Automated Matrix Element Generation

Thorsten Ohl

Institute for Theoretical Physics and Astrophysics, University of Würzburg

Terascale Monte Carlo School 2009
DESY
April 20th, 2009

Contents

- A computer program implementing the function

$$(\mathcal{L}, \{\text{incoming}\}, \{\text{outgoing}\}) \mapsto \mathcal{M}(\alpha_1, \ldots; p_1, \ldots; s_1, \ldots)$$

where

- A computer program implementing the function

$$(\mathcal{L}, \{\text{incoming}\}, \{\text{outgoing}\}) \mapsto \mathcal{M}(\alpha_1, \ldots; p_1, \ldots; s_1, \ldots)$$

where

- $\mathcal{L}$: Lagrangian (or Feynman rules) of a model (SM, MSSM, ...)

- A computer program implementing the function

$$(\mathcal{L}, \{\text{incoming}\}, \{\text{outgoing}\}) \mapsto \mathcal{M}(\alpha_1, \ldots; p_1, \ldots; s_1, \ldots)$$

where

  - $\mathcal{L}$: Lagrangian (or Feynman rules) of a model (SM, MSSM, ...)
  - $\mathcal{M}(\alpha_1, \ldots; p_1, \ldots; s_1, \ldots)$: a function

$$\underbrace{\mathbf{R} \times \ldots \times \mathbf{R}}_{\text{masses, couplings, } \ldots} \times \underbrace{V^+ \times \ldots \times V^+}_{\text{4-momenta (forward light cones)}} \times \underbrace{\mathbf{Z} \times \ldots \times \mathbf{Z}}_{\text{helicities, colors}} \to \underbrace{\mathbf{C}}_{\text{amplitude}}$$

in a form that can be evaluated numerically, typically as C, C++ or Fortran code in that can be compiled and linked to Monte Carlo phase space integrators and generators

- A computer program implementing the function

$$(\mathcal{L}, \{\text{incoming}\}, \{\text{outgoing}\}) \mapsto \mathcal{M}(\alpha_1, \ldots; p_1, \ldots; s_1, \ldots)$$

where

- $\mathcal{L}$: Lagrangian (or Feynman rules) of a model (SM, MSSM, ...)
- $\mathcal{M}(\alpha_1, \ldots; p_1, \ldots; s_1, \ldots)$: a function

$$\underbrace{\mathbf{R} \times \ldots \times \mathbf{R}}_{\text{masses, couplings, \ldots}} \times \underbrace{V^+ \times \ldots \times V^+}_{\text{4-momenta (forward light cones)}} \times \underbrace{\mathbf{Z} \times \ldots \times \mathbf{Z}}_{\text{helicities, colors}} \to \underbrace{\mathbf{C}}_{\text{amplitude}}$$

in a form that can be evaluated numerically, typically as C, C++ or Fortran code in that can be compiled and linked to Monte Carlo phase space integrators and generators

- NB: in some cases only $\mathcal{L} \to \sum |\mathcal{M}(\alpha_1, \ldots; p_1, \ldots; s_1, \ldots \ldots)|^2$ is required. It is often better defined (infrared/collinear cancellations) and sometimes more compact (spin/polarization sums).

- A computer program implementing the function

$$(\mathcal{L}, \{\text{incoming}\}, \{\text{outgoing}\}) \mapsto \mathcal{M}(\alpha_1, \ldots; p_1, \ldots; s_1, \ldots)$$

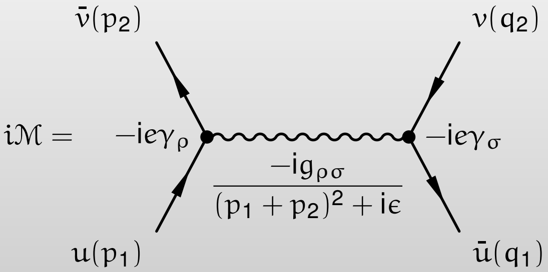
  where
  - $\mathcal{L}$: Lagrangian (or Feynman rules) of a model (SM, MSSM, ...)
  - $\mathcal{M}(\alpha_1, \ldots; p_1, \ldots; s_1, \ldots)$: a function

$$\underbrace{\mathbf{R} \times \ldots \times \mathbf{R}}_{\text{masses, couplings, ...}} \times \underbrace{V^+ \times \ldots \times V^+}_{\text{4-momenta (forward light cones)}} \times \underbrace{\mathbf{Z} \times \ldots \times \mathbf{Z}}_{\text{helicities, colors}} \to \underbrace{\mathbf{C}}_{\text{amplitude}}$$

    in a form that can be evaluated numerically, typically as C, C++ or Fortran code in that can be compiled and linked to Monte Carlo phase space integrators and generators
- NB: in some cases only $\mathcal{L} \to \sum |\mathcal{M}(\alpha_1, \ldots; p_1, \ldots; s_1, \ldots \ldots)|^2$ is required. It is often better defined (infrared/collinear cancellations) and sometimes more compact (spin/polarization sums).
- first robust and usable examples in the early 1990s: CompHEP, FeynArts, Grace, MadGraph, ...

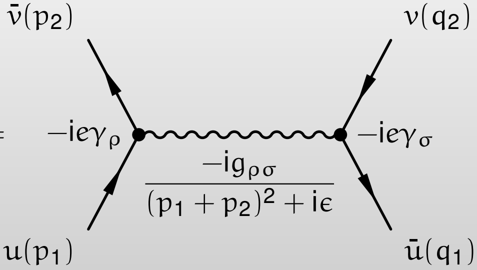
- for simplicity: $e^+e^- \to \mu^+\mu^-$ at PETRA (i. e. QED, mostly)

- for simplicity: $e^+e^- \to \mu^+\mu^-$ at PETRA (i. e. QED, mostly)
- just one Feynman diagram



- for simplicity: $e^+e^- \to \mu^+\mu^-$ at PETRA (i. e. QED, mostly)
- just one Feynman diagram

$i\mathcal{M} =$



- analytical expression

$$\begin{aligned} i\mathcal{M} &= \bar{v}(p_2)(-ie\gamma^\rho)u(p_1)\frac{-ig_{\rho\sigma}}{(p_1+p_2)^2+i\epsilon}\bar{u}(q_1)(-ie\gamma^\sigma)v(q_2) \\ &= ie^2\frac{1}{s}\left[\bar{v}(p_2)\gamma_\rho u(p_1)\right]\left[\bar{u}(q_1)\gamma^\rho v(q_2)\right] \end{aligned}$$

- e. g. command line for O'Mega to compute $e^-e^+ \to \mu^-\mu^+$ in QED

```
$ f90_QED -scatter "e- e+ -> m- m+"
```

- e. g. command line for O'Mega to compute $e^-e^+ \to \mu^-\mu^+$ in QED

  `$ f90_QED -scatter "e- e+ -> m- m+"`
- resulting Fortran95 code

```
pure function eleposmuoamu (k, s) result (amp)
  real(kind=omega_prec), dimension(0:,:), intent(in) :: k
  integer, dimension(:), intent(in) :: s
  complex(kind=omega_prec) :: amp
  type(momentum) :: p1, p2, p3, p4
  type(spinor) :: muo_4, ele_1
  type(conjspinor) :: amu_3, pos_2
  type(vector) :: gam_12
  type(momentum) :: p12
  p1 = - k(:,1) ! incoming e-
  p2 = - k(:,2) ! incoming e+
  p3 =   k(:,3) ! outgoing m-
  p4 =   k(:,4) ! outgoing m+
  ele_1 = u (mass(11), - p1, s(1))                    ! u_{s_1}(k_1)
  pos_2 = vbar (mass(11), - p2, s(2))                 ! \bar{v}_{s_2}(k_2)
  amu_3 = ubar (mass(13), p3, s(3))                   ! \bar{u}_{s_3}(k_3)
  muo_4 = v (mass(13), p4, s(4))                      ! v_{s_4}(k_4)
  p12 = p1 + p2
  gam_12 = pr_feynman(p12, + v_ff(qlep,pos_2,ele_1)) ! (1/s) e\bar{v}(k_2)\gamma_\mu u(k_1)
  amp = 0
  amp = amp + gam_12*( + v_ff(qlep,amu_3,muo_4))      ! (1/s) e\bar{v}(k_2)\gamma_\mu u(k_1) e\bar{u}(k_3)\gamma^\mu v(k_4)
  amp = - amp ! 2 vertices, 1 propagators
end function eleposmuoamu
```

(some additional interface routines suppressed)

- the usual rules for manual calculations are algorithmic

☺ the usual rules for manual calculations are algorithmic

∴ can be implemented in a computer program

- the usual rules for manual calculations are algorithmic
- $\therefore$ can be implemented in a computer program
  - for a given set of Feynman rules and external particles

- the usual rules for manual calculations are algorithmic
- ∴ can be implemented in a computer program
- ▶ for a given set of Feynman rules and external particles
  1. *draw all topologically inequivalent Feynman diagrams connecting the external particles (up to the chosen number of loops)*

- the usual rules for manual calculations are algorithmic

∴ can be implemented in a computer program

- for a given set of Feynman rules and external particles
    1. *draw all topologically inequivalent Feynman diagrams connecting the external particles (up to the chosen number of loops)*
    2. *in each diagram, assign a 4-momentum to each internal line according to 4-momentum conservation at each vertex*

☺ the usual rules for manual calculations are algorithmic

∴ can be implemented in a computer program

- for a given set of Feynman rules and external particles
  1. *draw all topologically inequivalent Feynman diagrams connecting the external particles (up to the chosen number of loops)*
  2. *in each diagram, assign a 4-momentum to each internal line according to 4-momentum conservation at each vertex*
  3. *in each diagram, replace each external line with a wave function according to the particle type and momentum*

☺ the usual rules for manual calculations are algorithmic

∴ can be implemented in a computer program

- for a given set of Feynman rules and external particles
  1. *draw all topologically inequivalent Feynman diagrams connecting the external particles (up to the chosen number of loops)*
  2. *in each diagram, assign a 4-momentum to each internal line according to 4-momentum conservation at each vertex*
  3. *in each diagram, replace each external line with a wave function according to the particle type and momentum*
  4. *in each diagram, replace each internal line with a propagator according to the particle type and momentum*

- the usual rules for manual calculations are algorithmic

∴ can be implemented in a computer program

- for a given set of Feynman rules and external particles
  1. *draw all topologically inequivalent Feynman diagrams connecting the external particles (up to the chosen number of loops)*
  2. *in each diagram, assign a 4-momentum to each internal line according to 4-momentum conservation at each vertex*
  3. *in each diagram, replace each external line with a wave function according to the particle type and momentum*
  4. *in each diagram, replace each internal line with a propagator according to the particle type and momentum*
  5. *for each diagram, replace each vertex with a vertex function according to Feynman rules*

