evchain

An Event Chaining Utility

Ian-Woo Kim
CERN

Coordinating a Simplified Models Effort

30 Oct 2013

CERN

Motivation

- To ease event generation with long cascade
- Same goal as BRIDGE, but aim at more general
- Fine control of event topology
- Started from a real project...

LHC physics of Asymmetric Dark Matter

arXiv:1310.2617 IWK & K. Zurek

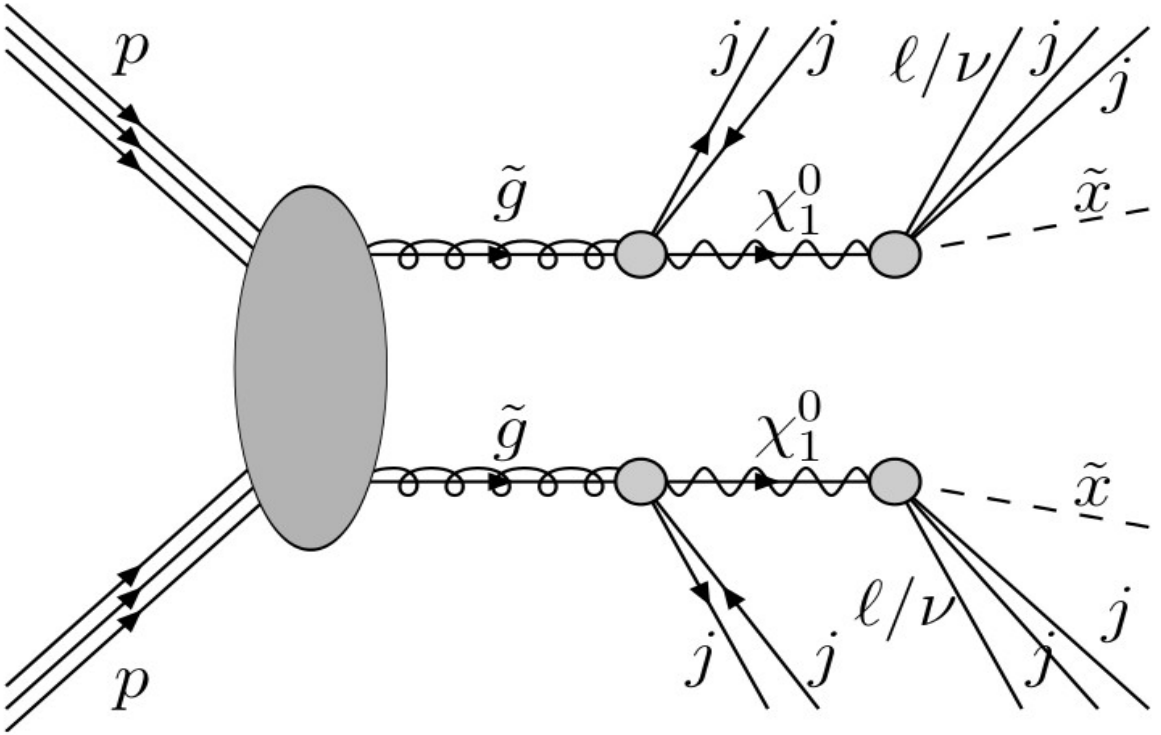
- Dark Matter X carries $B - L$ $\qquad m_X \sim \mathcal{O}(\mathrm{GeV})$
- Talk to SM via higher dim ops:

$$W = \frac{1}{\Lambda} X u^c d^c d^c, \frac{1}{\Lambda} X q \ell d^c, \frac{1}{\Lambda} X \ell \ell e^c$$

- Decays of MSSM LSP: More js/ls + Less MET

Challenges

- Very high multiplicity of final particles
- 4-body decays / exotic color vertices
- Need to splice a single process into multiple sub-parts



