MadGraph5 Tutorial

Olivier Mattelaer
UCL

Johan Alwall
FermiLab

Michel Herquet
NIKHEF*

Fabio Maltoni
UCL

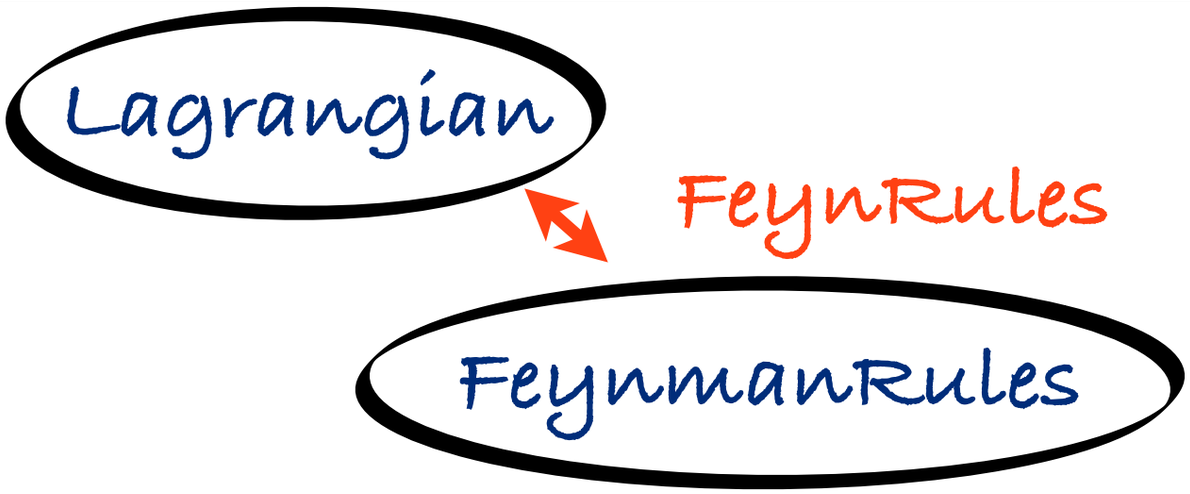
Tim Stelzer
UIUC

From Theory to Detector

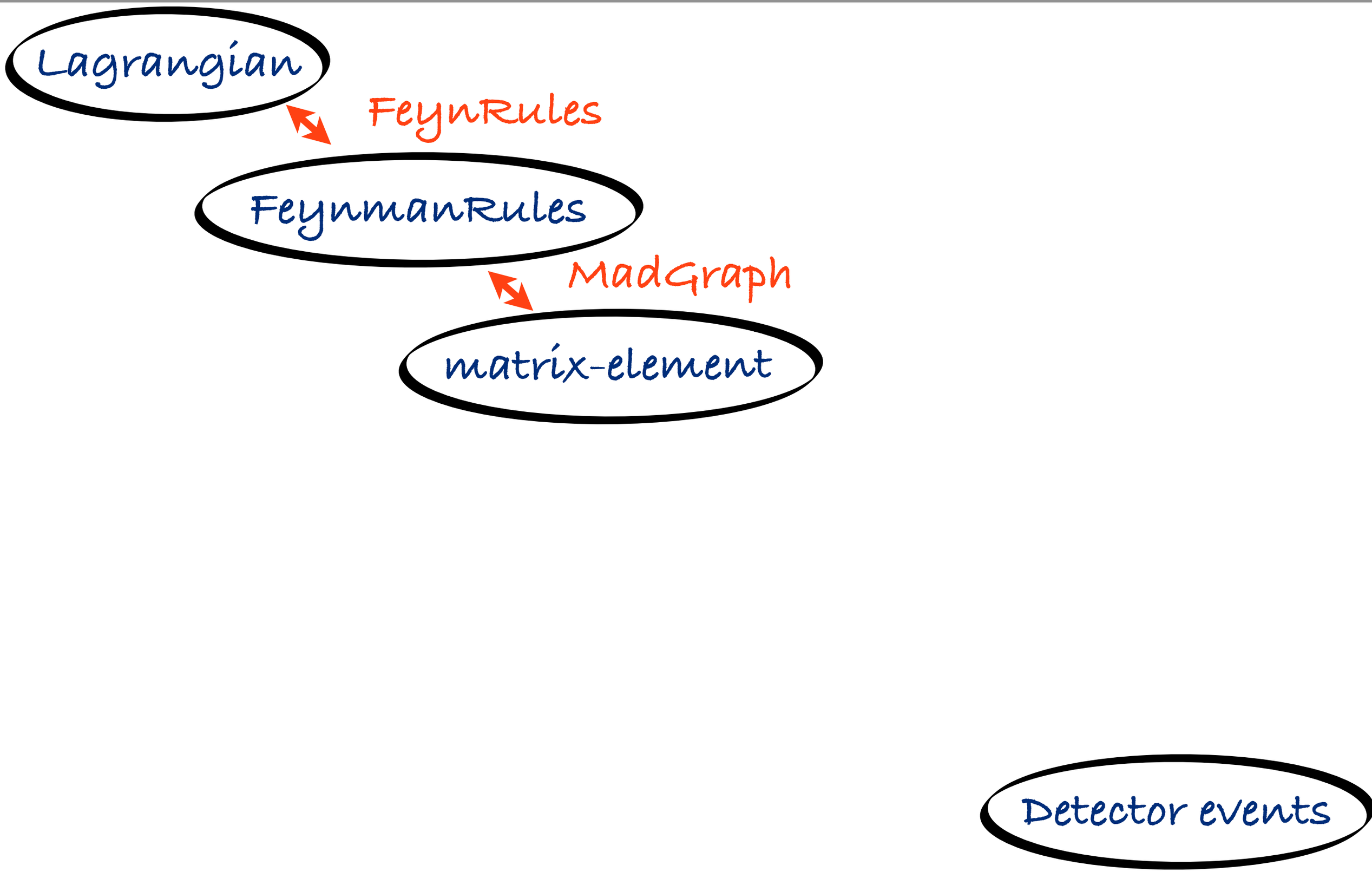
Lagrangian

Detector events

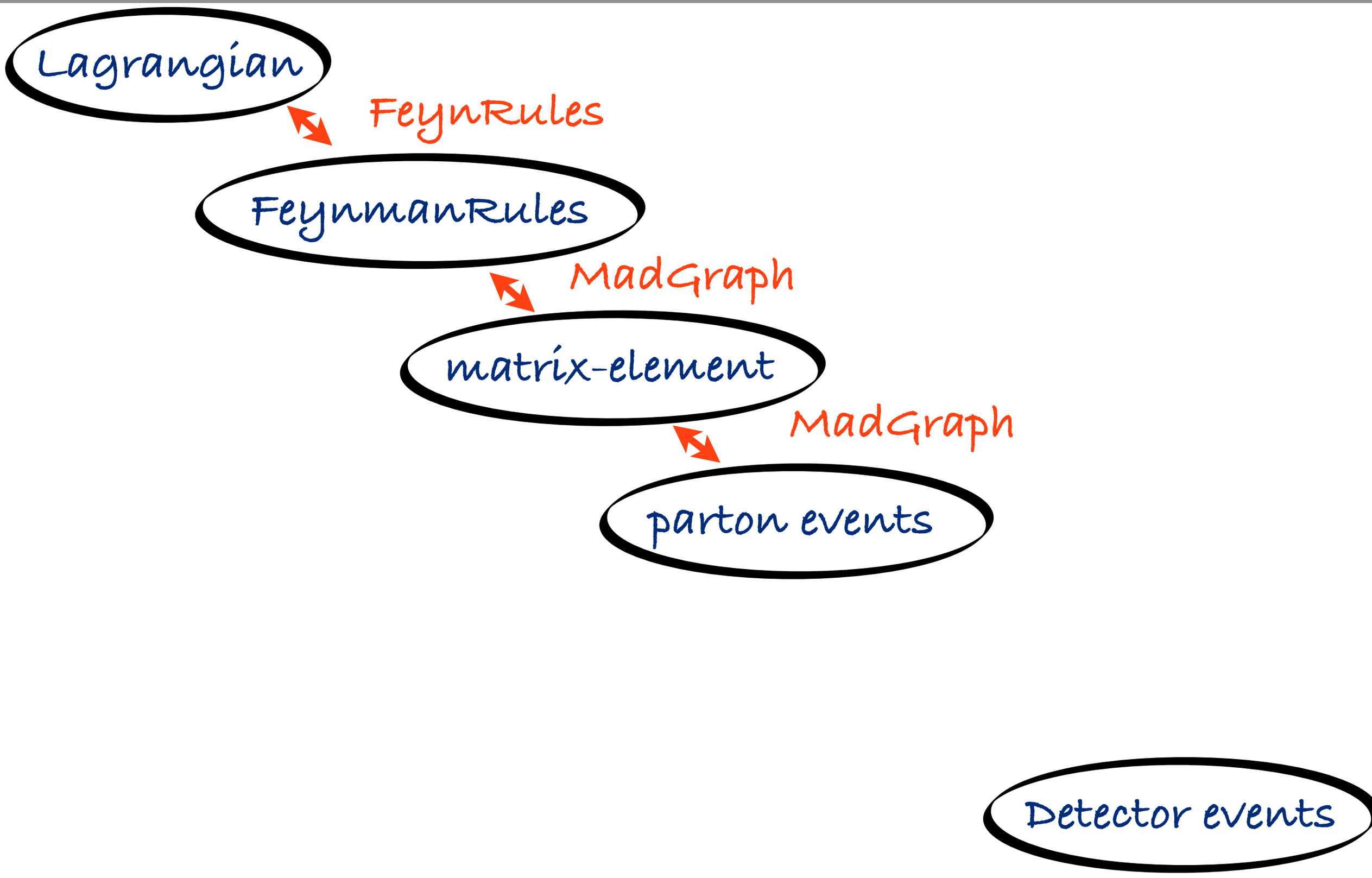
From Theory to Detector