- the usual rules for manual calculations are algorithmic
- ∴ can be implemented in a computer program
- for a given set of Feynman rules and external particles
  1. *draw all topologically inequivalent Feynman diagrams connecting the external particles (up to the chosen number of loops)*
  2. *in each diagram, assign a 4-momentum to each internal line according to 4-momentum conservation at each vertex*
  3. *in each diagram, replace each external line with a wave function according to the particle type and momentum*
  4. *in each diagram, replace each internal line with a propagator according to the particle type and momentum*
  5. *for each diagram, replace each vertex with a vertex function according to Feynman rules*
  6. *divide each diagram by the size of its automorphism group*

☺ the usual rules for manual calculations are algorithmic

∴ can be implemented in a computer program

- for a given set of Feynman rules and external particles
  1. *draw all topologically inequivalent Feynman diagrams connecting the external particles (up to the chosen number of loops)*
  2. *in each diagram, assign a 4-momentum to each internal line according to 4-momentum conservation at each vertex*
  3. *in each diagram, replace each external line with a wave function according to the particle type and momentum*
  4. *in each diagram, replace each internal line with a propagator according to the particle type and momentum*
  5. *for each diagram, replace each vertex with a vertex function according to Feynman rules*
  6. *divide each diagram by the size of its automorphism group*
  7. *multiply each diagram with a factor of $+1$ or -1 in order to make the sum of the diagrams symmetric/antisymmetric under the exchange of bosons/fermions*

- the usual rules for manual calculations are algorithmic
- ∴ can be implemented in a computer program
- for a given set of Feynman rules and external particles
  1. *draw all topologically inequivalent Feynman diagrams connecting the external particles (up to the chosen number of loops)*
  2. *in each diagram, assign a 4-momentum to each internal line according to 4-momentum conservation at each vertex*
  3. *in each diagram, replace each external line with a wave function according to the particle type and momentum*
  4. *in each diagram, replace each internal line with a propagator according to the particle type and momentum*
  5. *for each diagram, replace each vertex with a vertex function according to Feynman rules*
  6. *divide each diagram by the size of its automorphism group*
  7. *multiply each diagram with a factor of $+1$ or -1 in order to make the sum of the diagrams symmetric/antisymmetric under the exchange of bosons/fermions*
  8. *finally: add them up!*

- NB: familiar trace techniques for

$$\sum_{s_1,\ldots} |\mathcal{M}(\alpha_1,\ldots;p_1,\ldots,;s_1,\ldots\ldots)|^2$$

rarely used, because
  - tedious for polarized scattering (longer traces, due to spin projections)

- NB: familiar trace techniques for

$$\sum_{s_1,\ldots} |\mathcal{M}(\alpha_1,\ldots;p_1,\ldots,;s_1,\ldots\ldots)|^2$$

rarely used, because

  - tedious for polarized scattering (longer traces, due to spin projections)
  - intermediate expressions grow quadratically with the number of Feynman diagrams

$$\left|\sum_{i}^{N_D}\mathcal{M}_i\right|^2 = \sum_{i=1}^{N_D}\sum_{j=1}^{N_D}\mathcal{M}_i^*\mathcal{M}_j$$

- NB: familiar trace techniques for

$$\sum_{s_1,\ldots} |\mathcal{M}(\alpha_1,\ldots;p_1,\ldots,;s_1,\ldots\ldots)|^2$$

rarely used, because

  - tedious for polarized scattering (longer traces, due to spin projections)
  - intermediate expressions grow quadratically with the number of Feynman diagrams

$$\left|\sum_{i}^{N_D} \mathcal{M}_i\right|^2 = \sum_{i=1}^{N_D}\sum_{j=1}^{N_D} \mathcal{M}_i^* \mathcal{M}_j$$

    - number of polarization states grows with a power of the number of particles

- NB: familiar trace techniques for

$$\sum_{s_1,\ldots} |\mathcal{M}(\alpha_1,\ldots;p_1,\ldots,;s_1,\ldots\ldots)|^2$$

rarely used, because
  - tedious for polarized scattering (longer traces, due to spin projections)
  - intermediate expressions grow quadratically with the number of Feynman diagrams

$$\left|\sum_{i}^{N_D} \mathcal{M}_i\right|^2 = \sum_{i=1}^{N_D}\sum_{j=1}^{N_D} \mathcal{M}_i^* \mathcal{M}_j$$

    - number of polarization states grows with a power of the number of particles
    - number of Feynman diagrams grows with a factorial of the number of particles

- NB: familiar trace techniques for

$$\sum_{s_1,\ldots} |\mathcal{M}(\alpha_1,\ldots;p_1,\ldots,;s_1,\ldots\ldots)|^2$$

rarely used, because

  - tedious for polarized scattering (longer traces, due to spin projections)
  - intermediate expressions grow quadratically with the number of Feynman diagrams

$$\left|\sum_i^{N_D} \mathcal{M}_i\right|^2 = \sum_{i=1}^{N_D}\sum_{j=1}^{N_D} \mathcal{M}_i^* \mathcal{M}_j$$

    - number of polarization states grows with a power of the number of particles
    - number of Feynman diagrams grows with a factorial of the number of particles

    ∴ helicity amplitudes win (eventually)!

- 4 of the 8 steps are straightforward translations
  3. *in each diagram, replace each external line with a wave function according to the particle type and momentum*
  4. *in each diagram, replace each internal line with a propagator according to the particle type and momentum*
  5. *for each diagram, replace each vertex with a vertex function according to Feynman rules*
  8. *finally: add them up!*

- 4 of the 8 steps are straightforward translations
  3. *in each diagram, replace each external line with a wave function according to the particle type and momentum*
  4. *in each diagram, replace each internal line with a propagator according to the particle type and momentum*
  5. *for each diagram, replace each vertex with a vertex function according to Feynman rules*
  8. *finally: add them up!*

∴ big boring `case` statements, only problem: striking a balance between legibility (debugging!) and efficiency

- 4 of the 8 steps are straightforward translations
  3. *in each diagram, replace each external line with a wave function according to the particle type and momentum*
  4. *in each diagram, replace each internal line with a propagator according to the particle type and momentum*
  5. *for each diagram, replace each vertex with a vertex function according to Feynman rules*
  8. *finally: add them up!*

∴ big boring `case` statements, only problem: striking a balance between legibility (debugging!) and efficiency

- 1 of the 8 steps is straightforward only for tree diagrams
  2. *in each diagram, assign a 4-momentum to each internal line according to 4-momentum conservation at each vertex*

  in tree diagrams all 4-momenta are determined by 4-momentum conservation and have a unique representation as a linear combination of $n-1$ external momenta with coefficients $\{-1, 0, 1\}$.

- 4 of the 8 steps are straightforward translations
  3. *in each diagram, replace each external line with a wave function according to the particle type and momentum*
  4. *in each diagram, replace each internal line with a propagator according to the particle type and momentum*
  5. *for each diagram, replace each vertex with a vertex function according to Feynman rules*
  8. *finally: add them up!*

∴ big boring `case` statements, only problem: striking a balance between legibility (debugging!) and efficiency

- 1 of the 8 steps is straightforward only for tree diagrams
  2. *in each diagram, assign a 4-momentum to each internal line according to 4-momentum conservation at each vertex*

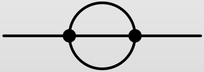
  in tree diagrams all 4-momenta are determined by 4-momentum conservation and have a unique representation as a linear combination of $n - 1$ external momenta with coefficients $\{-1, 0, 1\}$.
- in loop diagrams there are undermined loop momenta that must (usually) be integrated analytically: $\int d^4 p_i$

- the 3 remaining steps require non-trivial algorithms

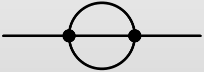
- the 3 remaining steps require non-trivial algorithms
  1. *draw all topologically inequivalent Feynman diagrams connecting the external particles (up to the chosen number of loops)*

- the 3 remaining steps require non-trivial algorithms
  1. *draw all topologically inequivalent Feynman diagrams connecting the external particles (up to the chosen number of loops)*
  6. *divide each diagram by the size of its automorphism group*

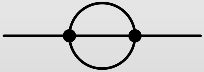
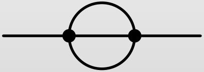
- the 3 remaining steps require non-trivial algorithms
  1. *draw all topologically inequivalent Feynman diagrams connecting the external particles (up to the chosen number of loops)*
  6. *divide each diagram by the size of its automorphism group*
     - e. g.



$$\to \frac{1}{3!} \cdot$$

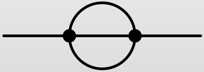
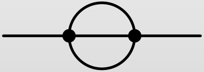


- the 3 remaining steps require non-trivial algorithms
  1. *draw all topologically inequivalent Feynman diagrams connecting the external particles (up to the chosen number of loops)*
  6. *divide each diagram by the size of its automorphism group*
     - e. g.

 $\rightarrow \frac{1}{3!} \cdot$ 

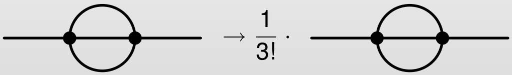
     - only $\neq 1$ for loop diagrams, no general polynomial algorithm

- the 3 remaining steps require non-trivial algorithms
  1. *draw all topologically inequivalent Feynman diagrams connecting the external particles (up to the chosen number of loops)*
  6. *divide each diagram by the size of its automorphism group*
     - e. g.


$$\to \frac{1}{3!} \cdot$$


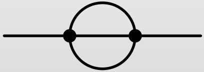
     - only $\neq 1$ for loop diagrams, no general polynomial algorithm
  7. *multiply each diagram with a factor of $+1$ or -1 in order to make the sum of the diagrams symmetric/antisymmetric under the exchange of bosons/fermions*

- the 3 remaining steps require non-trivial algorithms
  1. *draw all topologically inequivalent Feynman diagrams connecting the external particles (up to the chosen number of loops)*
  6. *divide each diagram by the size of its automorphism group*
     - e. g.

