GAMBIT Tutorial

(a Global And Modular BSM Inference Tool)

Are Raklev & Anders Kvellestad, University of Oslo

Les Houches 2023 — June 28, 2023

UNIVERSITETET I OSLO

Outline

1. Global fits
2. GAMBIT
3. GAMBIT-light
4. GUM
5. Summary

1. Global fits

The basic steps of a BSM global fit

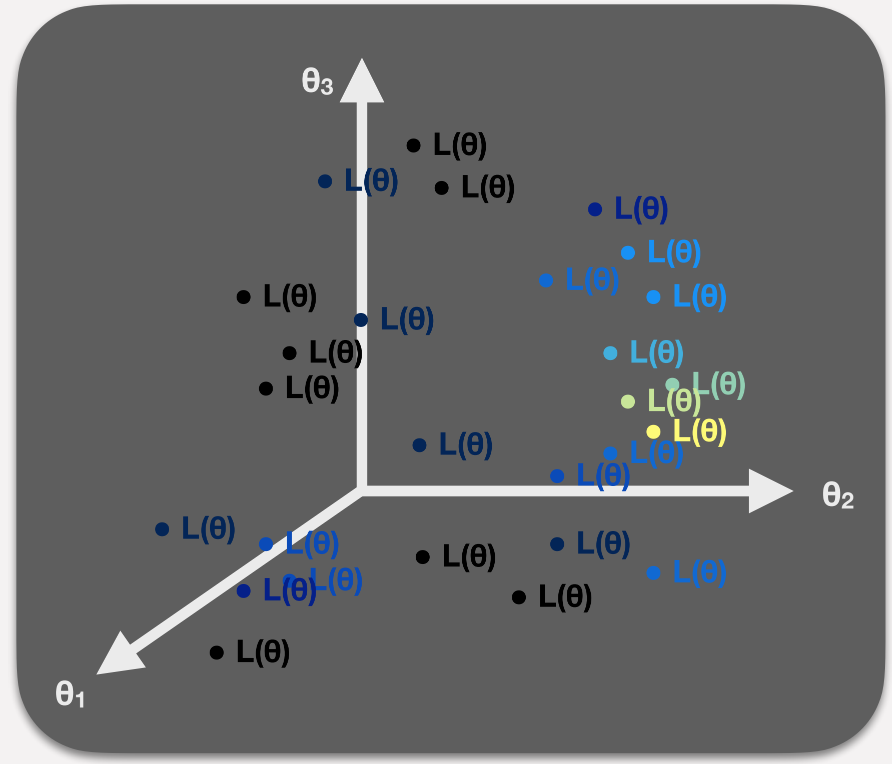
- Choose your **BSM theory and parameterisation**
- Construct the **joint likelihood function** including observables from collider physics, dark matter, flavour physics, cosmology, +++

$$\mathcal{L} = \mathcal{L}_{\text{collider}} \mathcal{L}_{\text{DM}} \mathcal{L}_{\text{flavor}} \mathcal{L}_{\text{EWPO}} \cdots$$

- Use (**sophisticated**) **scanning techniques** to:
  - Explore the model parameter space (θ_1, θ_2, θ_3, …)
  - At every point θ: **calculate predictions(θ) → evaluate joint likelihood L(θ)**
- From likelihood samples, carry out frequentist or Bayesian inference (parameter estimation, model comparison, …)

Computational challenges:

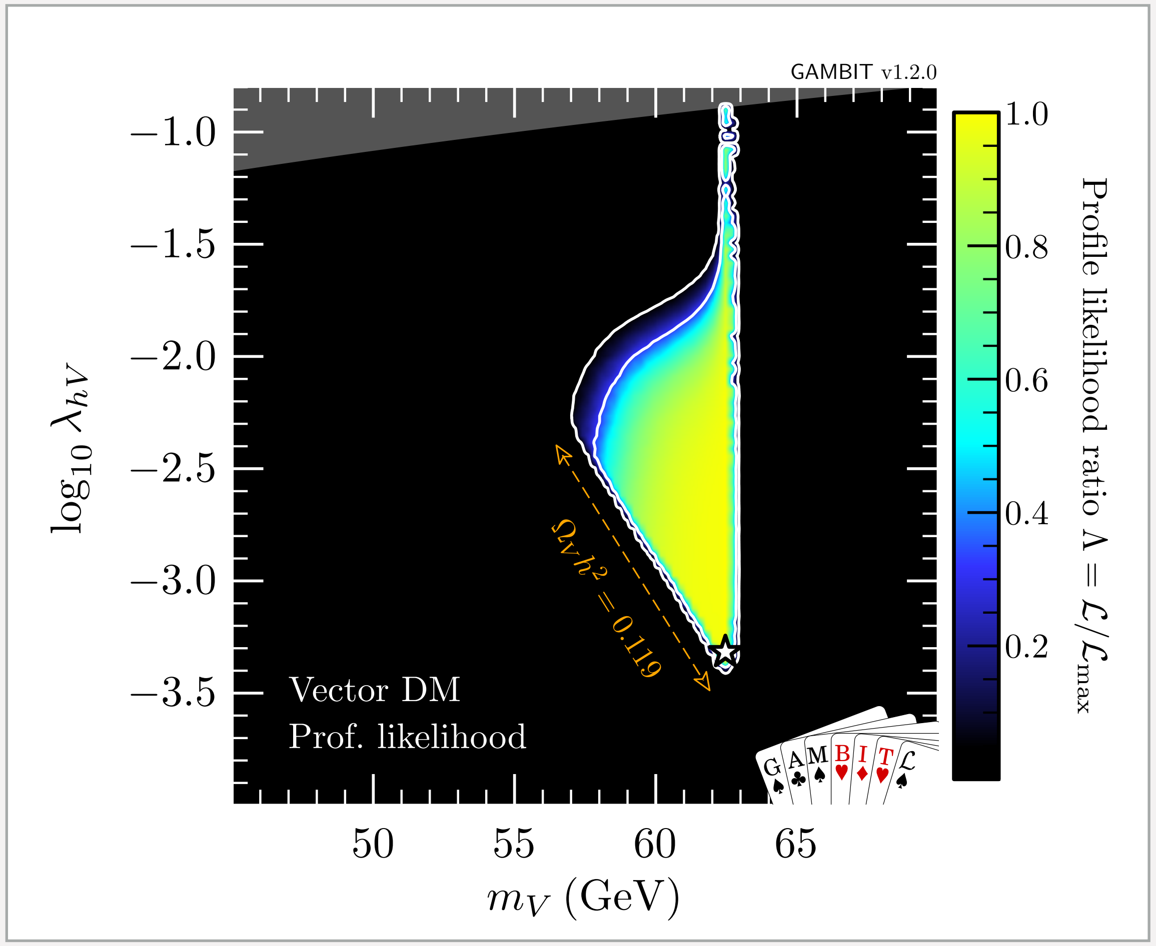
- Need **smart exploration** of parameter space
- Need **fast theory calculations**
- Need **fast simulations of experiments** (e.g. LHC)
- Need **sufficiently detailed likelihoods** or **full statistical models**