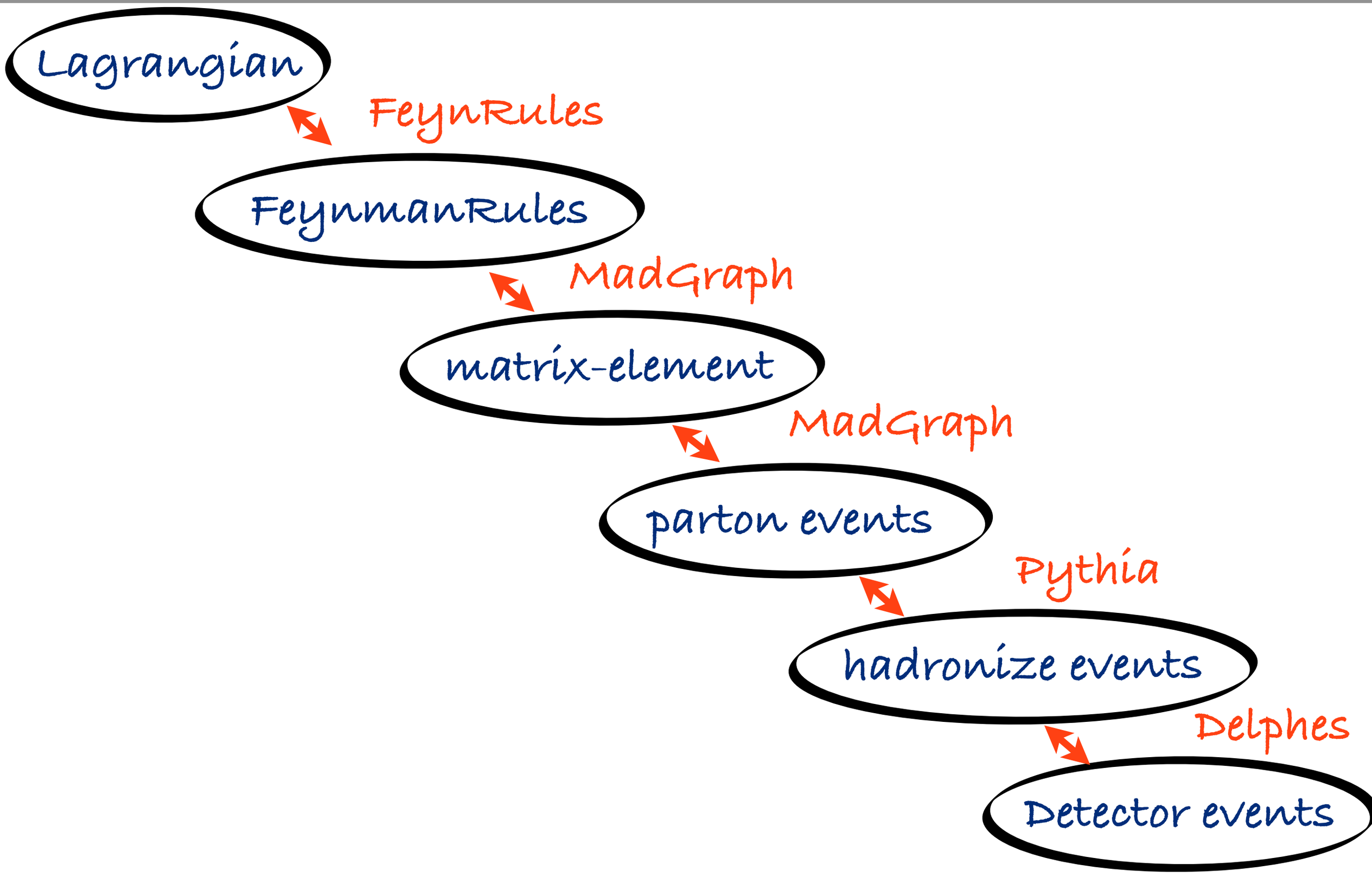
From Theory to Detector



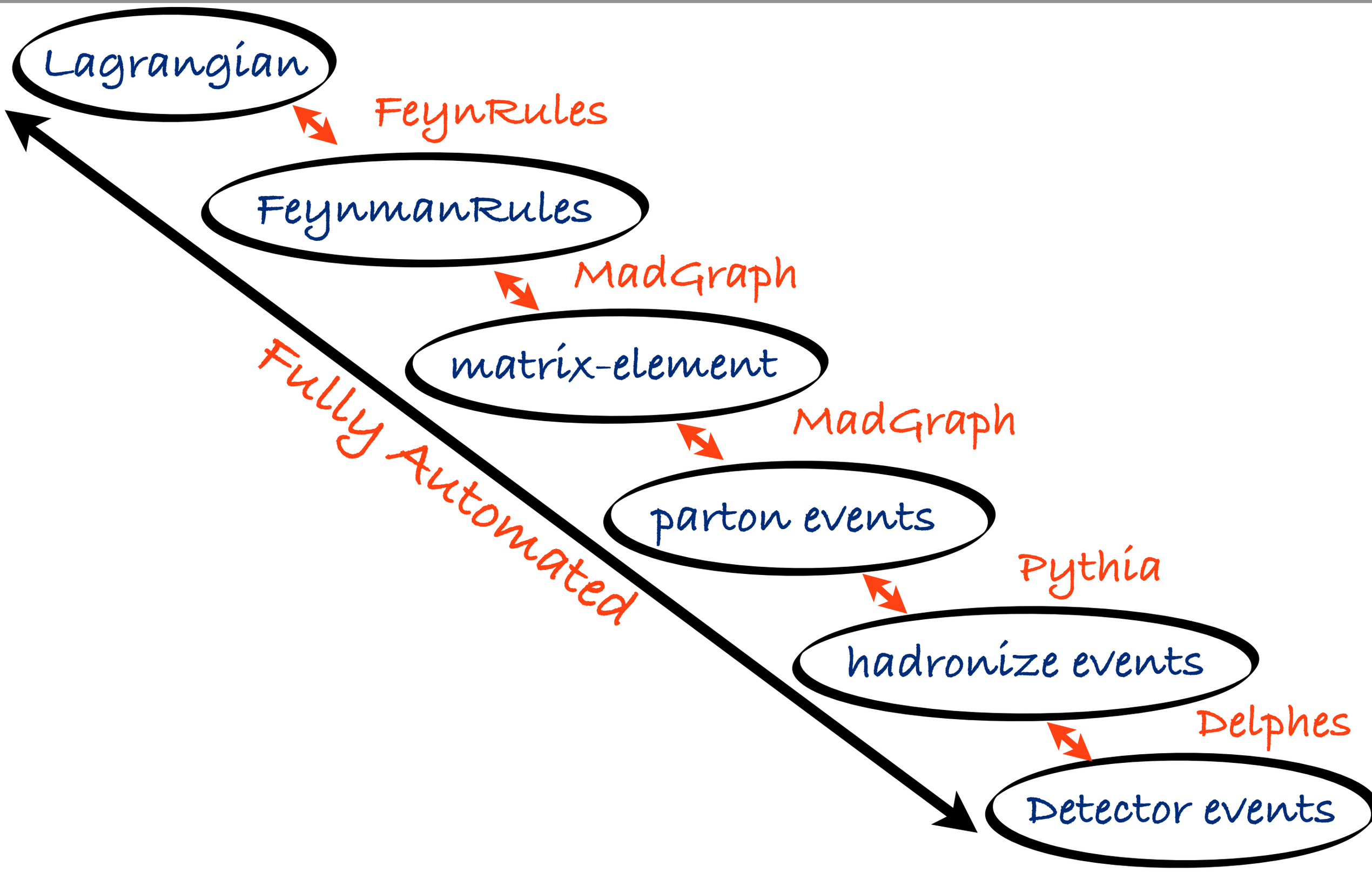
From Theory to Detector



From Theory to Detector



From Theory to Detector



PLAN

- ☐ How to install
- ☐ 2 Common situation
- ☐ 1 Full chain
- ☐ Focus on MG5 command / behavior

Installation

Requirements

- Python 2.6 (default on mac 10.6)

For Madevent Output

- fortran 77 compiler
- bash
- perl 5.8 (or higher)

For C++ Output

- C++ compiler

Note: MadGraph/MadEvent are available online

Where to find the code

- For user:
  - http://madgraph.hep.uiuc.edu/
  - http://madgraph.phys.ucl.ac.be/
  - https://launchpad.net/madgraph5

- For develloper:
  - install bazaar
  - $> bzr branch lp:madgraph5
  - dev in https://code.launchpad.net/madgraph5

How to install/start?

- $> tar -xzpvf MadGraph5_v1.1.0.tar.gz
- $> cd MadGraph5_v1_1_0/
- $> ./bin/mg5

MadGraph5 is running Now!

For Learning MadGraph5:

- mg5> help
- mg5> tutorial

How to install/start?