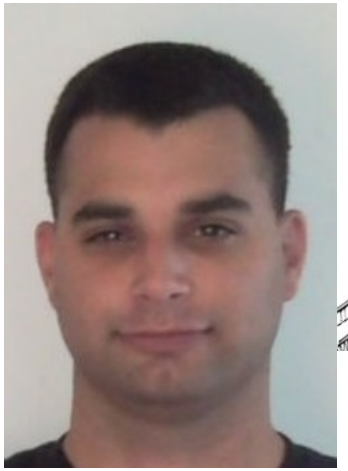
NWA is desirable

BRIDGE

Branching Ratio Inquiry/Decay Generated Events

http://www.lepp.cornell.edu/Research/TPP/BridgeSoftware.html

hep-ph/0703031, P.Meade & M.Reece



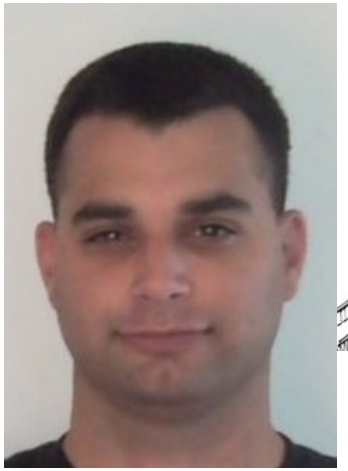
- MadGraph for Cross Process + BRIDGE for Decay Process
- Equipped with BR calculator and decay event generator using HELAS
- Generic 2-body / 3-body decays implemented

BRIDGE

Branching Ratio Inquiry/Decay Generated Events

http://www.lepp.cornell.edu/Research/TPP/BridgeSoftware.html

hep-ph/0703031, P.Meade & M.Reece



- MadGraph for Cross Process + BRIDGE for Decay Process
- Equipped with BR calculator and decay event generator using HELAS
- Generic 2-body / 3-body decays implemented

Wait, No 4-body? Eek, Deal Breaker for me!

BRIDGE

Branching Ratio Inquiry/Decay Generated Events

http://www.lepp.cornell.edu/Research/TPP/BridgeSoftware.html

hep-ph/0703031, P.Meade & M.Reece

- MadGraph for Cross Process + BRIDGE for Decay Process
- Equipped with BR calculator and decay event generator using HELAS
- Generic 2-body / 3-body decays implemented

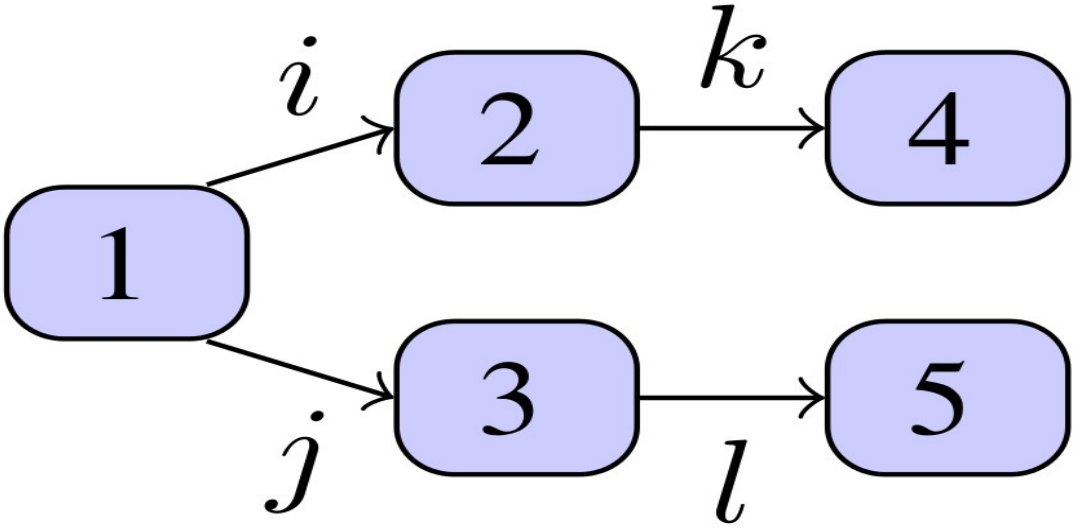
Wait, No 4-body? Eek, Deal Breaker for me!

Process specified only by decay rules, not by event topology

Let them out-sourced!