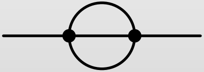


     - only $\neq 1$ for loop diagrams, no general polynomial algorithm
  7. *multiply each diagram with a factor of $+1$ or -1 in order to make the sum of the diagrams symmetric/antisymmetric under the exchange of bosons/fermions*
     - if all fermions are Dirac fermions, i. e. their lines can be followed through the diagram, use the sign of the permutation of the endpoints

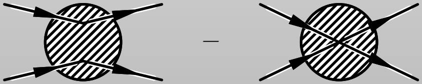
- the 3 remaining steps require non-trivial algorithms
  1. *draw all topologically inequivalent Feynman diagrams connecting the external particles (up to the chosen number of loops)*
  6. *divide each diagram by the size of its automorphism group*
     - e. g.



$$\to \frac{1}{3!} \cdot$$



     - only $\neq 1$ for loop diagrams, no general polynomial algorithm
  7. *multiply each diagram with a factor of $+1$ or -1 in order to make the sum of the diagrams symmetric/antisymmetric under the exchange of bosons/fermions*
     - if all fermions are Dirac fermions, i. e. their lines can be followed through the diagram, use the sign of the permutation of the endpoints



     - if Majorana fermions are present (MSSM!), use the clever algorithm of [Denner et al. Phys. Lett. **B291**, 278 (1992)]

- Challenge: generate each diagram exactly once

- Challenge: generate each diagram exactly once
- ideal representation: one-to-one correspondence with diagrams

- Challenge: generate each diagram exactly once
- ideal representation: one-to-one correspondence with diagrams

☹ impossible, in general

- Challenge: generate each diagram exactly once
- ideal representation: one-to-one correspondence with diagrams

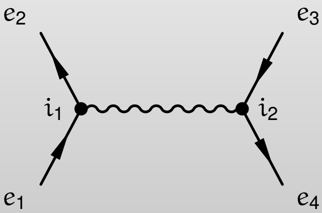
☹ impossible, in general

- choices

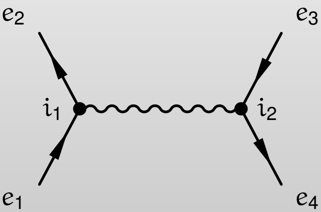
- Challenge: generate each diagram exactly once
- ideal representation: one-to-one correspondence with diagrams

☹ impossible, in general

- choices
   0. diagram (not machine readable!)



- Challenge: generate each diagram exactly once
- ideal representation: one-to-one correspondence with diagrams
- ☹ impossible, in general
- choices
  0. diagram (not machine readable!)



  1. sets of external and internal vertices and edges

$$\left(\begin{Bmatrix} (e_1, e^-, p_1) \\ (e_2, e^+, p_2) \\ (e_3, \mu^+, q_2) \\ (e_4, \mu^-, q_1) \end{Bmatrix}, \begin{Bmatrix} (i_1, -ie\gamma_\mu) \\ (i_2, -ie\gamma_\mu) \end{Bmatrix}, \begin{Bmatrix} (e_1, i_1, \{e^-\}) \\ (e_2, i_1, \{e^+\}) \\ (e_3, i_2, \{\mu^-\}) \\ (e_4, i_2, \{\mu^+\}) \\ (i_1, i_2, \{\gamma\}) \end{Bmatrix} \right)$$

- choices (cont'd)
  2. incidence matrix

$$\left\{\begin{array}{c}(e_1, i_1, \{e^-\}) \\ (e_2, i_1, \{e^+\}) \\ (e_3, i_2, \{\mu^-\}) \\ (e_4, i_2, \{\mu^+\}) \\ (i_1, i_2, \{\gamma\})\end{array}\right\} \sim \begin{array}{c|cccccc} & e_1 & e_2 & e_3 & e_4 & i_1 & i_2 \\ \hline e_1 & \emptyset & \emptyset & \emptyset & \emptyset & \{e^-\} & \emptyset \\ e_2 & \emptyset & \emptyset & \emptyset & \emptyset & \{e^+\} & \emptyset \\ e_3 & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \{\mu^-\} \\ e_4 & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \{\mu^+\} \\ i_1 & \{e^+\} & \{e^-\} & \emptyset & \emptyset & \emptyset & \{\gamma\} \\ i_2 & \emptyset & \emptyset & \{\mu^+\} & \{\mu^-\} & \{\gamma\} & \emptyset \end{array}$$

- choices (cont'd)
  2. incidence matrix

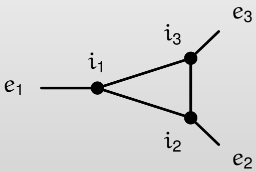
$$\left\{\begin{matrix}(e_1,i_1,\{e^-\})\\(e_2,i_1,\{e^+\})\\(e_3,i_2,\{\mu^-\})\\(e_4,i_2,\{\mu^+\})\\(i_1,i_2,\{\gamma\})\end{matrix}\right\} \sim \begin{array}{c|cccccc} & e_1 & e_2 & e_3 & e_4 & i_1 & i_2 \\ \hline e_1 & \emptyset & \emptyset & \emptyset & \emptyset & \{e^-\} & \emptyset \\ e_2 & \emptyset & \emptyset & \emptyset & \emptyset & \{e^+\} & \emptyset \\ e_3 & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \{\mu^-\} \\ e_4 & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \{\mu^+\} \\ i_1 & \{e^+\} & \{e^-\} & \emptyset & \emptyset & \emptyset & \{\gamma\} \\ i_2 & \emptyset & \emptyset & \{\mu^+\} & \{\mu^-\} & \{\gamma\} & \emptyset \end{array}$$

☹ different egde set and incidence matrix for same graph

$$\left\{\begin{matrix}(e_1,i_2,\{e^-\})\\(e_2,i_2,\{e^+\})\\(e_3,i_1,\{\mu^-\})\\(e_4,i_1,\{\mu^+\})\\(i_1,i_2,\{\gamma\})\end{matrix}\right\} \sim \begin{array}{c|cccccc} & e_1 & e_2 & e_3 & e_4 & i_1 & i_2 \\ \hline e_1 & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \{e^-\} \\ e_2 & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \{e^+\} \\ e_3 & \emptyset & \emptyset & \emptyset & \emptyset & \{\mu^-\} & \emptyset \\ e_4 & \emptyset & \emptyset & \emptyset & \emptyset & \{\mu^+\} & \emptyset \\ i_1 & \emptyset & \emptyset & \{\mu^+\} & \{\mu^-\} & \emptyset & \{\gamma\} \\ i_2 & \{e^+\} & \{e^-\} & \emptyset & \emptyset & \{\gamma\} & \emptyset \end{array}$$

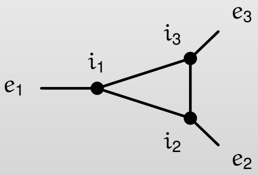
- ambiguity: external vertices distinguished by the momentum, internal vertices named ad-hoc

- ambiguity: external vertices distinguished by the momentum, internal vertices named ad-hoc
- solution: canonical ordering of internal vertices, e. g.



$$\sim \left\{ \begin{matrix} (e_1, i_1) \\ (e_2, i_2) \\ (e_3, i_3) \\ (i_1, i_2) \\ (i_2, i_3) \\ (i_3, i_1) \end{matrix} \right\} \sim \begin{array}{c|cccccc} & e_1 & e_2 & e_3 & i_1 & i_2 & i_3 \\ \hline e_1 & & & & 1 & & \\ e_2 & & & & & 1 & \\ e_3 & & & & & & 1 \\ i_1 & 1 & & & & 1 & \\ i_2 & & 1 & & & & 1 \\ i_3 & & & 1 & 1 & & \end{array}$$

- ambiguity: external vertices distinguished by the momentum, internal vertices named ad-hoc
- solution: canonical ordering of internal vertices, e. g.

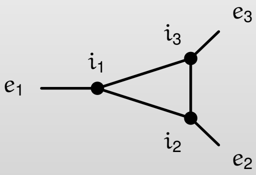


$$\sim \left\{ \begin{matrix} (e_1, i_1) \\ (e_2, i_2) \\ (e_3, i_3) \\ (i_1, i_2) \\ (i_2, i_3) \\ (i_3, i_1) \end{matrix} \right\} \sim$$

	e_1	e_2	e_3	i_1	i_2	i_3
e_1				1		
e_2					1	
e_3						1
i_1	1				1	
i_2		1				1
i_3			1	1		

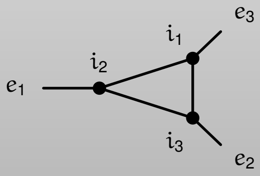
- rows/columns ordered lexicographically

- ambiguity: external vertices distinguished by the momentum, internal vertices named ad-hoc
- solution: canonical ordering of internal vertices, e. g.



$$\sim \begin{Bmatrix} (e_1, i_1) \\ (e_2, i_2) \\ (e_3, i_3) \\ (i_1, i_2) \\ (i_2, i_3) \\ (i_3, i_1) \end{Bmatrix} \sim \begin{array}{c|cccccc} & e_1 & e_2 & e_3 & i_1 & i_2 & i_3 \\ \hline e_1 & & & & 1 & & \\ e_2 & & & & & 1 & \\ e_3 & & & & & & 1 \\ i_1 & 1 & & & & 1 & \\ i_2 & & 1 & & & & 1 \\ i_3 & & & 1 & 1 & & \end{array}$$

- rows/columns ordered lexicographically, in contrast to



$$\sim \begin{Bmatrix} (e_1, i_2) \\ (e_2, i_3) \\ (e_3, i_1) \\ (i_1, i_2) \\ (i_2, i_3) \\ (i_3, i_1) \end{Bmatrix} \sim \begin{array}{c|cccccc} & e_1 & e_2 & e_3 & i_1 & i_2 & i_3 \\ \hline e_1 & & & & & 1 & \\ e_2 & & & & & & 1 \\ e_3 & & & & 1 & & \\ i_1 & & & 1 & & 1 & \\ i_2 & 1 & & & & & 1 \\ i_3 & & 1 & & 1 & & \end{array}$$

- tree diagrams are special: no need to enumerate vertices

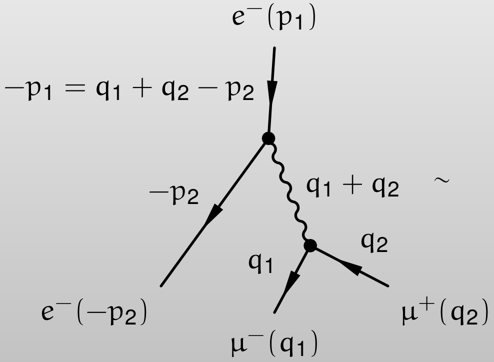
- tree diagrams are special: no need to enumerate vertices

$\because$ all propagators are uniquely specified by their momentum

- tree diagrams are special: no need to enumerate vertices

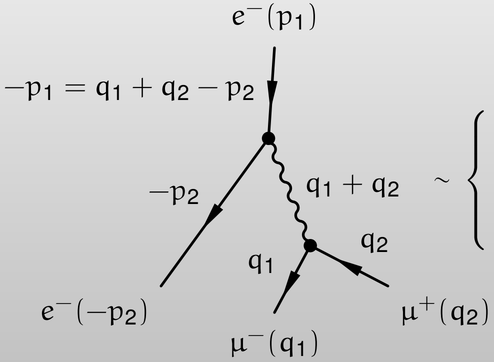
∵ all propagators are uniquely specified by their momentum