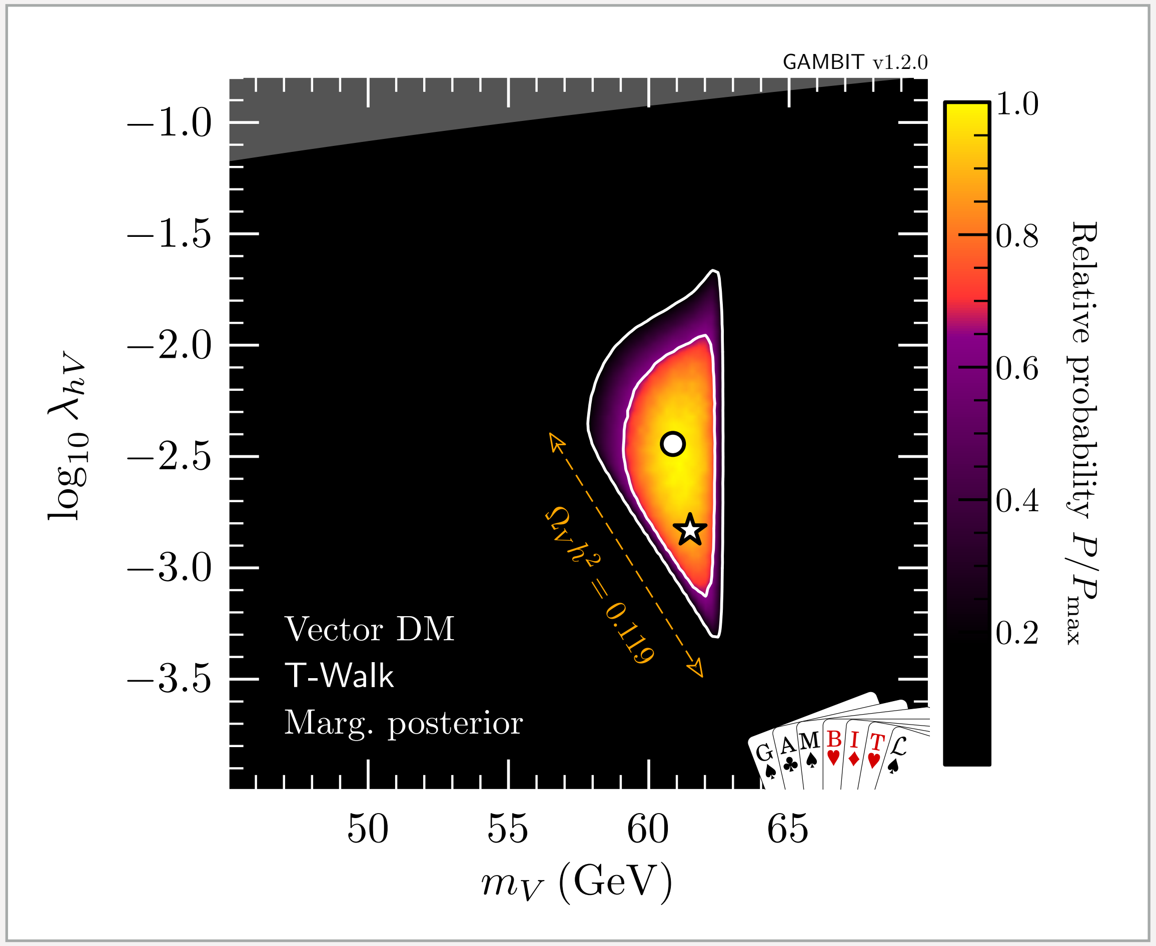


Some code infrastructure challenges:

- Need **different parameter scanning algorithms**
- Need **model-agnostic core framework**
- Need to interface ***many* external physics codes**
- Need **massive parallelisation**…
- …which implies a need for **diskless interfacing**
- …which implies a need to **stop external codes from calling STOP and kill your 10,000-CPU scan**… :)

Typical result:
Parameter estimation, presented as **profile likelihood** and/or **posterior density** plots





[arxiv:1808.10465]

2. GAMBIT

GAMBIT: The Global And Modular BSM Inference Tool

gambit.hepforge.org github.com/GambitBSM EPJC 77 (2017) 784 arXiv:1705.07908

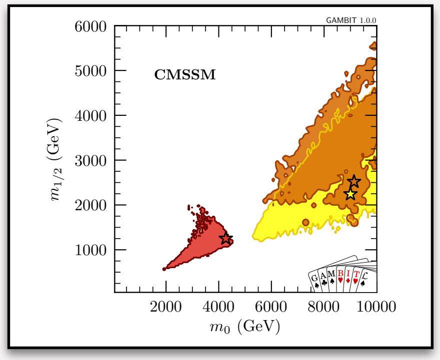
- Extensive model database, beyond SUSY
- Fast definition of new datasets, theories
- Extensive observable/data libraries
- Plug&play scanning/physics/likelihood packages
- Various statistical options (frequentist /Bayesian)
- Fast LHC likelihood calculator
- Massively parallel
- Fully open-source

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

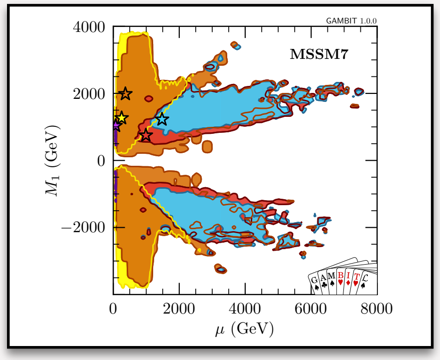
Authors of: BubbleProfiler, Capt'n General, Contur, DarkAges, DarkSUSY, DDCalc, DirectDM, Diver, EasyScanHEP, ExoCLASS, FlexibleSUSY, gamLike, GM2Calc, HEPLike, IsaTools, MARTY, nuLike, PhaseTracer, PolyChord, Rivet, SOFTSUSY, Superlso, SUSY-AI, xsec, Vevacious, WIMPSim

Recent collaborators: P Athron, C Balázs, A Beniwal, S Bloor, T Bringmann, A Buckley, J-E Camargo-Molina, C Chang, M Chrzaszcz, J Conrad, J Cornell, M Danninger, J Edsjö, T Emken, A Fowlie, T Gonzalo, W Handley, J Harz, S Hoof, F Kahlhoefer, A Kvellestad, P Jackson, D Jacob, C Lin, N Mahmoudi, G Martinez, MT Prim, A Raklev, C Rogan, R Ruiz, P Scott, N Serra, P Stöcker , W. Su, A Vincent, C Weniger, M White, Y Zhang, ++