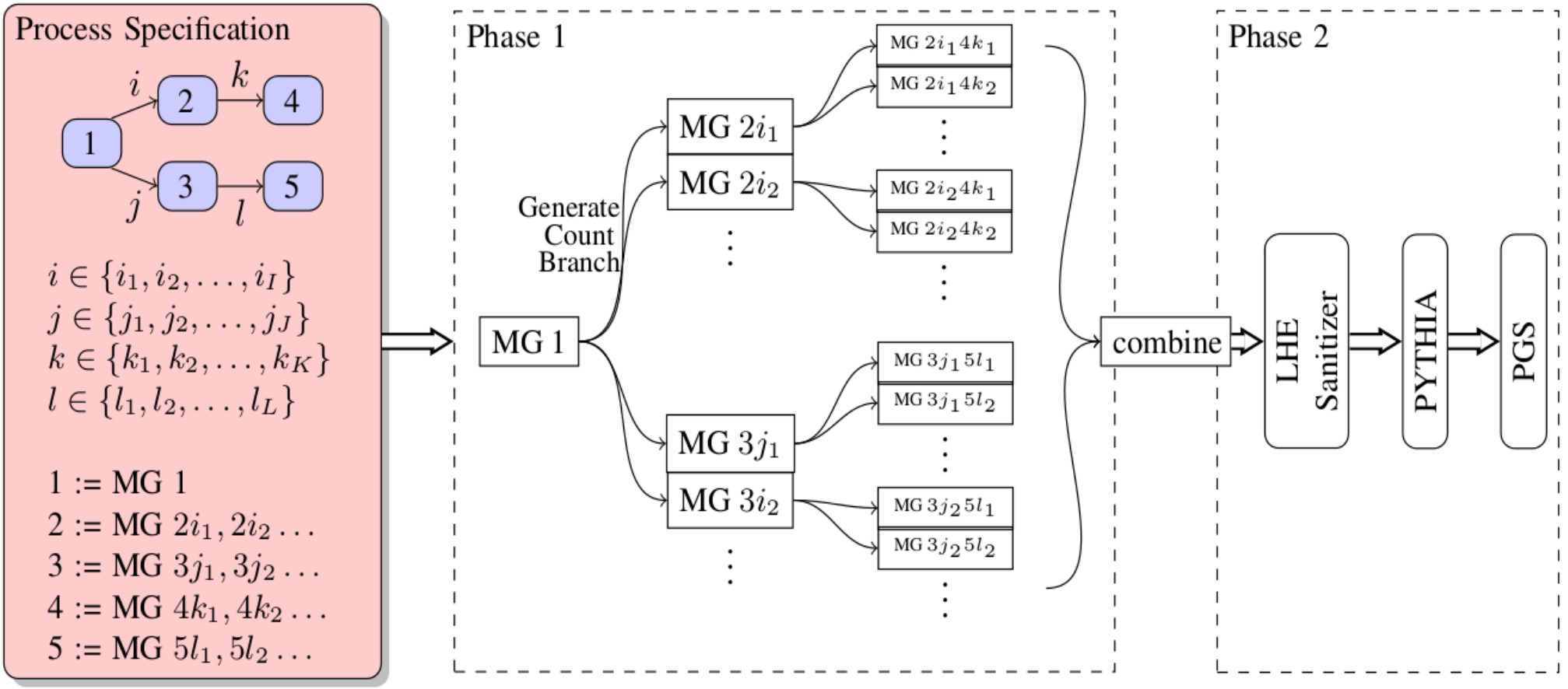
- Who cares about that my program cannot calculate BR if modern MC tools can do that for me anyway?
- Generic N-body decay events can be generated well by MadGraph or others.
- But I want to finely specify a single topology than grossly specify multiple topologies by decay rules.
- Evchain will work as a supervisor for such MC job coordiation

evchain as meta-montecarlo generator



- Arbitrary Tree Topology
- Arbitrary Processes Module 1,2.. with in-/out-particles specified
- Intermediate particles i,j,... are connecting modules: can be defined as multiparticles

evchain as meta-montecarlo generator



MG Directories are prepared, then orders MG event generation as required

evchain as meta-montecarlo generator

Event Generation Phase (Phase1)

- Each step is recursively ordered
- In each step, count and classify decay products
- In next daugther step, only generate events as many as required from previous counting
- Result is saved in LHE format (as MG does)
- Each MG dir name is hashed so to avoid name-clash
- Preparation step is only for once if one just want to generate events with different params w/o changing model/process

evchain as meta-montecarlo generator

Combining Phase (Phase2)

- LHE files in subdirectories are gathered
- Combining LHE files into a single LHE file
- Lorentz transformation/Color flow adjustment
- Send to Parton Shower (PYTHIA) and Detector Simulation (PGS) tools : Configurable

This tool is developed in

evchain as meta-montecarlo generator

Combining Phase (Phase2)

- LHE files in subdirectories are gathered
- Combining LHE files into a single LHE file
- Lorentz transformation/Color flow adjustment
- Send to Parton Shower (PYTHIA) and Detector Simulation (PGS) tools : Configurable

This tool is developed in (*alert*)

evchain as meta-montecarlo generator

Combining Phase (Phase2)

- LHE files in subdirectories are gathered
- Combining LHE files into a single LHE file
- Lorentz transformation/Color flow adjustment
- Send to Parton Shower (PYTHIA) and Detector Simulation (PGS) tools : Configurable

This tool is developed in (*alert*) haskell

FAQ: Why Haskell?