- crossing symmetry allows drawing as mathematical tree



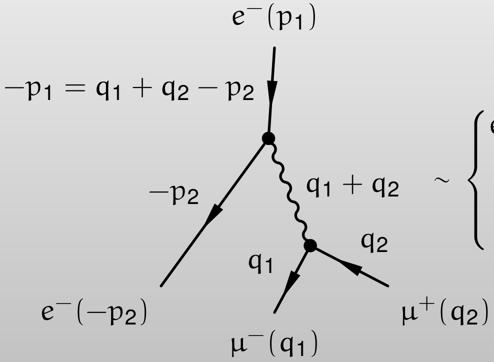
$\sim$

- tree diagrams are special: no need to enumerate vertices
- ∵ all propagators are uniquely specified by their momentum
- crossing symmetry allows drawing as mathematical tree



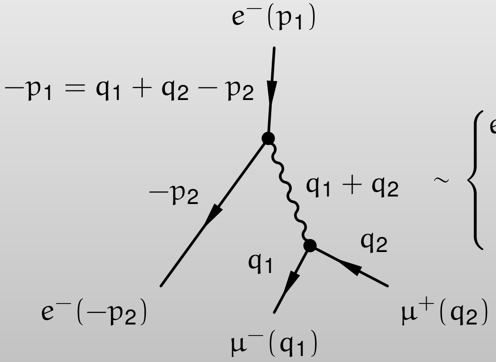
$$\sim \left\{ \begin{array}{l} \gamma(q_1+q_2) \mapsto \\ \qquad (\mu^-(q_1), \mu^+(q_2)) \end{array} \right\}$$

- tree diagrams are special: no need to enumerate vertices
- ∵ all propagators are uniquely specified by their momentum
- crossing symmetry allows drawing as mathematical tree



$$\sim \left\{ \begin{array}{l} e^-(q_1+q_2-p_2) \mapsto \\ \qquad (e^-(-p_2), \gamma(q_1+q_2)) \\ \gamma(q_1+q_2) \mapsto \\ \qquad (\mu^-(q_1), \mu^+(q_2)) \end{array} \right\}$$

- tree diagrams are special: no need to enumerate vertices
- ∵ all propagators are uniquely specified by their momentum
- crossing symmetry allows drawing as mathematical tree



$$\sim \left\{ \begin{array}{l} e^-(q_1+q_2-p_2) \mapsto \\ \qquad (e^-(-p_2), \gamma(q_1+q_2)) \\ \quad \gamma(q_1+q_2) \mapsto \\ \qquad\quad (\mu^-(q_1), \mu^+(q_2)) \end{array} \right\}$$

- a harmless 2-fold ambiguity from overall momentum conservation can be avoided, if we don't use the momentum at the root, i. e. p_1

- searching all diagrams directly using the Feynman rules is inefficient, because too many dead ends

- searching all diagrams directly using the Feynman rules is inefficient, because too many dead ends
  - more efficient 2 step approach

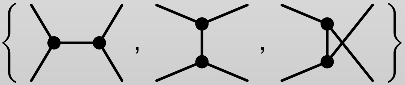
- searching all diagrams directly using the Feynman rules is inefficient, because too many dead ends
- more efficient 2 step approach
  1. generate all topologies, i. e. discard all quantum numbers (mass, spin, charges, flavor, color) and generate all Feynman diagrams with given number of loops and legs with Lagrangian

$$\mathcal{L}_{\text{topologies}} = \frac{1}{2}\partial_\mu\phi\partial^\mu\phi - \frac{\lambda_3}{3!}\phi^3 - \frac{\lambda_4}{4!}\phi^4 - \ldots$$

- searching all diagrams directly using the Feynman rules is inefficient, because too many dead ends
- more efficient 2 step approach
  1. generate all topologies, i. e. discard all quantum numbers (mass, spin, charges, flavor, color) and generate all Feynman diagrams with given number of loops and legs with Lagrangian

$$\mathcal{L}_{\text{topologies}} = \frac{1}{2}\partial_\mu\phi\partial^\mu\phi - \frac{\lambda_3}{3!}\phi^3 - \frac{\lambda_4}{4!}\phi^4 - \ldots$$

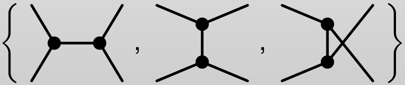
e. g.



- searching all diagrams directly using the Feynman rules is inefficient, because too many dead ends
- more efficient 2 step approach
  1. generate all topologies, i. e. discard all quantum numbers (mass, spin, charges, flavor, color) and generate all Feynman diagrams with given number of loops and legs with Lagrangian

$$\mathcal{L}_{\text{topologies}} = \frac{1}{2}\partial_\mu\phi\partial^\mu\phi - \frac{\lambda_3}{3!}\phi^3 - \frac{\lambda_4}{4!}\phi^4 - \ldots$$

     e. g.



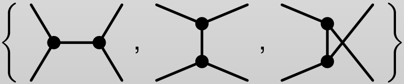
  2. add quantum numbers in all ways compatible with the Feynman rules and the external quantum numbers

- searching all diagrams directly using the Feynman rules is inefficient, because too many dead ends
  - more efficient 2 step approach

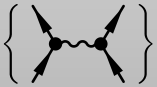
1. generate all topologies, i. e. discard all quantum numbers (mass, spin, charges, flavor, color) and generate all Feynman diagrams with given number of loops and legs with Lagrangian

$$\mathcal{L}_{\text{topologies}} = \frac{1}{2}\partial_\mu\phi\partial^\mu\phi - \frac{\lambda_3}{3!}\phi^3 - \frac{\lambda_4}{4!}\phi^4 - \ldots$$

e. g.



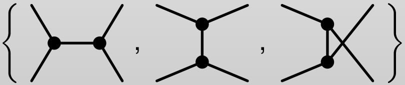
2. add quantum numbers in all ways compatible with the Feynman rules and the external quantum numbers , e. g.



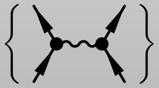
- searching all diagrams directly using the Feynman rules is inefficient, because too many dead ends
  - more efficient 2 step approach
    1. generate all topologies, i. e. discard all quantum numbers (mass, spin, charges, flavor, color) and generate all Feynman diagrams with given number of loops and legs with Lagrangian

$$\mathcal{L}_{\text{topologies}} = \frac{1}{2}\partial_\mu\phi\partial^\mu\phi - \frac{\lambda_3}{3!}\phi^3 - \frac{\lambda_4}{4!}\phi^4 - \ldots$$

e. g.



    2. add quantum numbers in all ways compatible with the Feynman rules and the external quantum numbers , e. g.

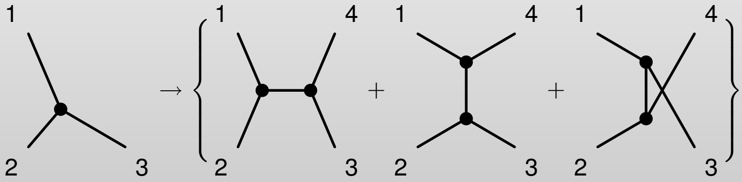


- avoids an enormous number of fruitless attempts

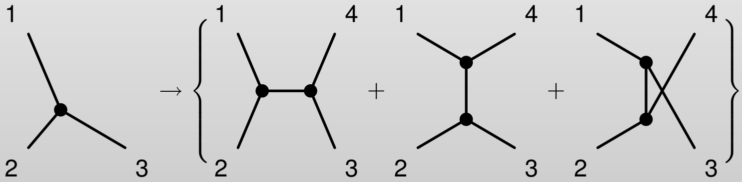
- Algorithm for generating all tree diagram topologies with n external legs

- Algorithm for generating all tree diagram topologies with n external legs
  1. generate all tree diagrams with $n-1$ external legs (to iterate is human, to recurse is divine ...)

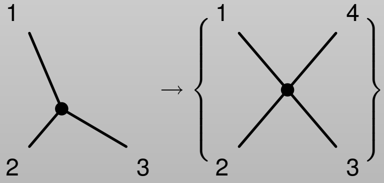
- Algorithm for generating all tree diagram topologies with n external legs
  1. generate all tree diagrams with $n-1$ external legs (to iterate is human, to recurse is divine ...)
  2. insert the nth line once
     - in every propagator, e. g.



- Algorithm for generating all tree diagram topologies with n external legs
  1. generate all tree diagrams with $n-1$ external legs (to iterate is human, to recurse is divine ...)
  2. insert the nth line once
     - in every propagator, e. g.



     - in very vertex of degree less than the maximum, e. g.



- corollary: there are

$$F(n) = (2n-5)!! = (2n-5) \cdot (2n-7) \cdot \ldots \cdot 3 \cdot 1$$

tree topologies with n legs

- corollary: there are

$$F(n) = (2n-5)!! = (2n-5)\cdot(2n-7)\cdot\ldots\cdot 3\cdot 1$$

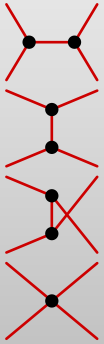
  tree topologies with n legs
- proof:

$$\begin{aligned} F(3) &= 1 \\ F(n) &= \Big(\underbrace{(n-1)}_{\text{external lines}} + \underbrace{(n-4)}_{\text{internal lines}}\Big)\cdot F(n-1) \\ &= (2n-5)\cdot F(n-1) = (2n-5)\cdot(2n-7)\cdot F(n-2) = \ldots \end{aligned}$$

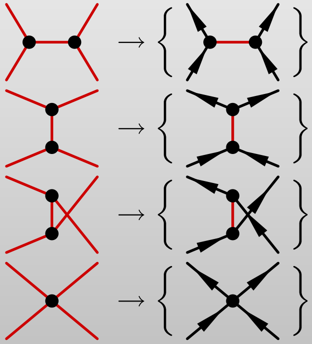
- adding quantum numbers from the outside in

- adding quantum numbers from the outside in
- e. g. $e^- e^+ \to \mu^- \mu^+$

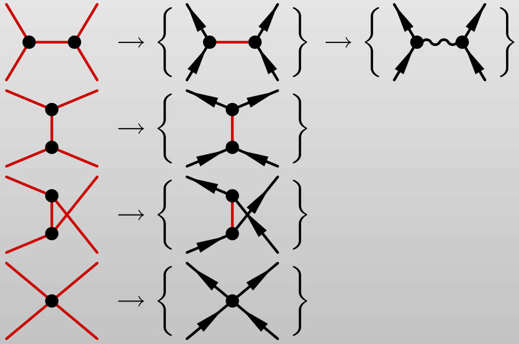
- adding quantum numbers from the outside in
- e. g. $e^- e^+ \to \mu^- \mu^+$



