GUM: GAMBIT Universal Models

Sanjay Bloor, on behalf of the GAMBIT collaboration

TeVPa 2019

Imperial College London

The University of Queensland Australia

gambit.hepforge.org

Outline

- Features of GAMBIT
- HEP toolchains
  - Lagrangian level tools
  - Including GAMBIT in the toolchain: GUM

GAMBIT: The Global And Modular BSM Inference Tool

gambit.hepforge.org EPJC **77** (2017) 784 arXiv:1705.07908

- Extensive model database – not just SUSY
- Extensive observable/data libraries
- Many statistical and scanning options (Bayesian & frequentist)
- *Fast* LHC likelihood calculator
- Massively parallel
- Fully open-source

- Fast definition of new datasets and theories
- Plug and play scanning, physics and likelihood packages

Members of:
ATLAS, Belle-II, CLiC, CMS, CTA, *Fermi*-LAT, DARWIN, IceCube, LHCb, SHiP, XENON

Authors of:
DarkSUSY, DDCalc, Diver, FlexibleSUSY, gamlike, GM2Calc, IsaTols, nulike, PolyChord, Rivet, SoftSUSY, SuperISO, SUSY-AI, WIMPSim

Recent collaborators:
Peter Athron, Csaba Balázs, Ankit Beniwal, Sanjay Bloor, Torsten Bringmann, Andy Buckley, José Eliel Camargo-Molina, Marcin Chrząszcz, Jonathan Cornell, Matthias Danninger, Joakim Edsjö, Ben Farmer, Andrew Fowlie, Tomás E. Gonzalo, Will Handley, Sebastian Hoof, Selim Hotinli, Felix Kahlhoefer, Anders Kvellestad, Julia Harz, Paul Jackson, Farvah Mahmoudi, Greg Martinez, Are Raklev, Janina Renk, Chris Rogan, Roberto Ruiz de Austri, Pat Scott, Patrick Stöcker, Aaron Vincent, Christoph Weniger, Martin White, Yang Zhang

40+ participants in 11 experiments and 14 major theory codes

Modules

Core – models, bookkeeping (EPJC, arXiv:1705.07908)

DarkBit – relic density, direct + indirect detection, axions (EPJC, arXiv:1705.07920)

ColliderBit – collider, higgs observables (EPJC, arXiv:1705.07919)

FlavBit – flavour physics (EPJC, arXiv:1705.07933)