Official Answer

- Haskell is a pure functional programming language:
  elegant, safe, powerful and fast.
- Compiled binary (but interactive mode also exist)
- Strong Static Type: Very powerful for abstraction
- Native Parallel/Concurrent programming support
- Standardized Foreign Function Interface with C
- Rapidly attains industrial strength recently

Personal Answer

- I like it. This is my project anyway
- I am quite involved in development of haskell strongly attached with core community
- For example, my projects: **`fficxx`**: Haskell-C++ binding generator, **HROOT**: ROOT binding to Haskell (similar to PyROOT)

FAQ: Why Haskell?

Official Answer

- Haskell is a pure functional programming language:
  elegant, safe, powerful and fast.
- Compiled binary (but interactive mode also exist)
- Strong Static Type: Very powerful for abstraction
- Native Parallel/Concurrent programming support
- Standardized Foreign Function Interface with C
- Rapidly attains industrial strength recently

Personal Answer ghc (haskell compiler) is installed on lxplus.cern.ch last Monday

- I like it. This is my project anyway
- I am quite involved in development of haskell strongly attached with core community
- For example, my projects: **`fficxx`**: Haskell-C++ binding generator, **HROOT**: ROOT binding to Haskell (similar to PyROOT)

Embedded Topology-Specification Language

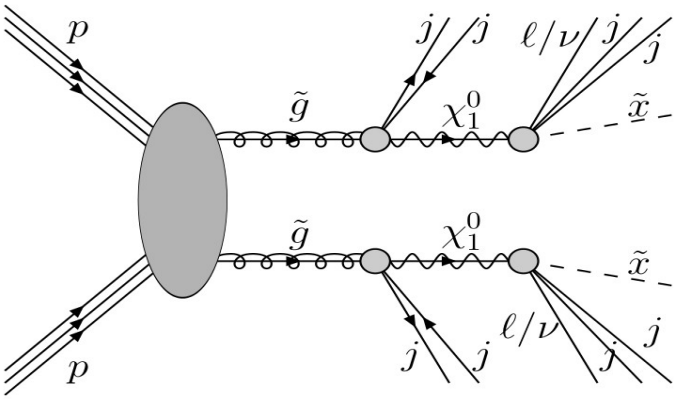
```
gluino = [1000021]
neutralino = [1000022]
jets = [1,2,3,4,-1,-2,-3,-4,21]
lepton_and_neutrino = [11,12,13,14,-11,-12,-13,-14]
adms = [9000201,-9000201,9000202,-9000202]

decay_gluino :: DDecay
decay_gluino = d (gluino, [decay_neutralino, t jets, t jets])

decay_neutralino :: DDecay
decay_neutralino = d (neutralino, [t lepton_and_neutrino, t jets, t jets, t adms])

total_process :: DCross
total_process = x (t proton, t proton, [decay_gluino, decay_gluino])

madgraph_process_map :: ProcSpecMap
madgraph_process_map =
    fromList [ (Nothing             , MGProc [] [ "p p > go go QED=0" ])
             , (Just (3,1000021,[]) , MGProc [] [ "go > n1 j j " ] )
             , (Just (4,1000021,[]) , MGProc [] [ "go > n1 j j " ] )
             , (Just (1,1000022,[3]), MGProc [ "define lep = e+ e- mu+ mu- ve ve~ vm vm~ "
                                             , "define sxx = sxxp sxxp~ " ]
                                             [ "n1 > sxx lep j j " ] )
             , (Just (1,1000022,[4]), MGProc [ "define lep = e+ e- mu+ mu- ve ve~ vm vm~ "
                                             , "define sxx = sxxp sxxp~ " ]
                                             [ "n1 > sxx lep j j " ] ) ]
```



Current Status and Plan

- It worked. Successfully used in a real project.
- Manual preparation, Code Documentation and Clean-up needed
- Will generalize for other MC tools and PS, Detector simulators
- Now work on support for spin correlation
- Now work on support for ME/PS matching

Welcome to fork!

- http://www.github.com/hep-platform/evchain
- http://www.github.com/hep-platform
- http://www.github.com/wavewave

Thank you!