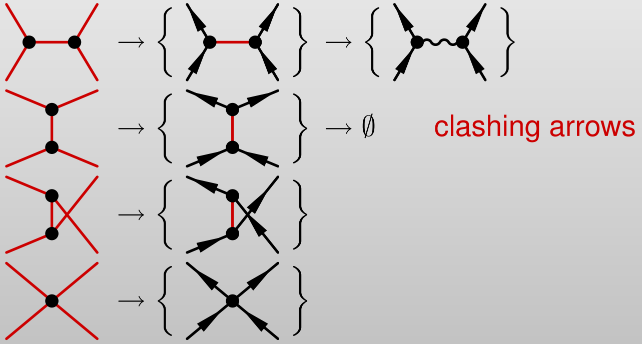
- adding quantum numbers from the outside in
- e. g. $e^- e^+ \to \mu^- \mu^+$



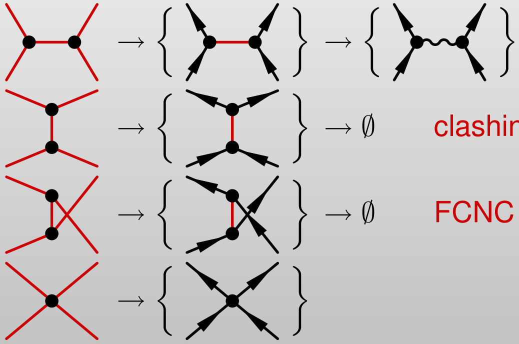
- adding quantum numbers from the outside in
- e. g. $e^- e^+ \to \mu^- \mu^+$



- adding quantum numbers from the outside in
- e. g. $e^- e^+ \to \mu^- \mu^+$



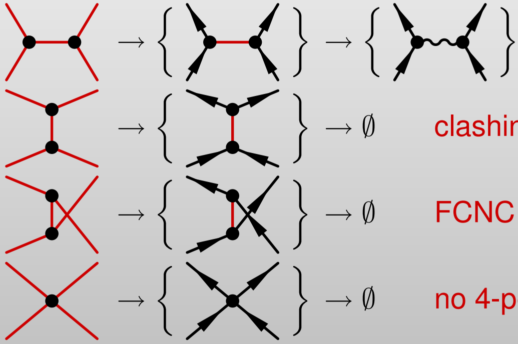
- adding quantum numbers from the outside in
- e. g. $e^- e^+ \to \mu^- \mu^+$



clashing arrows

FCNC

- adding quantum numbers from the outside in
- e. g. $e^- e^+ \to \mu^- \mu^+$



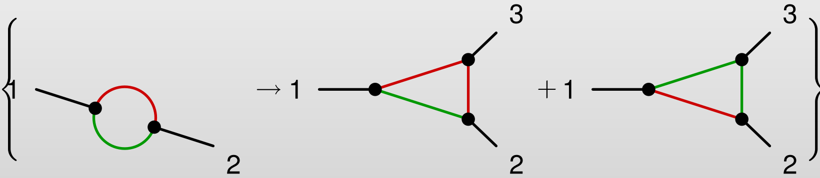
clashing arrows

FCNC

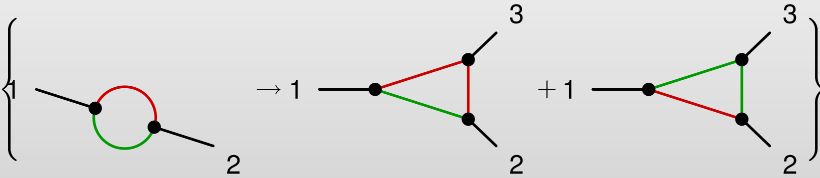
no 4-point vertex

- simple tree level algorithm $n \to n+1$ doesn't work for loop diagrams because

- simple tree level algorithm $n \to n+1$ doesn't work for loop diagrams because
  - diagrams will appear more than once

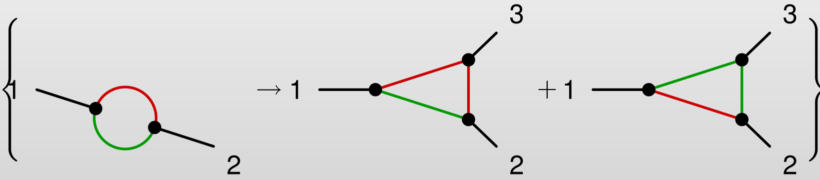


- simple tree level algorithm $n \to n+1$ doesn't work for loop diagrams because
  - diagrams will appear more than once



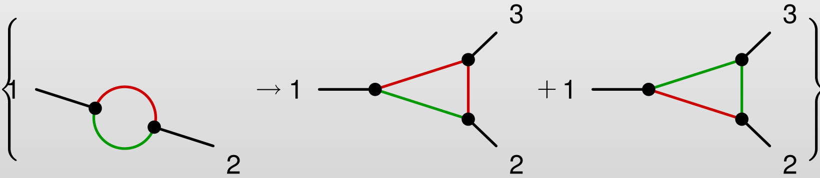
- easy to spot in this simple example

- simple tree level algorithm $n \to n + 1$ doesn't work for loop diagrams because
  - diagrams will appear more than once



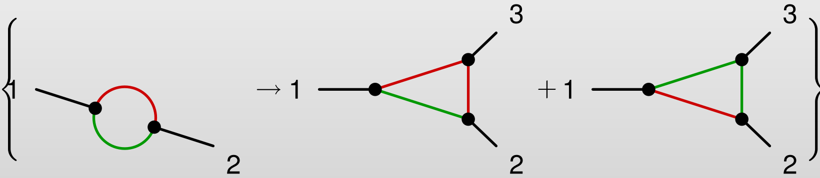
  - easy to spot in this simple example
  - nontrivial in the general case (e. g. ordered incidence matrices)

- simple tree level algorithm $n \to n+1$ doesn't work for loop diagrams because
  - diagrams will appear more than once



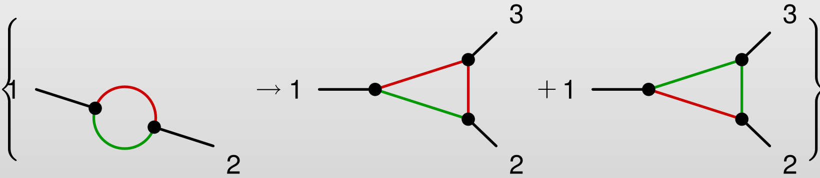
- easy to spot in this simple example
- nontrivial in the general case (e. g. ordered incidence matrices)
- NB: assignment of loop momenta not unique, can not be used to identify diagrams

- simple tree level algorithm $n \to n+1$ doesn't work for loop diagrams because
  - diagrams will appear more than once



-
  -
    - easy to spot in this simple example
    - nontrivial in the general case (e. g. ordered incidence matrices)
    - NB: assignment of loop momenta not unique, can not be used to identify diagrams
- we need a sufficient set starting topologies with n loops and without external legs (simple)

- simple tree level algorithm $n \to n+1$ doesn't work for loop diagrams because
  - diagrams will appear more than once



    - easy to spot in this simple example
    - nontrivial in the general case (e. g. ordered incidence matrices)
    - NB: assignment of loop momenta not unique, can not be used to identify diagrams
- we need a sufficient set starting topologies with n loops and without external legs (simple)
- most efficient, but badly documented, public tool for multi loop diagrams QGRAF [Nogueira, 1991]

The number of tree Feynman diagrams w/ n legs grows like a factorial, e. g. in ϕ^3-theory: $F(n) = (2n-5)!! = (2n-5)\cdot(2n-7)\cdot\ldots\cdot3\cdot1$

n		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

The number of tree Feynman diagrams w/ n legs grows like a factorial, e. g. in ϕ^3-theory: $F(n) = (2n-5)!! = (2n-5)\cdot(2n-7)\cdot\ldots\cdot 3\cdot 1$

n	$F(n)$	
4	3	
5	15	
6	105	
7	945	
8	10 395	
9	135 135	
10	2 027 025	
11	34 459 425	
12	654 729 075	
13	13 749 310 575	
14	316 234 143 225	
15	7 905 853 580 625	

The number of tree Feynman diagrams w/ n legs grows like a factorial, e. g. in ϕ^3-theory: $F(n) = (2n-5)!! = (2n-5)\cdot(2n-7)\cdot\ldots\cdot 3\cdot 1$

n	$F(n)$	
4	3	
5	15	
6	105	
7	945	
8	10 395	
9	135 135	
10	2 027 025	
11	34 459 425	
12	654 729 075	
13	13 749 310 575	
14	316 234 143 225	
15	7 905 853 580 625	

☹ computational costs grow beyond all reasonable limits

The number of tree Feynman diagrams w/ n legs grows like a factorial, e. g. in ϕ^3-theory: $F(n) = (2n-5)!! = (2n-5) \cdot (2n-7) \cdot \ldots \cdot 3 \cdot 1$

n	$F(n)$	
4	3	
5	15	
6	105	
7	945	
8	10 395	
9	135 135	
10	2 027 025	
11	34 459 425	
12	654 729 075	
13	13 749 310 575	
14	316 234 143 225	
15	7 905 853 580 625	

- ☹ computational costs grow beyond all reasonable limits
- ☹ gauge cancellations cause loss of precision

The number of tree Feynman diagrams w/ n legs grows like a factorial, e. g. in ϕ^3-theory: $F(n) = (2n-5)!! = (2n-5)\cdot(2n-7)\cdot\ldots\cdot 3\cdot 1$

n	$F(n)$	
4	3	
5	15	
6	105	
7	945	
8	10 395	
9	135 135	
10	2 027 025	
11	34 459 425	
12	654 729 075	
13	13 749 310 575	
14	316 234 143 225	
15	7 905 853 580 625	

- ☹ computational costs grow beyond all reasonable limits
- ☹ gauge cancellations cause loss of precision

Number of possible momenta in tree diagrams grows only exponentially

$$P(n) = \frac{2^n - 2}{2} - n = 2^{n-1} - n - 1$$

The number of tree Feynman diagrams w/ n legs grows like a factorial, e. g. in ϕ^3-theory: $F(n) = (2n-5)!! = (2n-5)\cdot(2n-7)\cdot\ldots\cdot 3\cdot 1$

n	$F(n)$	$P(n)$
4	3	3
5	15	10
6	105	25
7	945	56
8	10 395	119
9	135 135	246
10	2 027 025	501
11	34 459 425	1 012
12	654 729 075	2 035
13	13 749 310 575	4 082
14	316 234 143 225	8 177
15	7 905 853 580 625	16 368

- ☹ computational costs grow beyond all reasonable limits
- ☹ gauge cancellations cause loss of precision

Number of possible momenta in tree diagrams grows only exponentially

$$P(n) = \frac{2^n - 2}{2} - n = 2^{n-1} - n - 1$$

The number of tree Feynman diagrams w/ n legs grows like a factorial, e. g. in ϕ^3-theory: $F(n) = (2n-5)!! = (2n-5)\cdot(2n-7)\cdot\ldots\cdot 3\cdot 1$

n	$F(n)$	$P(n)$
4	3	3
5	15	10
6	105	25
7	945	56
8	10 395	119
9	135 135	246
10	2 027 025	501
11	34 459 425	1 012
12	654 729 075	2 035
13	13 749 310 575	4 082
14	316 234 143 225	8 177
15	7 905 853 580 625	16 368

- ☹ computational costs grow beyond all reasonable limits
- ☹ gauge cancellations cause loss of precision

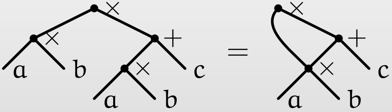
Number of possible momenta in tree diagrams grows only exponentially

$$P(n) = \frac{2^n - 2}{2} - n = 2^{n-1} - n - 1$$

∴ Feynman diagrams redundant for many external particles!

