_1, J_2) \left[m_u(3+\xi) - \xi \not{q}_1 \right] u(\mathbf{q}_1, J_2)$$

How to build a model

1. `QGRAF` model

2. `FORM` model

3. `richard_draw` information

How to build a model

1. QGRAF model

```
[ quark, antiquark, - ; pfunct='Qprop', m='Mq' ]
[ antiquark, quark, gluon ; vfunct='Ffg' ]
```

2. FORM model

3. richard_draw information

How to build a model

1. `QGRAF` model

```
[ quark, antiquark, - ; pfunct='Qprop', m='Mq' ]
[ antiquark, quark, gluon ; vfunct='Ffg' ]
```

2. `FORM` model

   implement all propagators, vertices

3. `richard_draw` information

How to build a model

1. `QGRAF` model

```
[ quark, antiquark, - ; pfunct='Qprop', m='Mq' ]
[ antiquark, quark, gluon ; vfunct='Ffg' ]
```

2. `FORM` model

   implement all propagators, vertices

3. `richard_draw` information

```
{
  "fields": {
    "quark": ["", "fermion"],
    "antiquark": ["", "anti fermion"],  ...
  }
}
```

FORM model (Overview)

Fold `INSERTPROPAGATORS`

```
                                                                    #include docolour
id Dg(xx1?,xx2?,lorentz,nu1?,nu2?,mom,p?,mass,M?,field,fname?) =
   -i_*Deno(p,0)*( d_(nu1,nu2) - [1-xi]*p(nu1)*p(nu2)*Deno(p,0) )*Dadcol(xx1,xx2);
```

FORM model (Overview)

Fold INSERTPROPAGATORS

```
#include docolour
id Dg(xx1?,xx2?,lorentz,nu1?,nu2?,mom,p?,mass,M?,field,fname?) =
   -i_*Deno(p,0)*( d_(nu1,nu2) - [1-xi]*p(nu1)*p(nu2)*Deno(p,0) )*Dadcol(xx1,xx2);
```

Fold INSERTVERTICES

```
id VUug(xx1?,xx2?,xx3?,lorentz,nu1?,nu2?,nu3?,mom,v1?,v2?,v3?) =
   Fcol(xx1,xx3,xx2)*(-v1(nu3))*vUug;
```

FORM model (Overview)

Fold INSERTPROPAGATORS

```
#include docolour
id Dg(xx1?,xx2?,lorentz,nu1?,nu2?,mom,p?,mass,M?,field,fname?) =
   -i_*Deno(p,0)*( d_(nu1,nu2) - [1-xi]*p(nu1)*p(nu2)*Deno(p,0) )*Dadcol(xx1,xx2);
```

Fold INSERTVERTICES

```
id VUug(xx1?,xx2?,xx3?,lorentz,nu1?,nu2?,nu3?,mom,v1?,v2?,v3?) =
   Fcol(xx1,xx3,xx2)*(-v1(nu3))*vUug;
```

Special care with Fermions

```
*--#[ FF :
  Fqqg, ...
*--#] FF :

*--#[ VERTICES :
  vqqg, ...
*--#] VERTICES :
```

FORM model (Overview)

Fold INSERTPROPAGATORS

```
#include docolour
id Dg(xx1?,xx2?,lorentz,nu1?,nu2?,mom,p?,mass,M?,field,fname?) =
   -i_*Deno(p,0)*( d_(nu1,nu2) - [1-xi]*p(nu1)*p(nu2)*Deno(p,0) )*Dadcol(xx1,xx2);
```

Fold INSERTVERTICES

```
id VUug(xx1?,xx2?,xx3?,lorentz,nu1?,nu2?,nu3?,mom,v1?,v2?,v3?) =
   Fcol(xx1,xx3,xx2)*(-v1(nu3))*vUug;
```

Special care with Fermions

```
*--#[ FF :
  Fqqg, ...
*--#] FF :
```

```
*--#[ INSERTFERMIONVERTICES :
  id FL(fli?,?a,vqqg,lorentz,uu1?,mom,v1?,v2?,v3?) = FL(fli,?a)*g_(fli,uu1);
*--#] INSERTFERMIONVERTICES :
```

```
*--#[ VERTICES :
  vqqg, ...
*--#] VERTICES :
```

FORM model (Overview)

Fold `INSERTPROPAGATORS`

```
#include docolour
id Dg(xx1?,xx2?,lorentz,nu1?,nu2?,mom,p?,mass,M?,field,fname?) =
   -i_*Deno(p,0)*( d_(nu1,nu2) - [1-xi]*p(nu1)*p(nu2)*Deno(p,0) )*Dadcol(xx1,xx2);
```

Fold `INSERTVERTICES`

```
id VUug(xx1?,xx2?,xx3?,lorentz,nu1?,nu2?,nu3?,mom,v1?,v2?,v3?) =
   Fcol(xx1,xx3,xx2)*(-v1(nu3))*vUug;
```

Special care with Fermions

```
*--#[ FF :
  Fqqg, ...
*--#] FF :
```

```
*--#[ INSERTFERMIONVERTICES :
  id FL(fli?,?a,vqqg,lorentz,uu1?,mom,v1?,v2?,v3?) = FL(fli,?a)*g_(fli,uu1);
*--#] INSERTFERMIONVERTICES :
```

```
*--#[ VERTICES :
  vqqg, ...
*--#] VERTICES :
```

insert couplings:

```
*--#[ INSERTCOUPLINGS :
  id vqqg = i_*gs;
  id vUug = -gs;
*--#] INSERTCOUPLINGS :
```

Summary / Outlook

» characteristics: all-in-one, versatile, independent

» public version update soonish

⟶ performance increase, 3L IRA, …

» further extension planned

⟶ 4L IRA via `FMFT` [Pikelner (2017)]

⟶ 3L with generic masses