- $> tar -xzpvf MadGraph5_v1.1.0.tar.gz
- $> cd MadGraph5_v1_1_0/
- $> ./bin/mg5

MadGraph5 is running Now!

For Learning MadGraph5:

- mg5> help
- mg5> tutorial

Important to learn MG5

Standard Model Example

Goal

- W jet cross-section

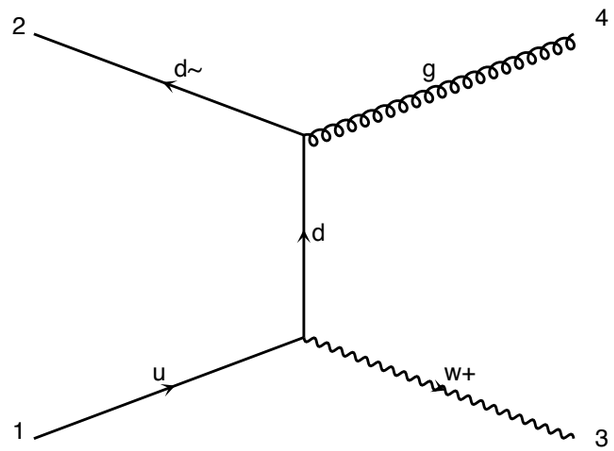


diagram 1 QED=1, QCD=1

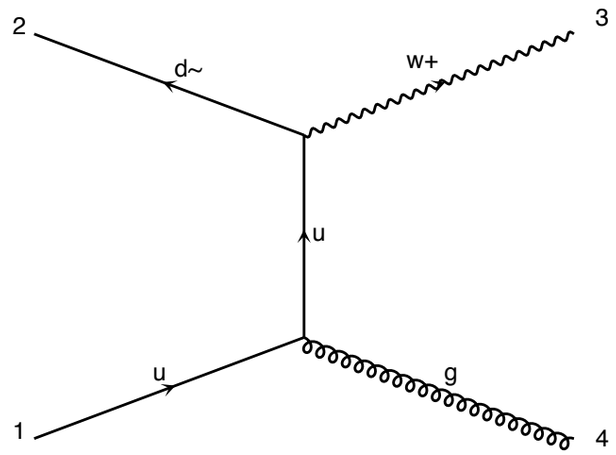


diagram 2 QED=1, QCD=1

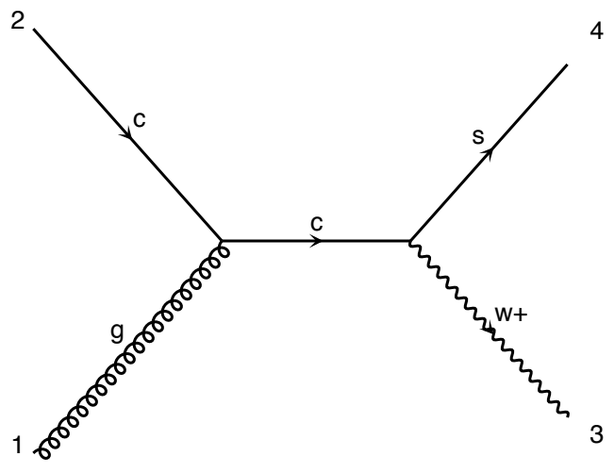


diagram 1 QED=1, QCD=1

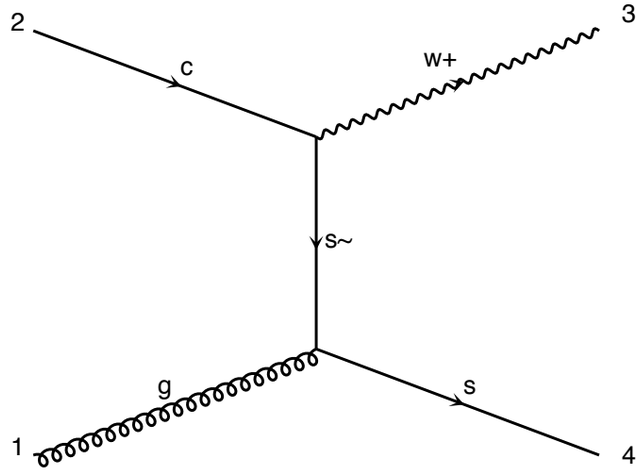


diagram 2 QED=1, QCD=1

List of command

```
mg5> generate p p > w+ j
mg5> output madevent
mg5> launch
```

Note:

- By default QED is set to its minimal value
- To launch pythia/pgs, you need to install the pythia-pgs package. (They are an install command)

only 3 command

It's very easy!