SpecBit – spectrum objects, RGE running

DecayBit – decay widths

PrecisionBit – precision BSM tests

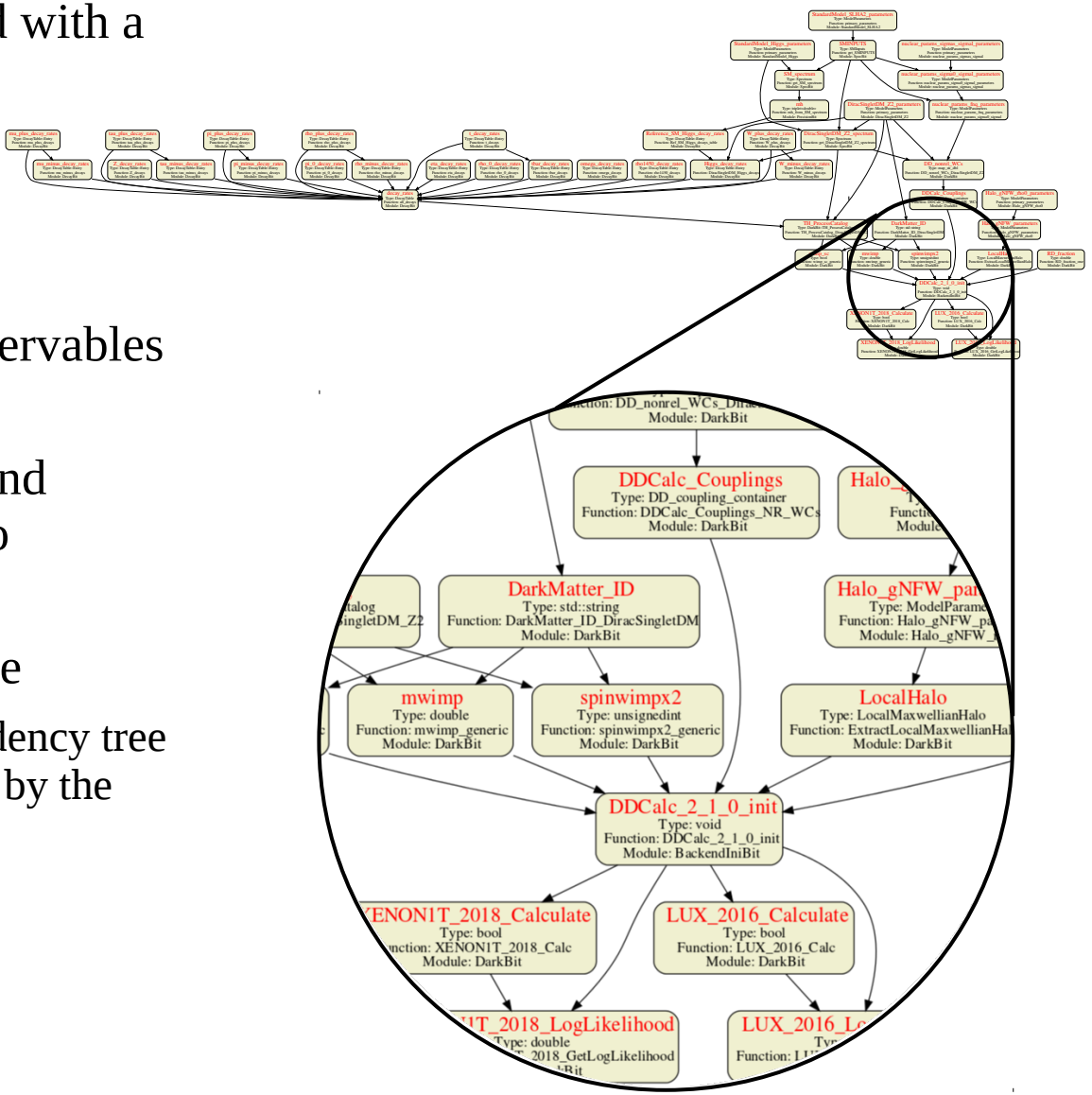
} (EPJC, arXiv:1705.07936)

ScannerBit – stats, sampling and optimisation (EPJC, arXiv:1705.07959)

NeutrinoBit – (active and sterile) neutrinos **NEW!** (arXiv:1908.02302)

Dependency resolution

- Every **function** in GAMBIT is tagged with a single **capability**
- Functions can have a **dependency** on capabilities
- At run time, GAMBIT organises a **dependency tree** of all requested observables and likelihoods
- Stitches together **module functions** and **backend functions** from parameter to observables/likelihoods
- GAMBIT ‘solves’ the dependency tree
  - will only run if there is a **unique** dependency tree which satisfies all of the **rules** specified by the user

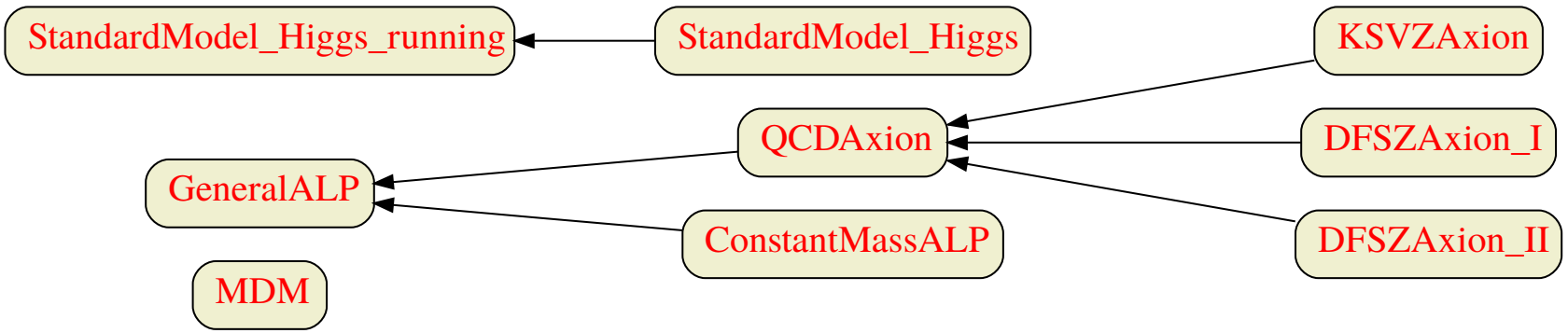


Hierarchical model database

- Models are defined by a set of parameters
- Child models can **inherit** from **parents**