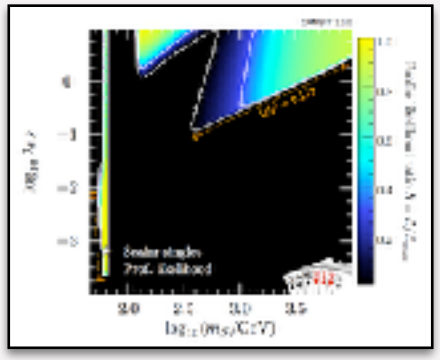
70+ participants in many experiments and numerous major theory codes



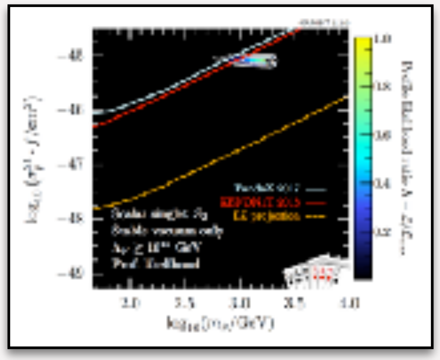
GUT-scale SUSY: 1705.07935



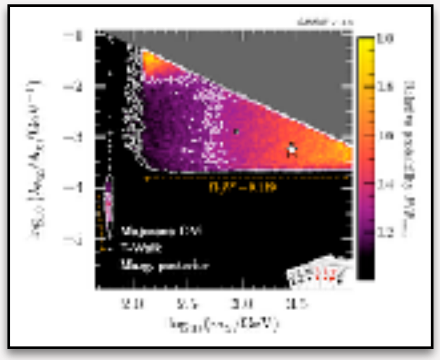
MSSM7: 1705.07917



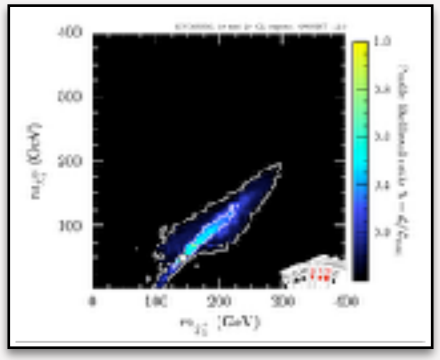
Scalar Higgs portal DM: 1705.07931



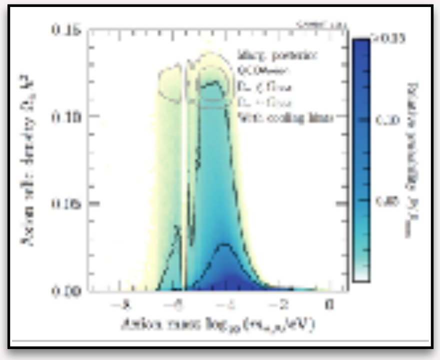
Scalar Higgs portal DM w/ vac. stability: 1806.11281



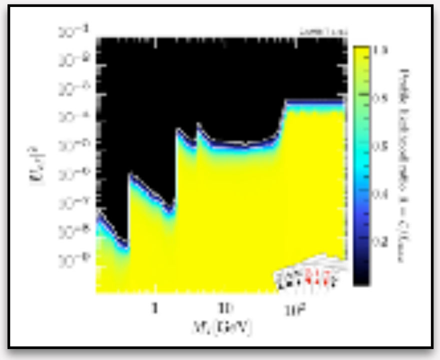
Vector and fermion Higgs portal DM: 1808.10465