Generate Command

Generate Command

- require s-channel: p p > w+ > e+ ve

Generate Command

- require s-channel: p p > w+ > e+ ve
- forbids s-channel: p p > e+ ve $ w+

Generate Command

- require s-channel: p p > w+ > e+ ve
- forbids s-channel: p p > e+ ve $ w+
- forbids particles: p p > j j / z

Generate Command

- require s-channel: p p > w+ > e+ ve
- forbids s-channel: p p > e+ ve $ w+
- forbids particles: p p > j j / z
- alternate s-channel: p p > w+ | h+ > ta+ vt

Generate Command

- require s-channel: p p > w+ > e+ ve
- forbids s-channel: p p > e+ ve $ w+
- forbids particles: p p > j j / z
- alternate s-channel: p p > w+ | h+ > ta+ vt

- Possibility of decay chain

```
p p > t t~,
        (t > b w+, w+ > j j),
        (t~ > b~ w-, w- > mu- vm~)
```

Generate Command

- require s-channel: p p > w+ > e+ ve
- forbids s-channel: p p > e+ ve $ w+
- forbids particles: p p > j j / z
- alternate s-channel: p p > w+ | h+ > ta+ vt

- Possibility of decay chain

  p p > t t~,

  (t > b w+, w+ > j j),

  (t~ > b~ w-, w- > mu- vm~)

- Minimal QED order is taken by default

  p p > t t~ is the same as p p > t t~ QED=0!

Output Command

- mg5> output OUTPUT_TYPE PATH

OUTPUT_TYPE:

  - madevent (default)
  - standalone
  - standalone_cpp
  - pythia8

launch command

- mg5> launch PATH [options]
  - default PATH is the last created directory
  - possibility to choose to run in cluster/multi cpu mode
  - Can launch pythia/pgs (if install)