  → Code **reusability**
- Model-specific **functions** are tagged

  → Code **safety**



Adding observables and likelihoods

- Add a new entry to a **module rollcall header**

```
#define CAPABILITY lnL_FermiLATdwarfs
START_CAPABILITY
  #define FUNCTION lnL_FermiLATdwarfs_gamLike
    START_FUNCTION(double)
    DEPENDENCY(GA_AnnYield, daFunk::Funk)
    DEPENDENCY(RD_fraction, double)
    BACKEND_REQ(lnL, (gam), double, (int, const std::vector<double> &, const std::vector<double> &))
    BACKEND_OPTION((gamLike), (gam))
  #undef FUNCTION
#undef CAPABILITY
```

- Define a **capability** for the new **module function** to return
  - Each function can have a **dependency** on other quantities
  - Can also depend on routines from **backends**
- Then simply write accompanying C++ code returning one result: the **capability**

GAMBIT as part of the HEP toolchain

GAMBIT as part of the HEP toolchain

- Lots of complexity in GAMBIT

GAMBIT as part of the HEP toolchain

- Lots of complexity in GAMBIT
  - Pros: very comprehensive BSM physics studies

GAMBIT as part of the HEP toolchain

- Lots of complexity in GAMBIT
  - Pros: very comprehensive BSM physics studies
  - Cons: there is a learning curve, and adding a new model by hand takes a lot of time

GAMBIT as part of the HEP toolchain

- Lots of complexity in GAMBIT
  - Pros: very comprehensive BSM physics studies
  - Cons: there is a learning curve, and adding a new model by hand takes a lot of time

    … until now!

GAMBIT as part of the HEP toolchain

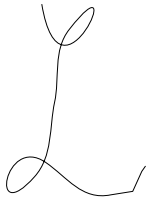
- Lots of complexity in GAMBIT
  - Pros: very comprehensive BSM physics studies
  - Cons: there is a learning curve, and adding a new model by hand takes a lot of time

    … until now!
- GUM provides a **fully automated** interface between GAMBIT and standard particle phenomenology toolchains!

HEP Toolchains

Well used pathways for phenomenology:

HEP Toolchains



Well used pathways for phenomenology:

1) Come up with a cool new theory

HEP Toolchains

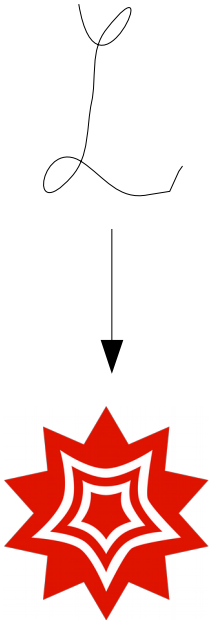
Well used pathways for phenomenology:

1) Come up with a cool new theory

2) Write down theory with Lagrangian level tools (LLTs)

(e.g. SARAH, FeynRules…)

sarah.hepforge.org
feynrules.irml.ucl.ac.be



HEP Toolchains

Well used pathways for phenomenology:

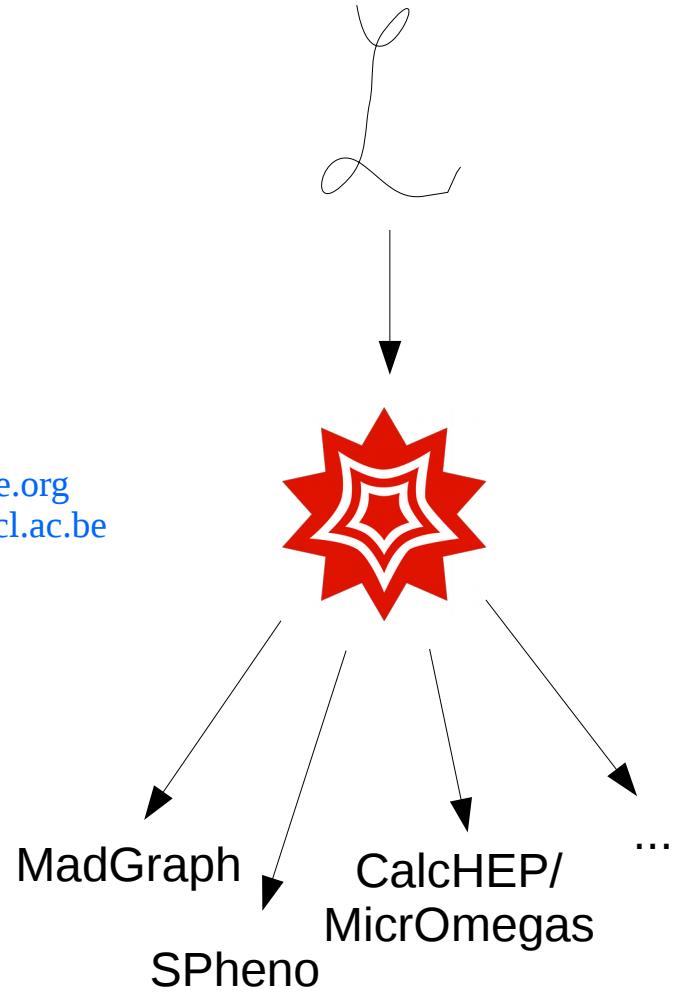
1) Come up with a cool new theory

2) Write down theory with Lagrangian level tools (LLTs)

(e.g. SARAH, FeynRules…)

sarah.hepforge.org
feynrules.irml.ucl.ac.be

3) LLTs create output for codes relating to Dark Matter, collider physics, flavour physics, spectrum calculators, decays…



launchpad.net/mg5amcnlo
spheno.hepforge.org
lapth.cnrs.fr/micromegas/
theory.sinp.msu.ru/~pukhov/calchep.html

HEP Toolchains

Well used pathways for phenomenology:

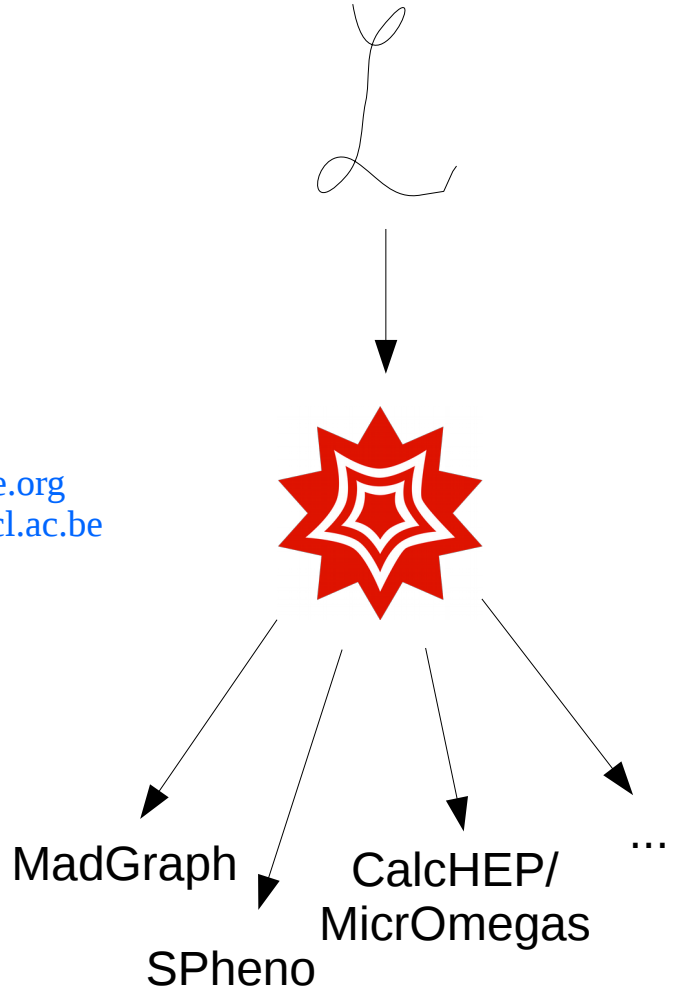
1) Come up with a cool new theory

2) Write down theory with Lagrangian level tools (LLTs)

(e.g. SARAH, FeynRules…)

sarah.hepforge.org
feynrules.irml.ucl.ac.be

3) LLTs create output for codes relating to Dark Matter, collider physics, flavour physics, spectrum calculators, decays…



and that's all in GAMBIT...

launchpad.net/mg5amcnlo
spheno.hepforge.org
lapth.cnrs.fr/micromegas/
theory.sinp.msu.ru/~pukhov/calchep.html

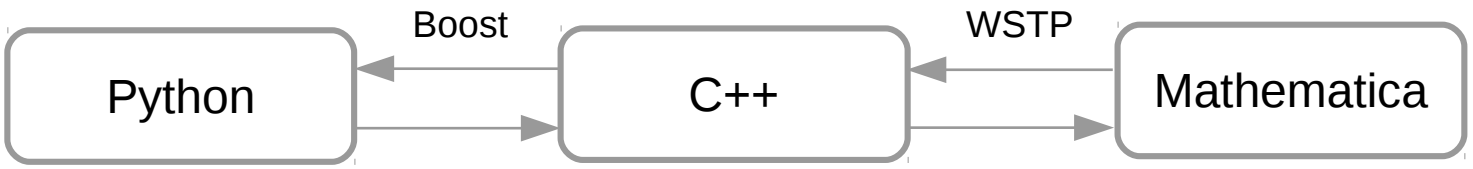
GUM: GAMBIT Universal Models

GUM is a **new tool** providing an interface between LLTs and GAMBIT.