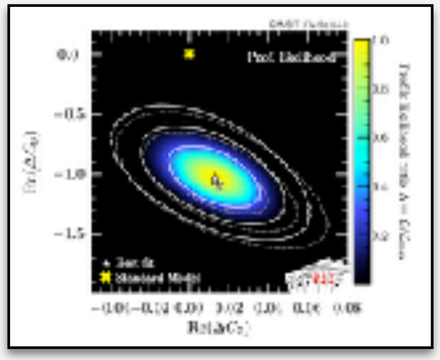
EW-MSSM: 1809.02097



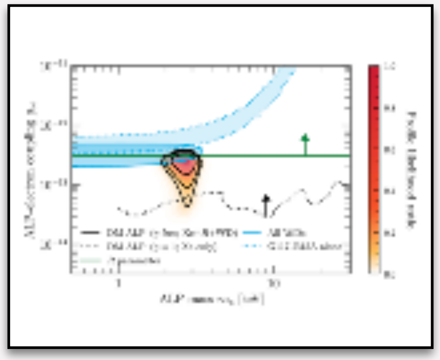
Axion-like particles: 1810.07192



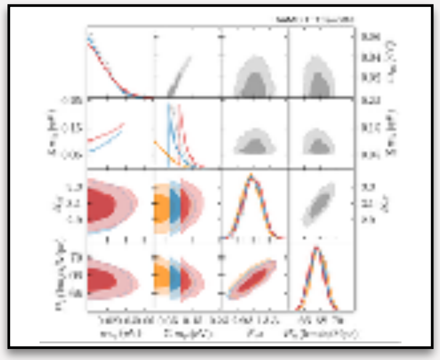
Right-handed neutrinos: 1908.02302



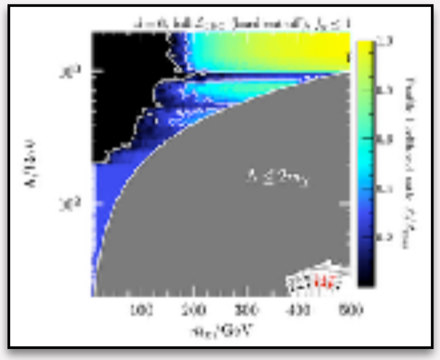
Flavour EFT: 2006.03489



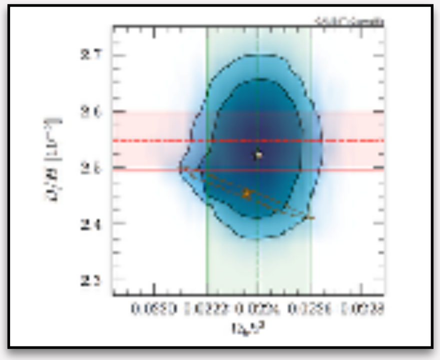
More axion-like particles: 2007.05517



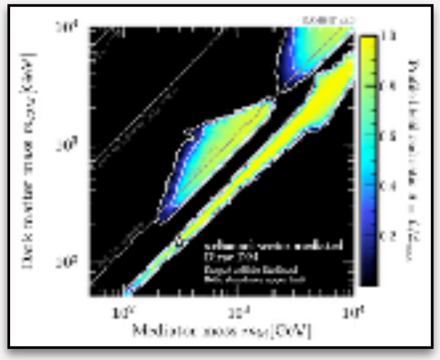
Neutrinos and cosmo: 2009.03287



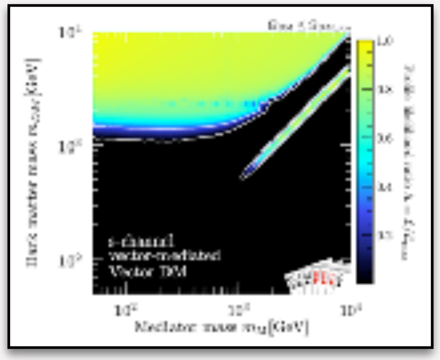
Dark matter EFTs: 2106.02056



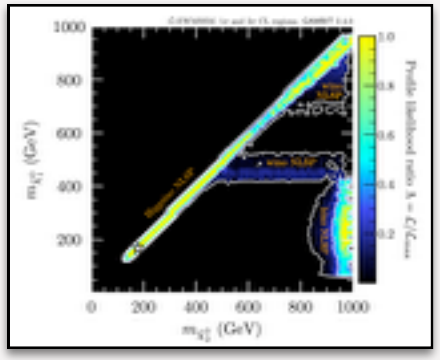
Cosmo ALPs: 2205.13549



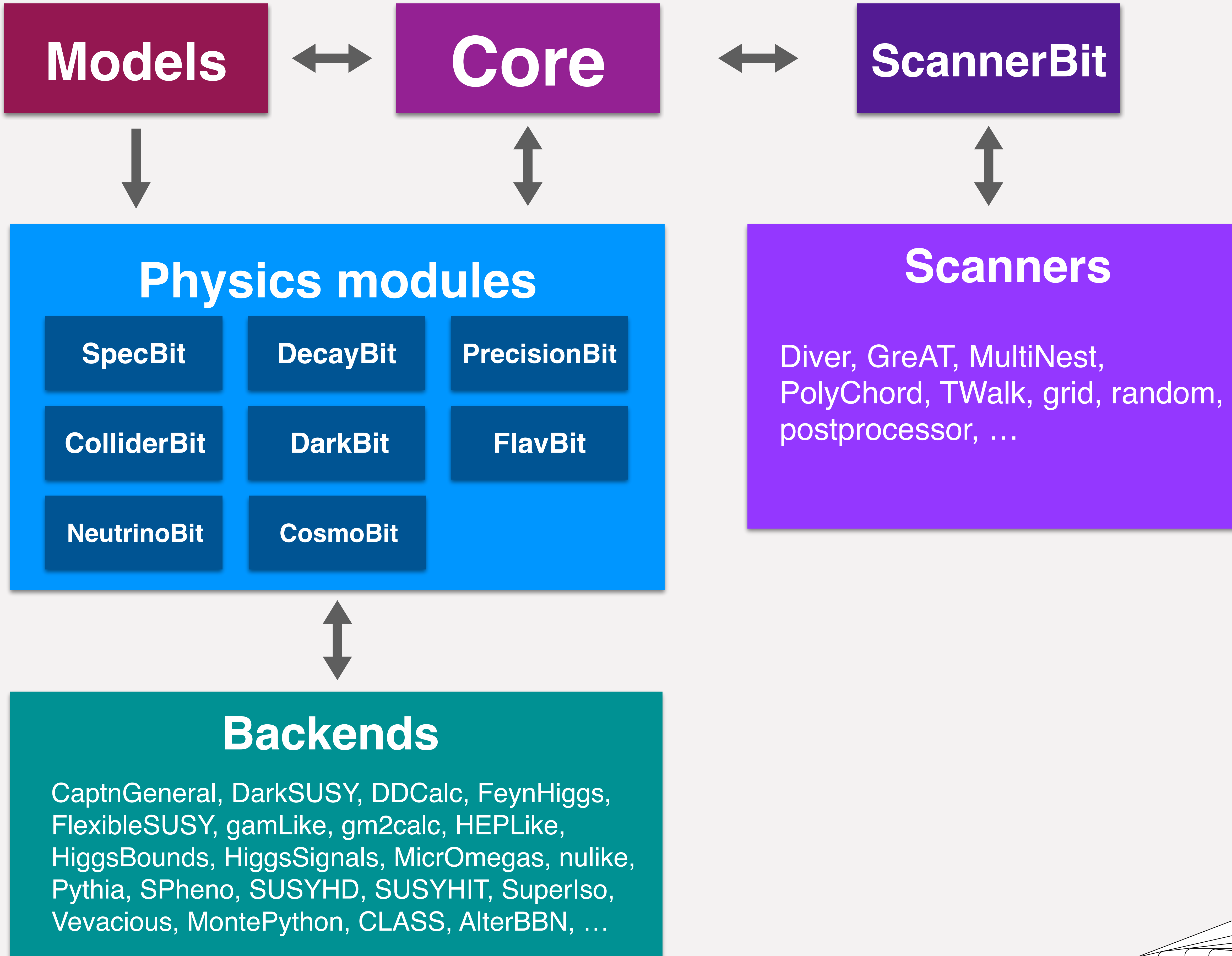
Simplified DM, scalar/fermion: 2209.13266



Simplified DM, vector: 2303.08351



EW-MSSM w/ light gravitino: 2303.09082


Models
Core
ScannerBit
Physics modules
SpecBit
DecayBit
PrecisionBit
ColliderBit
DarkBit
FlavBit
NeutrinoBit
CosmoBit
Scanners
Diver, GreAT, MultiNest, PolyChord, TWalk, grid, random, postprocessor, …
Backends
CaptnGeneral, DarkSUSY, DDCalc, FeynHiggs, FlexibleSUSY, gamLike, gm2calc, HEPLike, HiggsBounds, HiggsSignals, MicrOmegas, nulike, Pythia, SPheno, SUSYHD, SUSYHIT, SuperIso, Vevacious, MontePython, CLASS, AlterBBN, …