This is in addition to "old" way

- $> cd PATH
- $> ./bin/generate_events

MSSM Example

Goal

- squark pair production

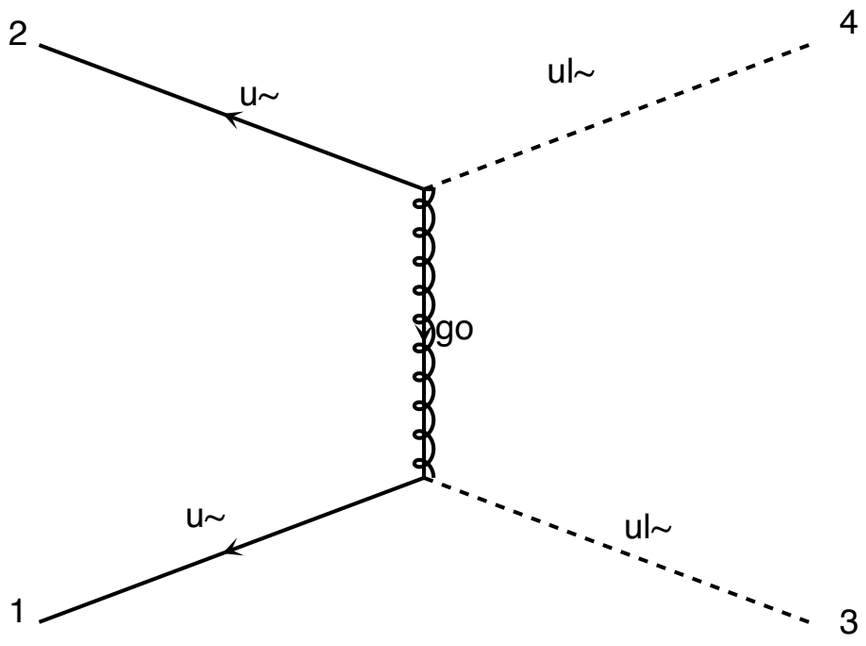


diagram 1 QED=0, QCD=2

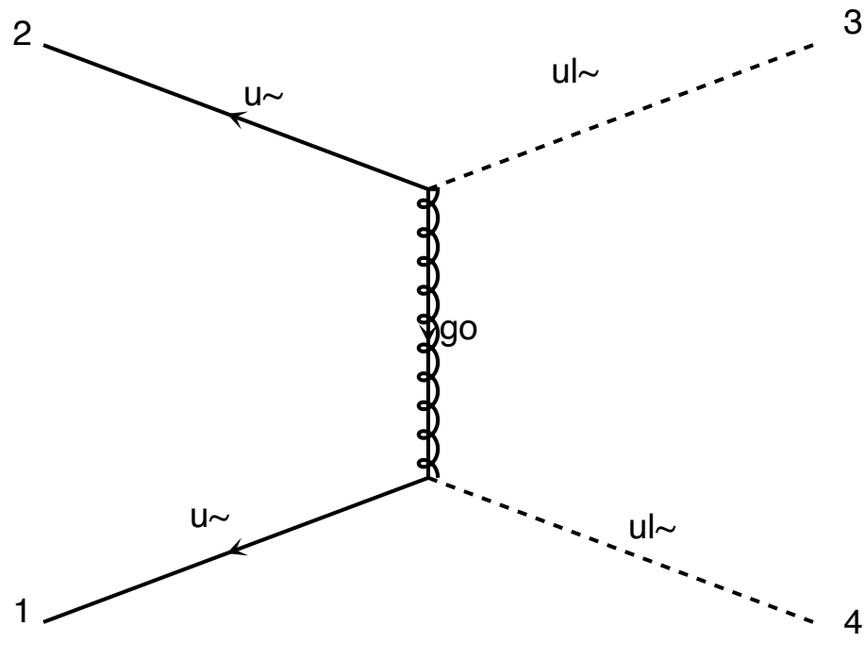


diagram 2 QED=0, QCD=2

List of command

```
mg5> import model mssm
mg5> define su = ur ur~ ul ul~
mg5> generate p p > su su
mg5> define sd = dr dr~ dl dl~
mg5> add process p p > sd sd
mg5> output
mg5> launch
```

import command

- mg5> import MODE PATH

MODE

  - model
  - model_v4
  - proc_v4
  - command

Decay chain Example

Decay Chain

Generate standalone output for those three diagrams (top quark pair production)

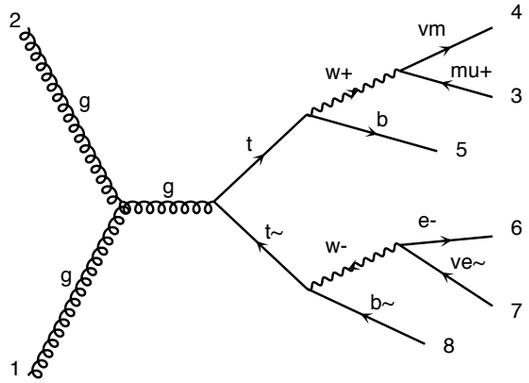


diagram 1

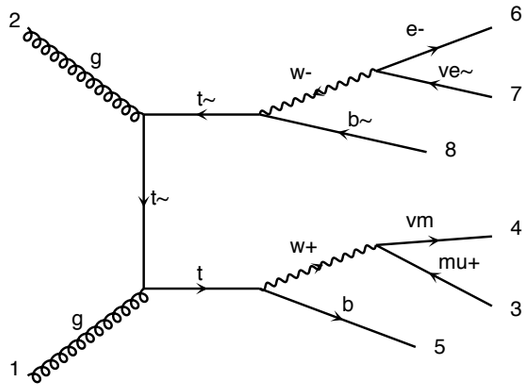
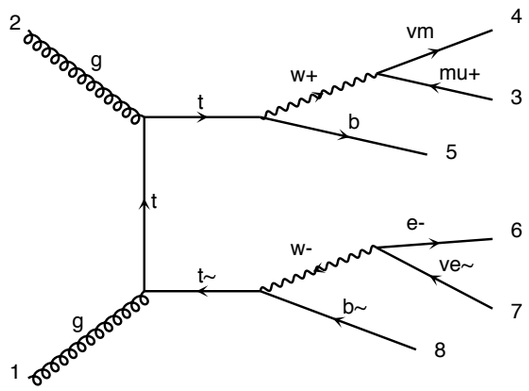


diagram 2



Advice:

help generate

help output

Solution

- import model sm
- generate p p > t t~, \
  (t > w+ b, w+ > mu+ vm), \
  (t~ > w- b~ , w- > e- ve~)
- output standalone

Decay chain

- paranthesis are for allowing sub-level
- Decay-chain forces the particles to be onshell.
- This is defined by the BW_cut

$$|m^* - m_0| \leq BW_{cut} * Width$$

- The $ command is the opposite of the decay chain. i.e the particles is forbidden to be onshell.

other command

other command

- Define : define a multi-particles

other command

- Define : define a multi-particles
- add process : same as generate but add a process

other command

- Define : define a multi-particles
- add process : same as generate but add a process
- set : some configuration

other command

- Define : define a multi-particles
- add process : same as generate but add a process
- set : some configuration
- check : validation of processs

other command

- Define : define a multi-particles
- add process : same as generate but add a process
- set : some configuration
- check : validation of processs
- display : status of diagram / model / ...

other command

- Define : define a multi-particles
- add process : same as generate but add a process
- set : some configuration
- check : validation of processs
- display : status of diagram / model / ...
- history : look at what you have done

other command

- Define : define a multi-particles
- add process : same as generate but add a process
- set : some configuration
- check : validation of processs
- display : status of diagram / model / ...
- history : look at what you have done
- open : open a file

other command

- Define : define a multi-particles
- add process : same as generate but add a process
- set : some configuration
- check : validation of processs
- display : status of diagram / model / ...
- history : look at what you have done
- open : open a file
- shell : execute a shell command (or !)