GUM communicates with LLTs to:

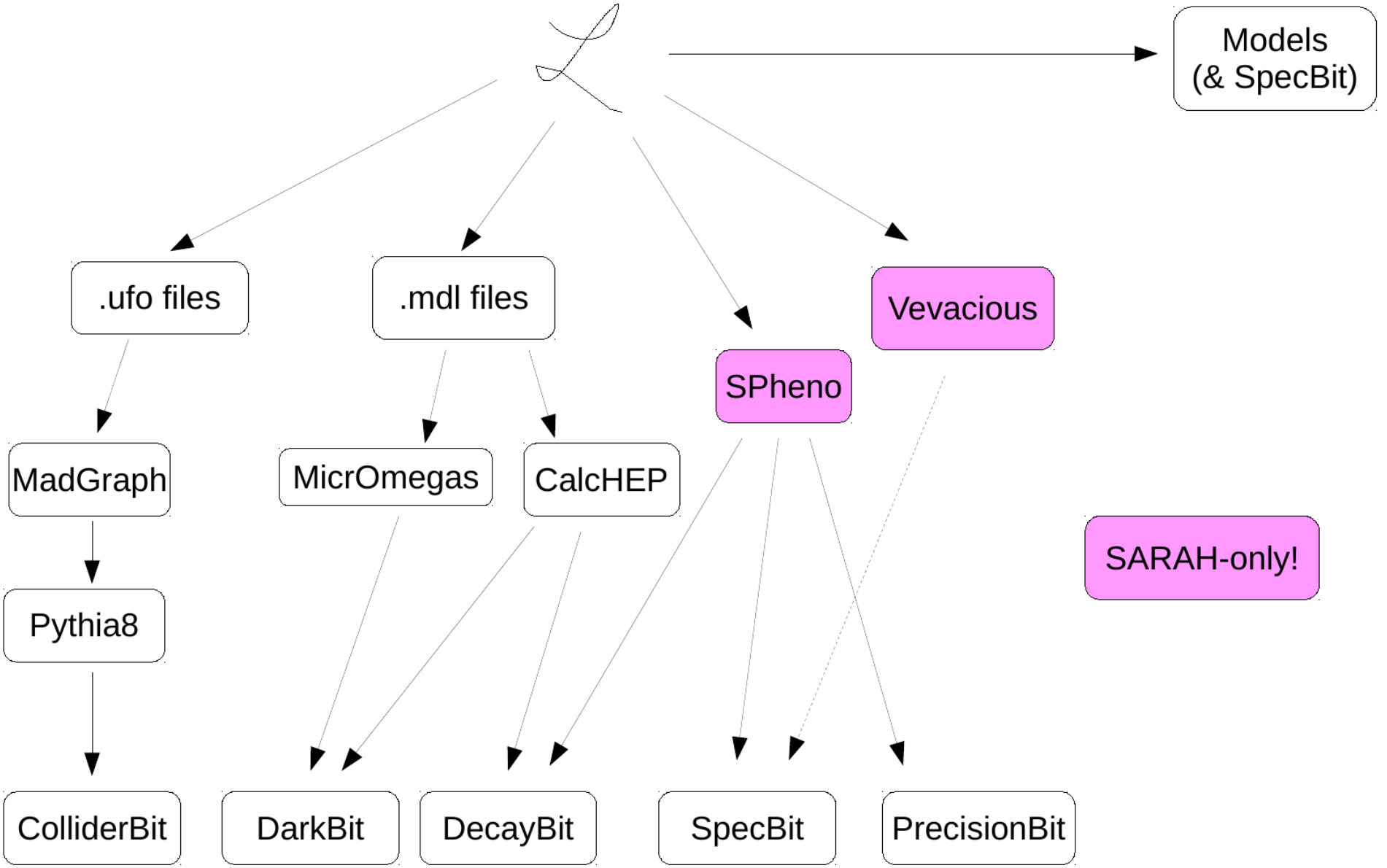
- Extract particle information and add them to the GAMBIT **particle database**
- Extract parameter information and add an entry to the **model hierarchy**
- Write output code for requested GAMBIT **backends** (& patch them appropriately to be GAMBIT-friendly)
- Automatically write **module functions** in GAMBIT for new observables and likelihoods
- And **automatically** interfaces everything in between: models, spectra, decays, ColliderBit 'model', ...