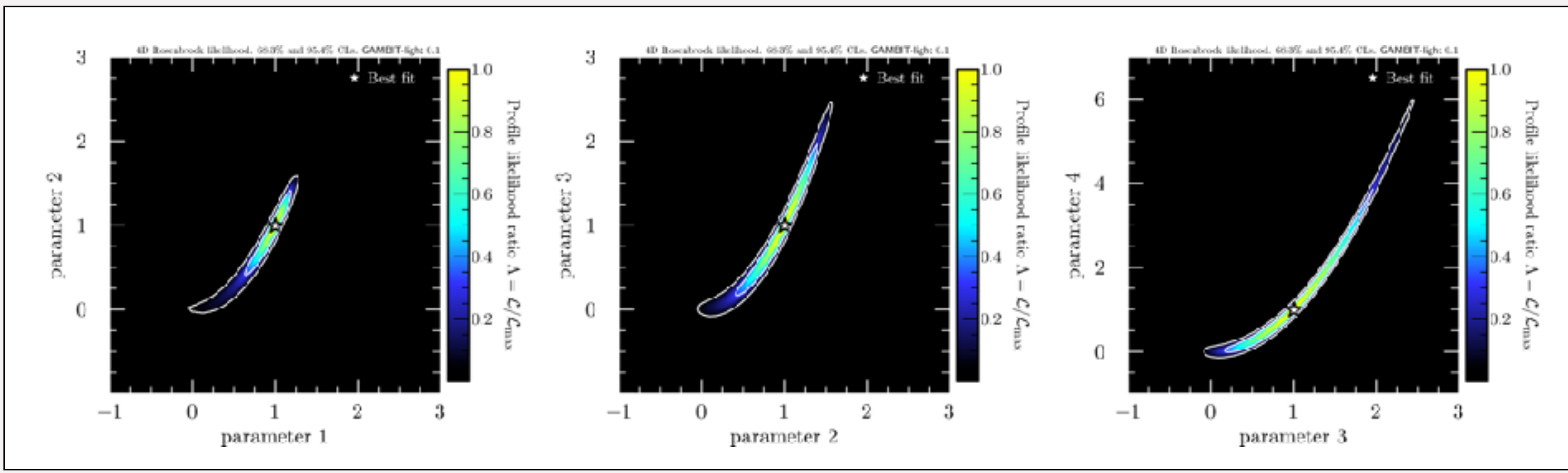
Some basic technical features

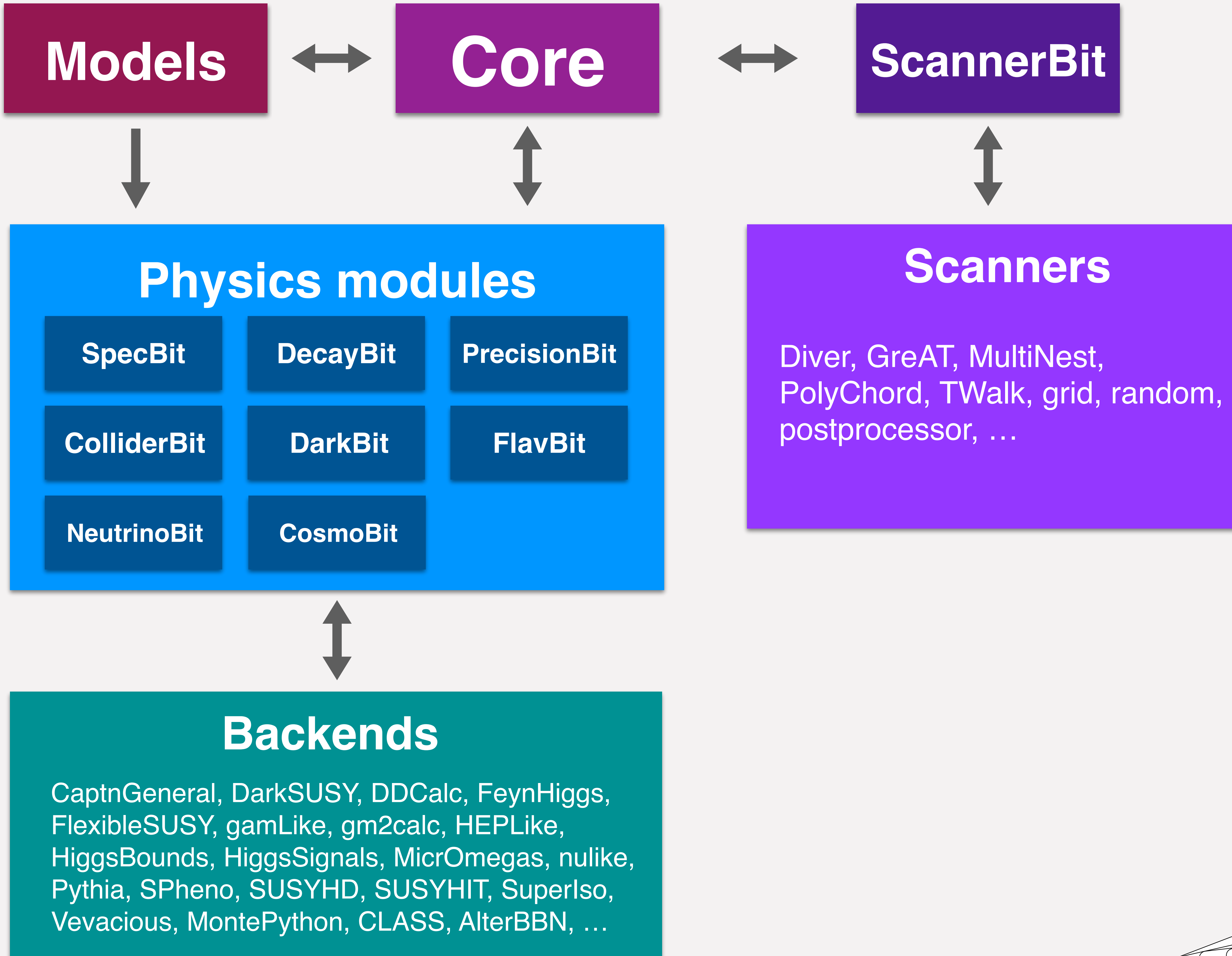
- **Two-level parallelisation**:
  - **MPI** for parameter sampling algorithm
  - **OpenMP** for per-point physics computations
- Collection of **sampling algorithms as plug-ins** (scanners)
  - Coming soon: plug in your own python sampling code
- Backend system for using **C**, **C++**, **Fortran**, **Python** and **Mathematica** codes as **runtime plug-ins** for physics computations
- Run configuration through **YAML** input file
- **Dynamic dependency resolution**: order of computations not hard-coded