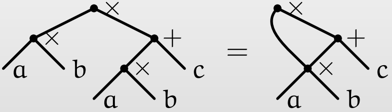
∴ Replace the forest of tree diagrams by the Directed Acyclical Graph (DAG) of the algebraic expression.

$$ab(ab+c) =$$

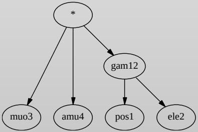


∴ Replace the forest of tree diagrams by the Directed Acyclical Graph (DAG) of the algebraic expression.

$$ab(ab+c) =$$

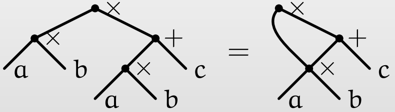


- simplest examples: $e^+e^- \to \mu^+\mu^-$, and (only QED)

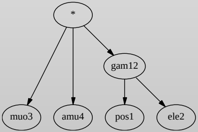


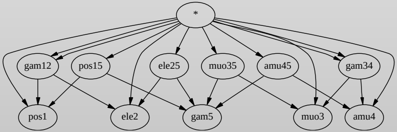
∴ Replace the forest of tree diagrams by the Directed Acyclical Graph (DAG) of the algebraic expression.

$ab(ab+c) =$

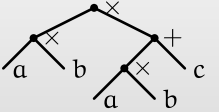
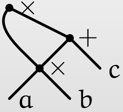


- simplest examples: $e^+e^- \to \mu^+\mu^-$, $e^+e^- \to \mu^+\mu^-\gamma$ and (only QED)

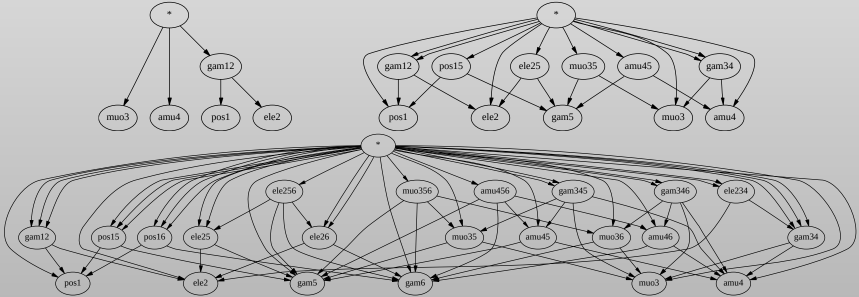




∴ Replace the forest of tree diagrams by the Directed Acyclical Graph (DAG) of the algebraic expression.

$ab(ab+c) =$  $=$ 

- simplest examples: $e^+e^- \to \mu^+\mu^-$, $e^+e^- \to \mu^+\mu^-\gamma$ and $e^+e^- \to \mu^+\mu^-\gamma\gamma$ (only QED)



Efficient tree amplitudes

- Berends-Giele Recursion Relations [Berends, Giele]
  - manual calculations

Efficient tree amplitudes

- Berends-Giele Recursion Relations [Berends, Giele]
  - manual calculations
- HELAS [Hagiwara et al.],
  - manual partial common subexpression elimination

Efficient tree amplitudes

- Berends-Giele Recursion Relations [Berends, Giele]
  - manual calculations
- HELAS [Hagiwara et al.],
  - manual partial common subexpression elimination
- Madgraph [Stelzer et al.], AMEGIC++, COMIX [Krauss et al.]:
  - partial common subexpression elimination

  $\therefore$ partial elimination of redundancy

Efficient tree amplitudes

- Berends-Giele Recursion Relations [Berends, Giele]
  - manual calculations
- HELAS [Hagiwara et al.],
  - manual partial common subexpression elimination
- Madgraph [Stelzer et al.], AMEGIC++, COMIX [Krauss et al.]:
  - partial common subexpression elimination
  - ∴ partial elimination of redundancy
- ALPHA [Caravaglios & Moretti]:
  - tree level scattering amplitude is Legendre transform of Lagragian
  - ☺ can be performed numerically, using only $P^*(n)$ independent variables

Efficient tree amplitudes

- Berends-Giele Recursion Relations [Berends, Giele]
  - manual calculations
- HELAS [Hagiwara et al.],
  - manual partial common subexpression elimination
- Madgraph [Stelzer et al.], AMEGIC++, COMIX [Krauss et al.]:
  - partial common subexpression elimination

  ∴ partial elimination of redundancy
- ALPHA [Caravaglios & Moretti]:
  - tree level scattering amplitude is Legendre transform of Lagragian

  ☺ can be performed numerically, using only $P^*(n)$ independent variables
- HELAC [Papadopoulos et al.]:
  - ALPHA algorithm can be reformulated as recursive numerical solution of Schwinger-Dyson equations

Efficient tree amplitudes

- Berends-Giele Recursion Relations [Berends, Giele]
  - manual calculations
- HELAS [Hagiwara et al.],
  - manual partial common subexpression elimination
- Madgraph [Stelzer et al.], AMEGIC++, COMIX [Krauss et al.]:
  - partial common subexpression elimination
  - ∴ partial elimination of redundancy
- ALPHA [Caravaglios & Moretti]:
  - tree level scattering amplitude is Legendre transform of Lagragian
  - ☺ can be performed numerically, using only $P^*(n)$ independent variables
- HELAC [Papadopoulos et al.]:
  - ALPHA algorithm can be reformulated as recursive numerical solution of Schwinger-Dyson equations
- O'Mega [TO et al.]:
  - systematic elimination of all redundancies
  - symbolic, generation of compilable code

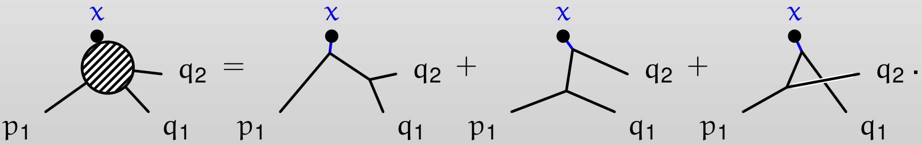
One particle off-shell wave functions (1POWs) are obtained from by applying the LSZ reduction formula to all but one line:

$$W(x; p_1, \ldots, p_n; q_1, \ldots, q_m) = \\ \langle \phi(q_1), \ldots, \phi(q_m); \text{out} | \Phi(x) | \phi(p_1), \ldots, \phi(p_n); \text{in} \rangle \; .$$

One particle off-shell wave functions (1POWs) are obtained from by applying the LSZ reduction formula to all but one line:

$$W(x; p_1, \ldots, p_n; q_1, \ldots, q_m) = \langle \phi(q_1), \ldots, \phi(q_m); \text{out} | \Phi(x) | \phi(p_1), \ldots, \phi(p_n); \text{in} \rangle \,.$$

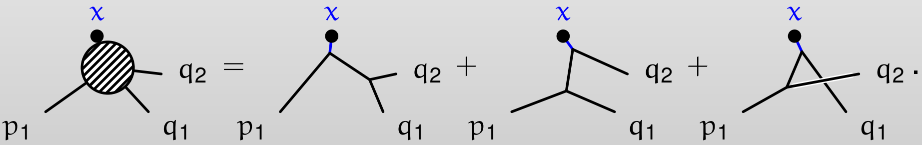
E. g. $\langle \phi(q_1), \phi(q_2); \text{out} | \Phi(x) | \phi(p_1); \text{in} \rangle$ in ϕ^3-theory at tree level



One particle off-shell wave functions (1POWs) are obtained from by applying the LSZ reduction formula to all but one line:

$$W(x; p_1, \ldots, p_n; q_1, \ldots, q_m) = \langle \phi(q_1), \ldots, \phi(q_m); \text{out} | \Phi(x) | \phi(p_1), \ldots, \phi(p_n); \text{in} \rangle \, .$$

E. g. $\langle \phi(q_1), \phi(q_2); \text{out} | \Phi(x) | \phi(p_1); \text{in} \rangle$ in ϕ^3-theory at tree level

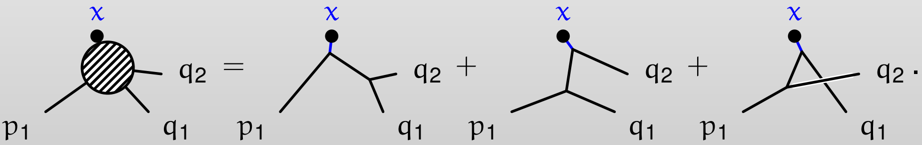


- the set of all 1POWs at tree level grows exponentially and can be constructed recursively from other 1POWs at tree level.

One particle off-shell wave functions (1POWs) are obtained from by applying the LSZ reduction formula to all but one line:

$$W(x; p_1, \ldots, p_n; q_1, \ldots, q_m) = \\ \langle \phi(q_1), \ldots, \phi(q_m); \text{out} | \Phi(x) | \phi(p_1), \ldots, \phi(p_n); \text{in} \rangle \,.$$

E. g. $\langle \phi(q_1), \phi(q_2); \text{out} | \Phi(x) | \phi(p_1); \text{in} \rangle$ in ϕ^3-theory at tree level



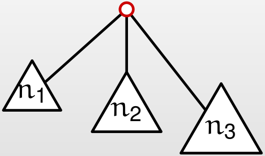
- the set of all 1POWs at tree level grows exponentially and can be constructed recursively from other 1POWs at tree level.

There exists a well defined set of keystones K that allow to express the sum of Feynman diagrams through 1POWs:

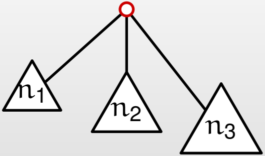
$$T = \sum_{i=1}^{F(n)} D_i = \sum_{k,l,m=1}^{P(n)} K^3_{f_k f_l f_m}(p_k, p_l, p_m) W_{f_k}(p_k) W_{f_l}(p_l) W_{f_m}(p_m)$$

Non-trivial: construction of inequivalent topologies for merging off-shell amplitudes to Feynman diagrams.