other command

- Define : define a multi-particles
- add process : same as generate but add a process
- set : some configuration
- check : validation of processs
- display : status of diagram / model / ...
- history : look at what you have done
- open : open a file
- shell : execute a shell command (or !)
- install: install optional package

The Full Chain

Objectives

- generate events for chromo-magnetic operator

$$\mathcal{L} = \frac{(H\bar{Q})\sigma^{\mu\nu}T^A t G^A_{\mu\nu}}{\Lambda^2} + h.c.,$$

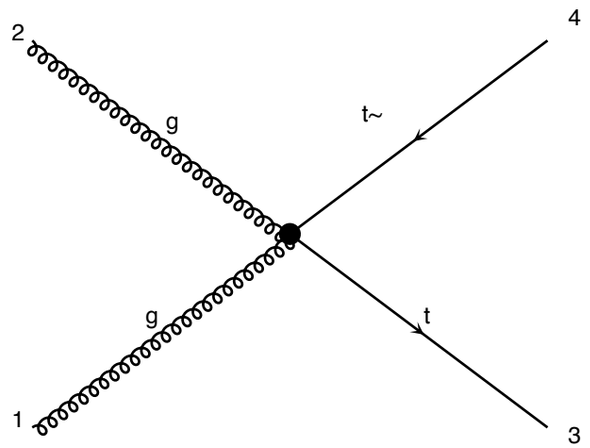


diagram 1 NP=2, QCD=1

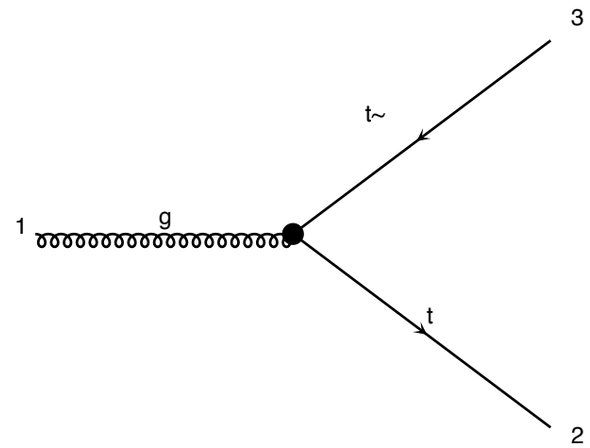


diagram 2 NP=2

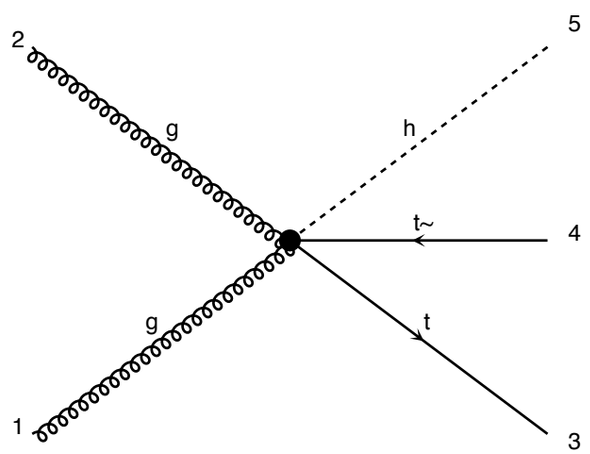


diagram 3 NP=2, QED=1, QCD=1

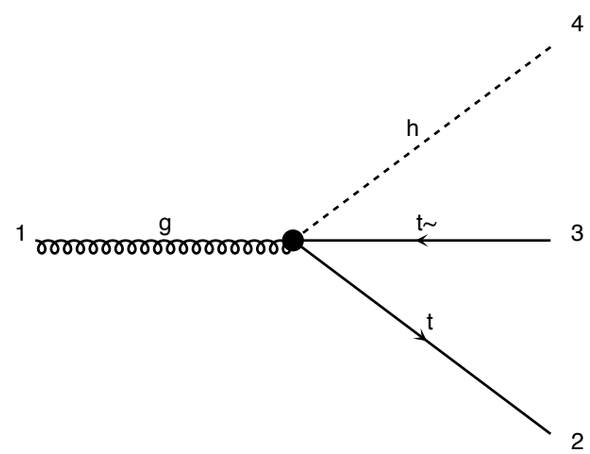


diagram 4 NP=2, QED=1

WorkSheet

- Write the Lagrangian in FR
- Write the UFO (WriteUFO command)
- mg5> import model Chromo
- mg5> display interactions
- mg5> check full p p > t t~ NP=2
- mg5> generate p p > t t~ NP=2
- output
- launch

Note

- FeynRules creates the UFO model (see FR talk)
- UFO model is the new type of model for MG5
- ALOHA creates automaticaly the HELAS routine (see talk on UFO/ALOHA)

The Full chain is automatic for BSM