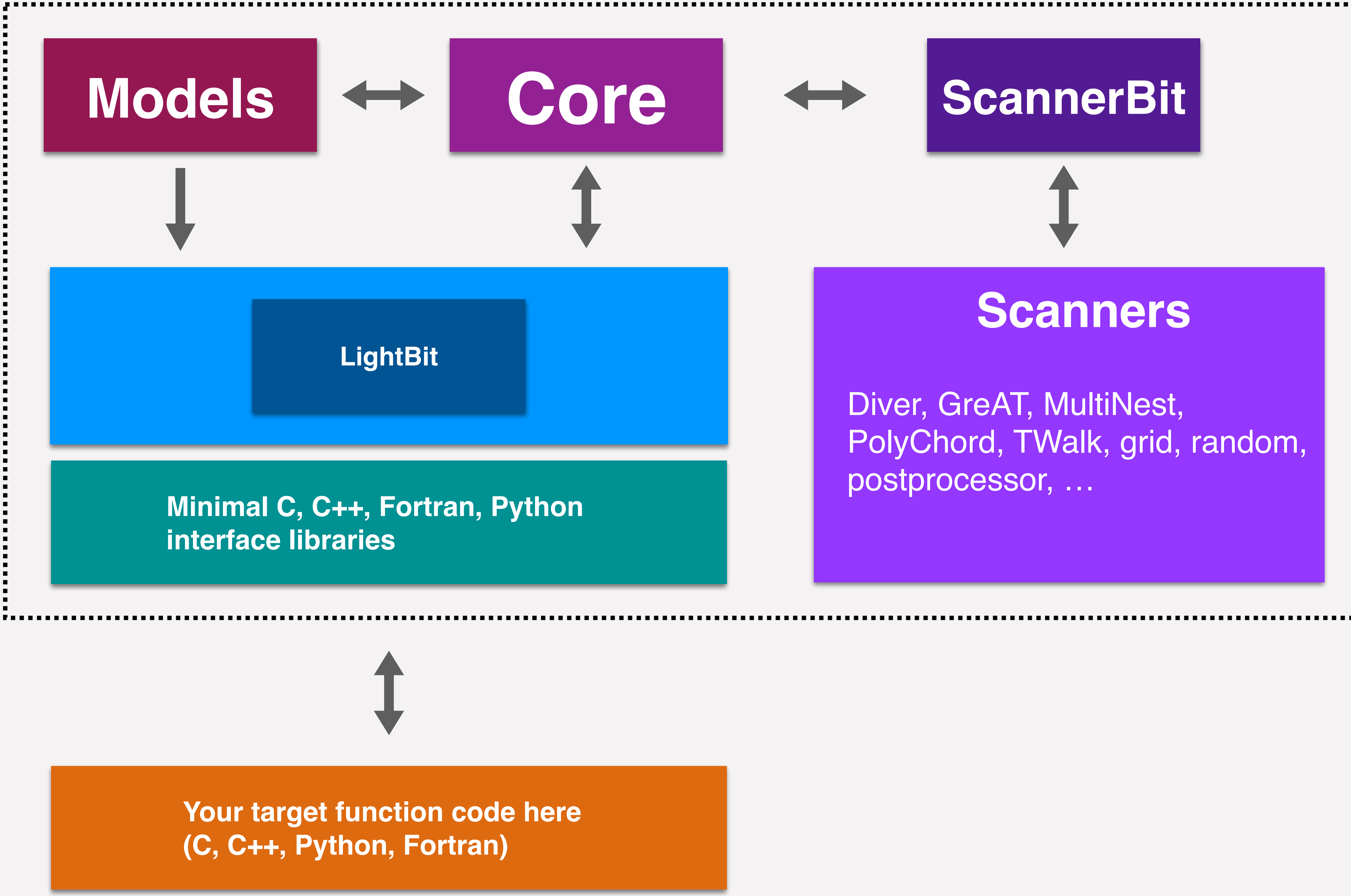
3. GAMBIT-light

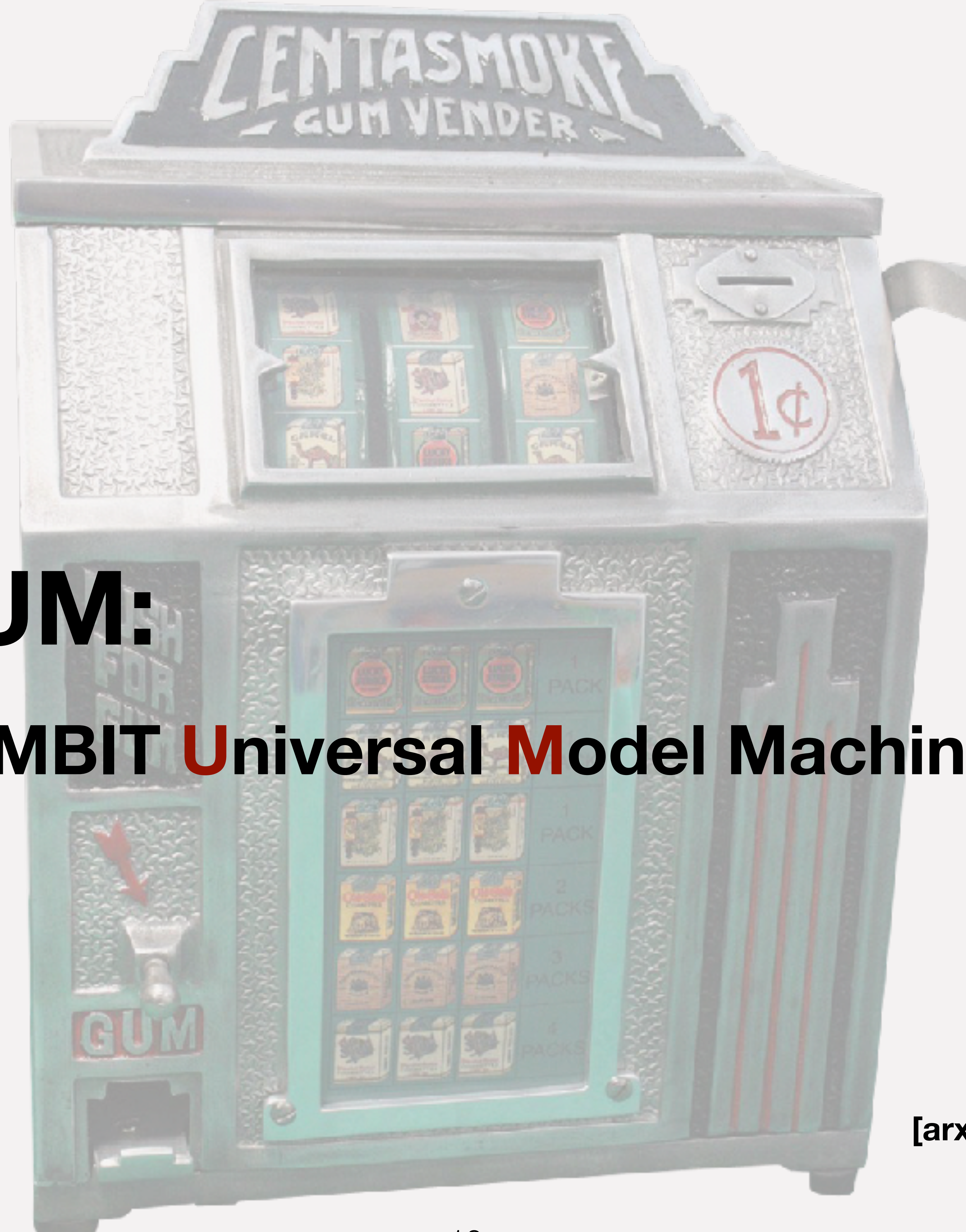
GAMBIT-light

- **GAMBIT** can be used **beyond particle physics**
- At its core: A general tool for **computationally heavy optimisation and parameter estimation** tasks
- Coming soon: **GAMBIT-light**
  A lightweight GAMBIT without the particle physics




Models
Core
ScannerBit
Physics modules
SpecBit
DecayBit
PrecisionBit
ColliderBit
DarkBit
FlavBit
NeutrinoBit
CosmoBit
Scanners
Diver, GreAT, MultiNest, PolyChord, TWalk, grid, random, postprocessor, …
Backends
CaptnGeneral, DarkSUSY, DDCalc, FeynHiggs, FlexibleSUSY, gamLike, gm2calc, HEPLike, HiggsBounds, HiggsSignals, MicrOmegas, nulike, Pythia, SPheno, SUSYHD, SUSYHIT, SuperIso, Vevacious, MontePython, CLASS, AlterBBN, …