Non-trivial: construction of inequivalent topologies for merging off-shell amplitudes to Feynman diagrams. Solution: inequivalent partitionings of external momenta for a fixed vertex

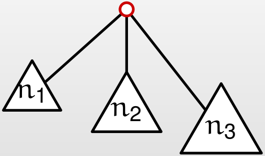


Non-trivial: construction of inequivalent topologies for merging off-shell amplitudes to Feynman diagrams. Solution: inequivalent partitionings of external momenta for a fixed vertex



n	$\sum$	$\sum$
4	4	$1 \cdot (1,1,1,1) + 3 \cdot (1,1,2)$
5	26	$1 \cdot (1,1,1,1,1) + 10 \cdot (1,1,1,2) + 15 \cdot (1,2,2)$
6	236	$1 \cdot (1,1,1,1,1,1) + 15 \cdot (1,1,1,1,2) + 40 \cdot (1,1,1,3)$
		$+ 45 \cdot (1,1,2,2) + 120 \cdot (1,2,3) + 15 \cdot (2,2,2)$

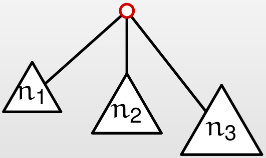
Non-trivial: construction of inequivalent topologies for merging off-shell amplitudes to Feynman diagrams. Solution: inequivalent partitionings of external momenta for a fixed vertex



n	$\sum$	$\sum$
4	4	$1 \cdot (1,1,1,1) + 3 \cdot (1,1,2)$
5	26	$1 \cdot (1,1,1,1,1) + 10 \cdot (1,1,1,2) + 15 \cdot (1,2,2)$
6	236	$1 \cdot (1,1,1,1,1,1) + 15 \cdot (1,1,1,1,2) + 40 \cdot (1,1,1,3)$
		$+ 45 \cdot (1,1,2,2) + 120 \cdot (1,2,3) + 15 \cdot (2,2,2)$

Subtlety: some partitions for an even number of external lines are degenerate

Non-trivial: construction of inequivalent topologies for merging off-shell amplitudes to Feynman diagrams. Solution: inequivalent partitionings of external momenta for a fixed vertex



n	$\sum$	$\sum$
4	4	$1 \cdot (1,1,1,1) + 3 \cdot (1,1,2)$
5	26	$1 \cdot (1,1,1,1,1) + 10 \cdot (1,1,1,2) + 15 \cdot (1,2,2)$
6	236	$1 \cdot (1,1,1,1,1,1) + 15 \cdot (1,1,1,1,2) + 40 \cdot (1,1,1,3)$
		$+ 45 \cdot (1,1,2,2) + 120 \cdot (1,2,3) + 15 \cdot (2,2,2)$

Subtlety: some partitions for an even number of external lines are degenerate, e. g. $(1,1,1,3)$ and $(1,2,3)$ contain the same diagram

and representatives must be chosen consistently.

Non trivial cross check from self-consistency of counting
$F(d_{\text{max}}, n)$ = # of Feynman diagrams with n external legs in unflavored

$$\mathcal{L} = \frac{1}{2}\partial_\mu\phi\partial^\mu\phi - \frac{m^2}{2}\phi^2 - \sum_{d=3}^{d_{\text{max}}} \frac{\lambda_d}{d!}\phi^d$$

theory. In a partition $N_{d,n} = \{n_1, n_2, \ldots, n_d\}$ with
$n = n_1 + n_2 + \cdots + n_d$, there are

$$\tilde{F}(d_{\text{max}}, N_{d,n}) = \frac{1}{(1+\delta_{n_d, n_1+n_2+\cdots+n_{d-1}})} \frac{n!}{|\mathcal{S}(N_{d,n})|} \prod_{i=1}^{d} \frac{F(d_{\text{max}}, n_i + 1)}{n_i!}$$

Feynman diagrams ($|\mathcal{S}(N)|$ the size of the symmetric group of N).

Non trivial cross check from self-consistency of counting $F(d_{\max}, n)$ = # of Feynman diagrams with n external legs in unflavored

$$\mathcal{L} = \frac{1}{2}\partial_\mu\phi\partial^\mu\phi - \frac{m^2}{2}\phi^2 - \sum_{d=3}^{d_{\max}} \frac{\lambda_d}{d!}\phi^d$$

theory. In a partition $N_{d,n} = \{n_1, n_2, \ldots, n_d\}$ with $n = n_1 + n_2 + \cdots + n_d$, there are

$$\tilde{F}(d_{\max}, N_{d,n}) = \frac{1}{(1 + \delta_{n_d, n_1+n_2+\cdots+n_{d-1}})} \frac{n!}{|\mathcal{S}(N_{d,n})|} \prod_{i=1}^{d} \frac{F(d_{\max}, n_i + 1)}{n_i!}$$

Feynman diagrams ($|\mathcal{S}(N)|$ the size of the symmetric group of N).

☺

$$F(d_{\max}, n) = \sum_{d=3}^{d_{\max}} \sum_{\substack{N=\{n_1,n_2,\ldots,n_d\} \\ n_1+n_2+\cdots+n_d=n \\ 1\leqslant n_1\leqslant n_2\leqslant\cdots\leqslant n_d\leqslant\lfloor n/2\rfloor}} \tilde{F}(d_{\max}, N)$$

can be be checked numerically up to $n = \mathcal{O}(100)$.

By virtue of they recursive construction, tree level 1POWs form a DAG

By virtue of they recursive construction, tree level 1POWs form a DAG:

$\therefore$ find the smallest DAG that corresponds to a given tree (i. e. a sum of Feynman diagrams)

By virtue of they recursive construction, tree level 1POWs form a DAG:

$\therefore$ find the smallest DAG that corresponds to a given tree (i. e. a sum of Feynman diagrams).

Systematic four step procedure:

Grow: starting from the external particles, build the tower of all 1POWs up to a given height (the height is always less than the number of external lines) and translate it to the equivalent DAG D.

By virtue of they recursive construction, tree level 1POWs form a DAG:

∴ find the smallest DAG that corresponds to a given tree (i. e. a sum of Feynman diagrams).

Systematic four step procedure:

Grow: starting from the external particles, build the tower of all 1POWs up to a given height (the height is always less than the number of external lines) and translate it to the equivalent DAG D.

Select: from D, determine all possible flavored keystones for the process under consideration and the 1POWs appearing in them.

By virtue of they recursive construction, tree level 1POWs form a DAG:

$\therefore$ find the smallest DAG that corresponds to a given tree (i. e. a sum of Feynman diagrams).

Systematic four step procedure:

Grow: starting from the external particles, build the tower of all 1POWs up to a given height (the height is always less than the number of external lines) and translate it to the equivalent DAG D.

Select: from D, determine all possible flavored keystones for the process under consideration and the 1POWs appearing in them.

Harvest: construct a sub-DAG $\mathrm{D}^* \subseteq \mathrm{D}$ consisting only of nodes that contribute to the 1POWs appearing in the flavored keystones.

By virtue of they recursive construction, tree level 1POWs form a DAG:

$\therefore$ find the smallest DAG that corresponds to a given tree (i. e. a sum of Feynman diagrams).

Systematic four step procedure:

Grow: starting from the external particles, build the tower of all 1POWs up to a given height (the height is always less than the number of external lines) and translate it to the equivalent DAG D.

Select: from D, determine all possible flavored keystones for the process under consideration and the 1POWs appearing in them.

Harvest: construct a sub-DAG $\mathrm{D}^* \subseteq \mathrm{D}$ consisting only of nodes that contribute to the 1POWs appearing in the flavored keystones.

Calculate: multiply the 1POWs as specified by the keystones and sum the keystones.

By virtue of they recursive construction, tree level 1POWs form a DAG:

∴ find the smallest DAG that corresponds to a given tree (i. e. a sum of Feynman diagrams).

Systematic four step procedure:

Grow: starting from the external particles, build the tower of all 1POWs up to a given height (the height is always less than the number of external lines) and translate it to the equivalent DAG D.

Select: from D, determine all possible flavored keystones for the process under consideration and the 1POWs appearing in them.

Harvest: construct a sub-DAG $\mathrm{D}^* \subseteq \mathrm{D}$ consisting only of nodes that contribute to the 1POWs appearing in the flavored keystones.

Calculate: multiply the 1POWs as specified by the keystones and sum the keystones.

☺ the resulting expression contains no more redundancies!

Even for vector particles, the 1POWs are 'almost' physical objects and satisfy simple Ward Identities in unbroken gauge theories

$$\frac{\partial}{\partial x_\mu} \langle \text{out} | A_\mu(x) | \text{in} \rangle_{\text{amp.}} = 0$$

Even for vector particles, the 1POWs are 'almost' physical objects and satisfy simple Ward Identities in unbroken gauge theories

$$\frac{\partial}{\partial x_\mu} \langle \text{out} | A_\mu(x) | \text{in} \rangle_{\text{amp.}} = 0$$

and in spontaneously gauge theories in R_ξ-gauge

$$\frac{\partial}{\partial x_\mu} \langle \text{out} | W_\mu(x) | \text{in} \rangle_{\text{amp.}} = \xi_W m_W \langle \text{out} | \phi_W(x) | \text{in} \rangle_{\text{amp.}} .$$

Even for vector particles, the 1POWs are 'almost' physical objects and satisfy simple Ward Identities in unbroken gauge theories

$$\frac{\partial}{\partial x_\mu} \langle \text{out} | A_\mu(x) | \text{in} \rangle_{\text{amp.}} = 0$$

and in spontaneously gauge theories in R_ξ-gauge

$$\frac{\partial}{\partial x_\mu} \langle \text{out} | W_\mu(x) | \text{in} \rangle_{\text{amp.}} = \xi_W m_W \langle \text{out} | \phi_W(x) | \text{in} \rangle_{\text{amp.}} .$$

☺ code for matrix elements can optionally be instrumented to check these Ward identities, testing the consistency a particular model and the numerical stability of expressions.

Even for vector particles, the 1POWs are ‘almost’ physical objects and satisfy simple Ward Identities in unbroken gauge theories

$$\frac{\partial}{\partial x_\mu} \langle \text{out} | A_\mu(x) | \text{in} \rangle_{\text{amp.}} = 0$$

and in spontaneously gauge theories in R_ξ-gauge

$$\frac{\partial}{\partial x_\mu} \langle \text{out} | W_\mu(x) | \text{in} \rangle_{\text{amp.}} = \xi_W m_W \langle \text{out} | \phi_W(x) | \text{in} \rangle_{\text{amp.}} .$$

☺ code for matrix elements can optionally be instrumented to check these Ward identities, testing the consistency a particular model and the numerical stability of expressions.

Amplitudes can be continued off-shell:

- Slavnov-Taylor Identities can be checked numerically by adding operator insertions implementing BRS transformations.