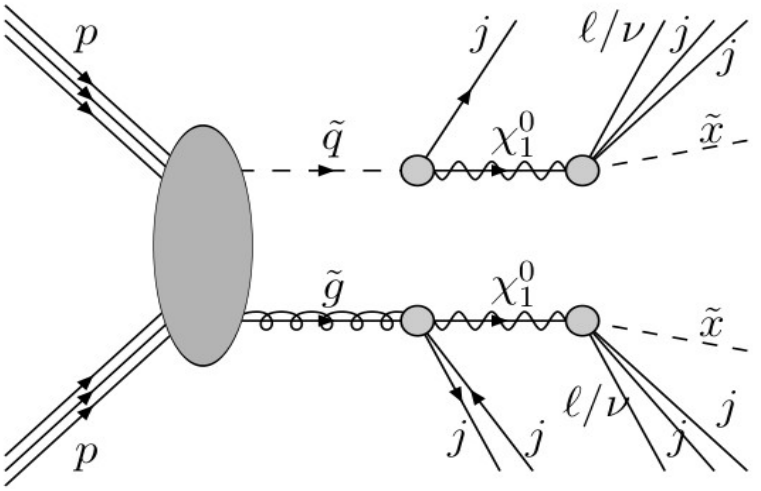
```
gluino = [1000021]
neutralino = [1000022]
jets = [1,2,3,4,-1,-2,-3,-4,21]
lepton_and_neutrino = [11,12,13,14,-11,-12,-13,-14]
adms = [9000201,-9000201,9000202,-9000202]
squarks = [  1000001, -1000001  -- sdown_L
           , 1000002, -1000002  -- sup_L
           , 1000003, -1000003  -- sstrange_L
           , 1000004, -1000004  -- scharm_L
           , 2000001, -2000001  -- sdown_R
           , 2000002, -2000002  -- sup_R
           , 2000003, -2000003  -- sstrange_R
           , 2000004, -2000004  -- scharm_R
          ]

decay_gluino :: DDecay
decay_gluino = d (gluino, [decay_neutralino, t jets, t jets])

decay_neutralino :: DDecay
decay_neutralino = d (neutralino, [t lepton_and_neutrino, t jets, t jets, t adms])

decay_squark :: DDecay
decay_squark = d (squarks, [p_neut, t jets])

total_process :: DCross
total_process = x (t proton, t proton, [decay_gluino, decay_squark])
```



```
madgraph_process_map :: ProcSpecMap
madgraph_process_map =
    fromList
      [ (Nothing, MGProc [ "define sq =  ul ul~ cl cl~ ur ur~ cr cr~ dl dl~ sl sl~ dr dr~ sr sr~" ]
                         [ "p p > go sq QED=0"])
      , (Just (3,1000021,[]), MGProc []  [ "go > n1 j j " ] )
      --
      , (Just (4,-1000001,[]), MGProc [] [ "dl~ > n1 j " ] )
      , (Just (4, 1000001,[]), MGProc [] [ "dl  > n1 j " ] )
      , (Just (4,-1000002,[]), MGProc [] [ "ul~ > n1 j " ] )
      , (Just (4, 1000002,[]), MGProc [] [ "ul  > n1 j " ] )
      , (Just (4,-1000003,[]), MGProc [] [ "sl~ > n1 j " ] )
      , (Just (4, 1000003,[]), MGProc [] [ "sl  > n1 j " ] )
      , (Just (4,-1000004,[]), MGProc [] [ "cl~ > n1 j " ] )
      , (Just (4, 1000004,[]), MGProc [] [ "cl  > n1 j " ] )
      , (Just (4,-2000001,[]), MGProc [] [ "dr~ > n1 j " ] )
      , (Just (4, 2000001,[]), MGProc [] [ "dr  > n1 j " ] )
      , (Just (4,-2000002,[]), MGProc [] [ "ur~ > n1 j " ] )
      , (Just (4, 2000002,[]), MGProc [] [ "ur  > n1 j " ])
      , (Just (4,-2000003,[]), MGProc [] [ "sr~ > n1 j " ] )
      , (Just (4, 2000003,[]), MGProc [] [ "sr  > n1 j " ] )
      , (Just (4,-2000004,[]), MGProc [] [ "cr~ > n1 j " ] )
      , (Just (4, 2000004,[]), MGProc [] [ "cr  > n1 j " ])
      --
      , (Just (1,1000022,[3]), MGProc [ "define lep = e+ e- mu+ mu- ve ve~ vm vm~ "
                                      , "define sxx = sxxp sxxp~ " ]
                                      [ "n1 > sxx lep j j " ] )
      , (Just (1,1000022,[4]), MGProc [ "define lep = e+ e- mu+ mu- ve ve~ vm vm~ "
                                      , "define sxx = sxxp sxxp~ " ]
                                      [ "n1 > sxx lep j j " ] )
      ]
```