Models
Core
ScannerBit
LightBit
Scanners
Diver, GreAT, MultiNest, PolyChord, TWalk, grid, random, postprocessor, …
Minimal C, C++, Fortran, Python interface libraries
Your target function code here (C, C++, Python, Fortran)

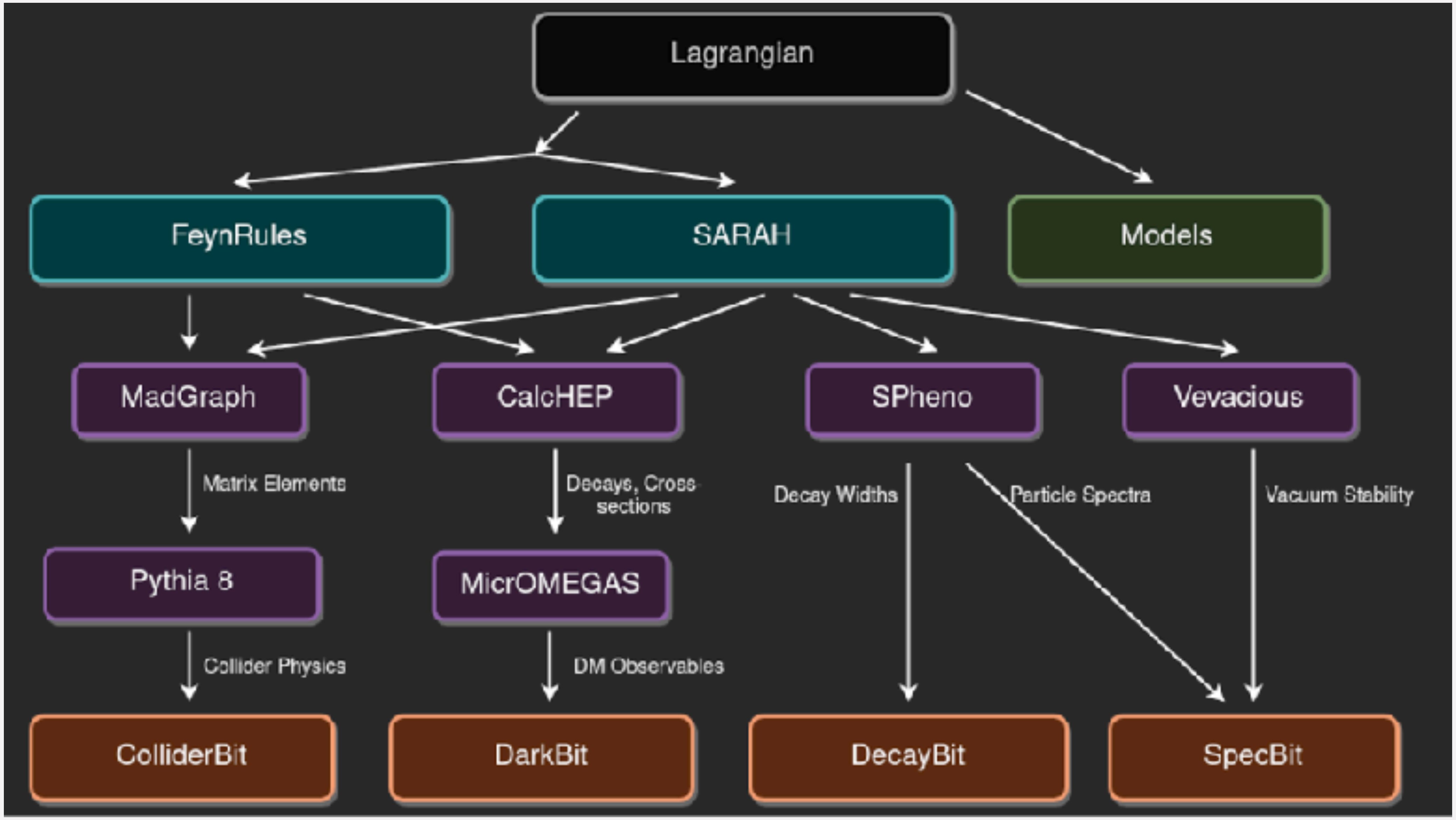


3. GUM:

the GAMBIT Universal Model Machine

[arxiv:2107.00030]

GUM: the GAMBIT Universal Model Machine



[Figure from Christopher Chang]

- From **Lagrangian** to a **GAMBIT global fit**
- The major addition in **GAMBIT 2.0**
- **Runs existing BSM tool chains** to generate model-specific physics libraries
- **Generates interfaces** for these libraries to the relevant Bits in GAMBIT
- **Generates additional GAMBIT code** (model definition, particle database additions, …)

GUM: the GAMBIT Universal Model Machine

Generated GAMBIT backends	FeynRules	SARAH	Usage in GAMBIT
CalcHEP	✓	✓	Decays, cross-sections
micrOMEGAs (via CalcHEP)	✓	✓	DM observables
Pythia (via MadGraph)	✓	✓	Collider physics
SPheno	✗	✓	Particle mass spectra, decay widths
Vevacious	✗	✓	Vacuum stability

From **FeynRules**

- Any Lagrangian (including EFTs), works at tree level
- CalcHEP
- micrOMEGAS (via CalcHEP)
- Pythia (via MadGraph)

From **SARAH**

- Renormalizable theories, one-loop
- CalcHEP
- micrOMEGAS (via CalcHEP)
- Pythia (via MadGraph)
- SPheno
- Vevacious
- + input for existing HiggsBounds + HiggsSignals backends (via SARAH-SPheno)