Slightly simplified `Model.T` signature that all models must implement:

```
module type Model.T =
  sig
    type flavor (* all quantum numbers *)
    val flavor_symbol : flavor -> string
    val conjugate : flavor -> flavor (* antiparticles *)
    val lorentz : flavor -> Coupling.lorentz (* spin *)
    val fermion : flavor -> int (* fermion, boson, antifermion *)
    val width : flavor -> Coupling.width (* scheme, not value! *)
    type gauge (* parametrized gauges *)
    val gauge_symbol : gauge -> string
    val propagator : flavor -> gauge Coupling.propagator
    type constant (* coupling constants *)
    val constant_symbol : constant -> string
    val fuse2 : flavor -> flavor ->
     (flavor * constant Coupling.t) list (* A_μ(p_12) ← gψ̄(p_1)γ_μψ(p_2) *)
    val fuse3 : flavor -> flavor -> flavor ->
     (flavor * constant Coupling.t) list (* φ(p_123) ← gφ(p_1)φ(p_2)φ(p_3) *)
    val fuse : flavor list -> (flavor * constant Coupling.t) list
  end
```

$e^+e^- \to \tilde\chi_1^+\tilde\chi_1^-$:

```
$ f90_MSSM -scatter "e+ e- -> ch1+ ch1-"
```

$e^+e^- \to \tilde{\chi}_1^+\tilde{\chi}_1^-$:

```
$ f90_MSSM -scatter "e+ e- -> ch1+ ch1-"
```

```
pure function l1bl1cp1cm1 (k, s) result (amp)
  real(kind=omega_prec), dimension(0:,:), intent(in) :: k
  integer, dimension(:), intent(in) :: s
  complex(kind=omega_prec) :: amp
  type(momentum) :: p1, p2, p3, p4
  type(bispinor) :: cp1_4, l1_2
  type(bispinor) :: cm1_3, l1b_1
  complex(kind=omega_prec) :: snc1_13
  type(vector) :: a_12, z_12
  type(momentum) :: p12, p13
  p1 = - k(:,1) ! incoming e+
  p2 = - k(:,2) ! incoming e-
  p3 =   k(:,3) ! outgoing ch1+
  p4 =   k(:,4) ! outgoing ch1-
  l1b_1 = u (mass(11), - p1, s(1))
  l1_2 = u (mass(11), - p2, s(2))
  cm1_3 = v (mass(69), p3, s(3))
  cp1_4 = v (mass(69), p4, s(4))
```

```
    p12 = p1 + p2
    a_12 = pr_feynman(p12, + v_ff(qlep,l1b_1,l1_2))
    z_12 = pr_unitarity(p12,mass(23),wd_tl(p12,width(23)), &
       + va_ff(gnclep(1),gnclep(2),l1b_1,l1_2))
    p13 = p1 + p3
    snc1_13 = pr_phi(p13,mass(54),wd_tl(p13,width(54)), &
       + sr_ff(g_yuk_ch1_sn1_1_c,l1b_1,cm1_3))
    amp = 0
    amp = amp + snc1_13*( - sl_ff(g_yuk_ch1_sn1_1,l1_2,cp1_4))
    amp = amp + z_12*( + va_ff(-gczc_1_1(1),-gczc_1_1(2),cm1_3,cp1_4))
    amp = amp + a_12*( + v_ff(qchar,cm1_3,cp1_4))
    amp = - amp ! 2 vertices, 1 propagators
  end function l1bl1cp1cm1
```

9 fusions, 3 propagators, 3 diagrams

```
    p12 = p1 + p2
    a_12 = pr_feynman(p12, + v_ff(qlep,l1b_1,l1_2))
    z_12 = pr_unitarity(p12,mass(23),wd_tl(p12,width(23)), &
       + va_ff(gnclep(1),gnclep(2),l1b_1,l1_2))
    p13 = p1 + p3
    snc1_13 = pr_phi(p13,mass(54),wd_tl(p13,width(54)), &
       + sr_ff(g_yuk_ch1_sn1_1_c,l1b_1,cm1_3))
    amp = 0
    amp = amp + snc1_13*( - sl_ff(g_yuk_ch1_sn1_1,l1_2,cp1_4))
    amp = amp + z_12*( + va_ff(-gczc_1_1(1),-gczc_1_1(2),cm1_3,cp1_4))
    amp = amp + a_12*( + v_ff(qchar,cm1_3,cp1_4))
    amp = - amp ! 2 vertices, 1 propagators
  end function l1bl1cp1cm1
```

9 fusions, 3 propagators, 3 diagrams

☺ readable code, can edited for exotic models or NLO vertex functions

Adding a photon $e^+e^- \to \tilde{\chi}_1^+\tilde{\chi}_1^-\gamma$:

```
$ f90_MSSM -scatter "e+ e- -> ch1+ ch1- A"
```

Adding a photon $e^+e^- \to \tilde{\chi}_1^+\tilde{\chi}_1^-\gamma$:

```
$ f90_MSSM -scatter "e+ e- -> ch1+ ch1- A"
```

```
pure function l1bl1cp1cm1a (k, s) result (amp)
  real(kind=omega_prec), dimension(0:,:), intent(in) :: k
  integer, dimension(:), intent(in) :: s
  complex(kind=omega_prec) :: amp
  type(momentum) :: p1, p2, p3, p4, p5
  type(bispinor) :: cp1_4, l1_2
  type(bispinor) :: cm1_3, l1b_1
  type(vector) :: a_5
  complex(kind=omega_prec) :: sn1_24, snc1_13
  type(bispinor) :: cp1_45, l1_25
  type(bispinor) :: cm1_35, l1b_15
  type(vector) :: a_34, a_12, z_34, z_12
  type(momentum) :: p12, p13, p15, p24, p25, p34, p35, p45
  p1 = - k(:,1) ! incoming e+
  p2 = - k(:,2) ! incoming e-
  p3 =   k(:,3) ! outgoing ch1+
  p4 =   k(:,4) ! outgoing ch1-
  p5 =   k(:,5) ! outgoing A
  l1b_1 = u (mass(11), - p1, s(1))
  l1_2 = u (mass(11), - p2, s(2))
  cm1_3 = v (mass(69), p3, s(3))
  cp1_4 = v (mass(69), p4, s(4))
  a_5 = conjg (eps (mass(22), p5, s(5)))
  p12 = p1 + p2
  a_12 = pr_feynman(p12, + v_ff(qlep,l1b_1,l1_2))
  z_12 = pr_unitarity(p12,mass(23),wd_tl(p12,width(23)), &
     + va_ff(gnclep(1),gnclep(2),l1b_1,l1_2))
```

```
    p13 = p1 + p3
    snc1_13 = pr_phi(p13,mass(54),wd_tl(p13,width(54)), &
       + sr_ff(g_yuk_ch1_sn1_1_c,l1b_1,cm1_3))
    p24 = p2 + p4
    sn1_24 = pr_phi(p24,mass(54),wd_tl(p24,width(54)), &
       + sl_ff(g_yuk_ch1_sn1_1,l1_2,cp1_4))
    p34 = p3 + p4
    a_34 = pr_feynman(p34, + v_ff(qchar,cm1_3,cp1_4))
    z_34 = pr_unitarity(p34,mass(23),wd_tl(p34,width(23)), &
       + va_ff(-gczc_1_1(1),-gczc_1_1(2),cm1_3,cp1_4))
    p15 = p1 + p5
    l1b_15 = pr_psi(p15,mass(11),wd_tl(p15,width(11)), + f_vf(-qlep,a_5,l1b_1))
    p25 = p2 + p5
    l1_25 = pr_psi(p25,mass(11),wd_tl(p25,width(11)), + f_vf(qlep,a_5,l1_2))
    p35 = p3 + p5
    cm1_35 = pr_psi(p35,mass(69),wd_tl(p35,width(69)), &
       + f_vf(-qchar,a_5,cm1_3))
    p45 = p4 + p5
    cp1_45 = pr_psi(p45,mass(69),wd_tl(p45,width(69)), + f_vf(qchar,a_5,cp1_4))
    amp = 0
    amp = amp + sn1_24*( + sr_ff(g_yuk_ch1_sn1_1_c,cm1_35,l1b_1))
    amp = amp + snc1_13*( - sl_ff(g_yuk_ch1_sn1_1,l1_25,cp1_4) &
       + sl_ff(g_yuk_ch1_sn1_1,cp1_45,l1_2))
    amp = amp + l1_25*( - f_vf(-qlep,a_34,l1b_1) &
       - f_vaf(-(gnclep(1)),gnclep(2),z_34,l1b_1))
    amp = amp + l1b_15*( - f_srf(g_yuk_ch1_sn1_1_c,sn1_24,cm1_3) &
       + f_vf(qlep,a_34,l1_2) + f_vaf(gnclep(1),gnclep(2),z_34,l1_2))
    amp = amp + z_12*( - va_ff(-(-gczc_1_1(1)),-gczc_1_1(2),cp1_45,cm1_3) &
       + va_ff(-gczc_1_1(1),-gczc_1_1(2),cm1_35,cp1_4))
    amp = amp + a_12*( - v_ff(-qchar,cp1_45,cm1_3) + v_ff(qchar,cm1_35,cp1_4))
  end function l1bl1cp1cm1a
```

28 fusions, 10 propagators, 12 diagrams

Conclusions & Outlook

- automated construction of efficient tree level matrix elements well understood

Conclusions & Outlook

- automated construction of efficient tree level matrix elements well understood
- topics not covered due to lack of time

Conclusions & Outlook

- automated construction of efficient tree level matrix elements well understood
- topics not covered due to lack of time
  - MHV amplitudes and twistors

Conclusions & Outlook

- automated construction of efficient tree level matrix elements well understood
- topics not covered due to lack of time
  - MHV amplitudes and twistors
  - computer aided multi loop calculations

Conclusions & Outlook

- automated construction of efficient tree level matrix elements well understood
- topics not covered due to lack of time
  - MHV amplitudes and twistors
  - computer aided multi loop calculations
  - fully automated 1 loop calculations: FeynArts, FormCalc, LoopTools, “OPP”

Conclusions & Outlook

- automated construction of efficient tree level matrix elements well understood
- topics not covered due to lack of time
  - MHV amplitudes and twistors
  - computer aided multi loop calculations
  - fully automated 1 loop calculations: FeynArts, FormCalc, LoopTools, “OPP”
  - sector decomposition GOLEM

Conclusions & Outlook

- automated construction of efficient tree level matrix elements well understood
- topics not covered due to lack of time
  - MHV amplitudes and twistors
  - computer aided multi loop calculations
  - fully automated 1 loop calculations: FeynArts, FormCalc, LoopTools, "OPP"
  - sector decomposition GOLEM
  - dipole subtraction MadDipole

Conclusions & Outlook

- automated construction of efficient tree level matrix elements well understood
- topics not covered due to lack of time
  - MHV amplitudes and twistors
  - computer aided multi loop calculations
  - fully automated 1 loop calculations: FeynArts, FormCalc, LoopTools, “OPP”
  - sector decomposition GOLEM
  - dipole subtraction MadDipole
  - NLO matching (anything automated public???)