GUM: GAMBIT Universal Models



- Mostly written in **Python,** with C++ interface to Mathematica via **Wolfram Symbolic Transfer Protocol**
- Ships with all GAMBIT 2.0+ releases (currently 1.4), plus a whole host of new **backend codes**
- Easily extended to any new output of LLTs
- Most importantly – easy to use!

What can GUM (v1) do?



How does GUM work?

- User provides a simple .gum initialisation file…

```
## Minimal .gum file for the THDM-II model via SARAH

math:
  # Name of Mathematica package,
  # either 'feynrules' or 'sarah'
  package: sarah
  # Name of Model within that package.
  model: THDM-II

# Outputs for GUM to hook up to GAMBIT
output:
  spheno: true      # Spectrum generator, decays
  vevacious: true   # Vacuum stability
  pythia: false     # Collider physics
```

- ...and invokes GUM from the command line

```
sanjay@HP-ENVY:~/gumbit/gum$ ./gum -f gum_files/THDM_II.gum
```

- (Other options include: base models, WIMP DM candidate, collider processes, the Lagrangian, restriction files...)

How does GUM work?

- GUM writes source code for GAMBIT…

```
Now putting the new code into GAMBIT.
File ../Models/include/gambit/Models/models/THDM_II.hpp successfully created.
File ../Models/src/SpectrumContents/THDM_II.cpp successfully created.
File ../Models/include/gambit/Models/SimpleSpectra/THDM_IISimpleSpec.hpp successfully created.
File ../Models/include/gambit/Models/SpectrumContents/RegisteredSpectra.hpp successfully amended.
File ../SpecBit/src/SpecBit_THDM_II.cpp successfully created.
File ../SpecBit/include/gambit/SpecBit/SpecBit_THDM_II_rollcall.hpp successfully created.
File ../SpecBit/include/gambit/SpecBit/SpecBit_rollcall.hpp successfully amended.
File ../DecayBit/src/DecayBit.cpp successfully amended.
File ../DecayBit/include/gambit/DecayBit/DecayBit_rollcall.hpp successfully amended.
File ../DecayBit/include/gambit/DecayBit/DecayBit_rollcall.hpp successfully amended.
File ../DecayBit/include/gambit/DecayBit/DecayBit_rollcall.hpp successfully amended.
File ../DecayBit/include/gambit/DecayBit/DecayBit_rollcall.hpp successfully amended.
File ../DecayBit/include/gambit/DecayBit/DecayBit_rollcall.hpp successfully amended.
File ../DecayBit/include/gambit/DecayBit/DecayBit_rollcall.hpp successfully amended.
File ../DecayBit/src/DecayBit.cpp successfully amended.
File ../Backends/src/frontends/CalcHEP_3_6_27.cpp successfully amended.
File ../Backends/src/frontends/CalcHEP_3_6_27.cpp successfully amended.
File ../Backends/include/gambit/Backends/frontends/CalcHEP_3_6_27.hpp successfully amended.
File ../Backends/include/gambit/Backends/frontends/CalcHEP_3_6_27.hpp successfully amended.
File ../yaml_files/THDM_II.yaml successfully created.
File ../gum/THDM_II_config.sh successfully created.

Changes saved to mug_files/THDM_II.mug
If you need to reset GAMBIT, do:
        ./gum -r mug_files/THDM_II.mug

GUM has finished successfully!
Please (re)compile GAMBIT, by running THDM_II_config.sh
```

- Then the user can perform global fits of their new model!

Summary

- GUM provides a direct interface between Lagrangian level tools and GAMBIT
- Gives phenomenologists an easy way to perform global studies of BSM models – if you can write a SARAH/FeynRules file, you can add a model to GAMBIT
- Watch out for GAMBIT 2.0, coming soon
- New Cosmology module **CosmoBit** also nearing release

All results and samples from previous studies available on Zenodo
→ SUSY, axions, Higgs portal models, right-handed neutrinos…

GAMBIT code is public: gambit.hepforge.org