Summary

Summary

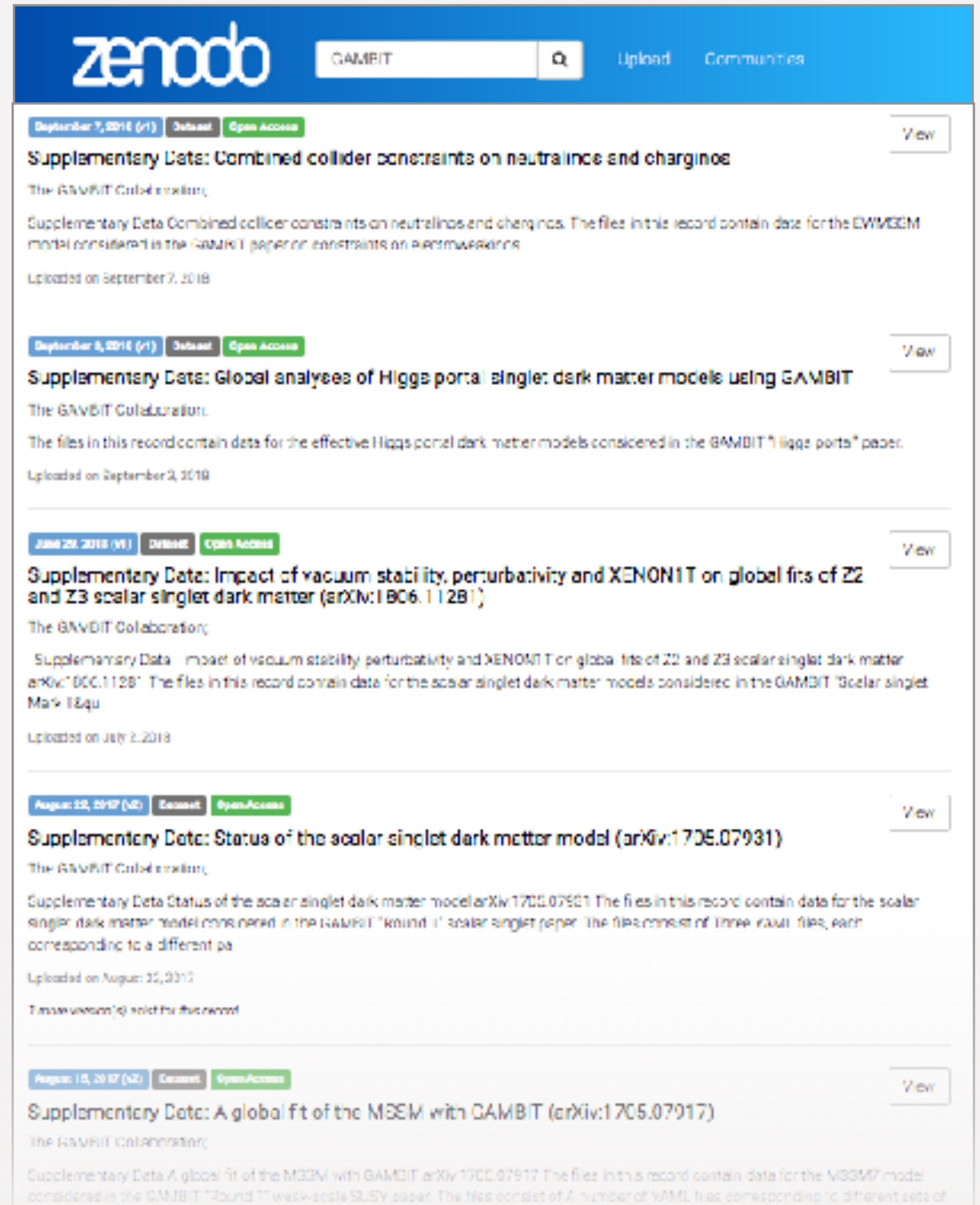
- **GAMBIT** is an open-source tool for **large-scale global fits** of new theories in particle physics
- A **modular** and **model-independent** core software framework
  → GAMBIT has been used to investigate a wide range of new theories
- Recent development: **GUM — the GAMBIT Universal Model machine**, allows the user to start from a Lagrangian model definition
- Coming soon: **GAMBIT-light**
- New webpage & GitHub: gambitbsm.org and github.com/GambitBSM/gambit_2.4
- GAMBIT Community results are publicly available: zenodo.org/communities/gambit-official

All GAMBIT Community results publicly available

Results available on zenodo.cern.ch

- Parameter point samples (hdf5 files)
- GAMBIT input files for all scans
- Example plotting routines

Links at gambitbsm.org/community/publications/



Bonus tracks

Model defined in a FeynRules/SARAH file

$$\mathcal{L} = \mathcal{L}_{\rm SM} + \frac{1}{2}\overline{\chi}\left(i\partial\!\!\!/ - m_\chi\right)\chi + \frac{1}{2}\partial_\mu Y \partial^\mu Y - \frac{1}{2}m_Y^2 Y^2 - \frac{g_\chi}{2}\overline{\chi}\chi Y - \frac{c_Y}{2}\sum_f y_f \overline{f} f Y \,.$$

Model defined in a FeynRules/SARAH file

$$\mathcal{L} = \mathcal{L}_{\rm SM} + \frac{1}{2}\overline{\chi}\left(i\partial\!\!\!/ - m_\chi\right)\chi + \frac{1}{2}\partial_\mu Y \partial^\mu Y - \frac{1}{2}m_Y^2 Y^2 - \frac{g_\chi}{2}\overline{\chi}\chi Y - \frac{c_Y}{2}\sum_f y_f \overline{f} f Y\,.$$

Write a .gum file

```
math:
  # Choose FeynRules
  package: feynrules
  # Name of the model
  model: MDMSM
  # Model builds on the Standard Model FeynRules file
  base_model: SM
  # The Lagrangian is defined by the DM sector (LDM),
  # defined in MDMSM.fr, plus the SM Lagrangian (LSM)
  # imported from the 'base model', SM.fr
  Lagrangian: LDM + LSM
  # Make CKM matrix = identity to simplify output
  restriction: DiagonalCKM

# PDG code of the annihilating DM candidate
# in the FeynRules file
wimp_candidate: 52

# Select outputs for DM physics.
# Collider physics is not as important in this model.
output:
  pythia: false
  calchep: true
  micromegas: true
```

Model defined in a FeynRules/SARAH file

$$\mathcal{L} = \mathcal{L}_{\text{SM}} + \frac{1}{2}\overline{\chi}\left(i\partial\!\!\!/ - m_\chi\right)\chi + \frac{1}{2}\partial_\mu Y \partial^\mu Y - \frac{1}{2}m_Y^2 Y^2 - \frac{g_\chi}{2}\overline{\chi}\chi Y - \frac{c_Y}{2}\sum_f y_f \overline{f} f Y\,.$$

Write a .gum file

```
math:
  # Choose FeynRules
  package: feynrules
  # Name of the model
  model: MDMSM
  # Model builds on the Standard Model FeynRules file
  base_model: SM
  # The Lagrangian is defined by the DM sector (LDM),
  # defined in MDMSM.fr, plus the SM Lagrangian (LSM)
  # imported from the 'base model', SM.fr
  Lagrangian: LDM + LSM
  # Make CKM matrix = identity to simplify output
  restriction: DiagonalCKM

# PDG code of the annihilating DM candidate
# in the FeynRules file
wimp_candidate: 52

# Select outputs for DM physics.
# Collider physics is not as important in this model.
output:
  pythia: false
  calchep: true
  micromegas: true
```

Run GUM

```
./gum -f Tutorial/MDMSM.gum
```

Compile GAMBIT + backends

```
cd ../build
cmake ..
make micromegas_MDMSM
make calchep
make -jn gambit
```

Model d

$\mathcal{L} = \mathcal{L}_{\rm SM}$

$-\frac{g_\chi}{2}\overline{\chi}\chi$

```
FeynRules file seems ok; firing up a Mathematica kernel...
Calling FeynRules with model MDMSM...
The environment is initialized successfully...
WSTP link started
Loading FeynRules... FeynRules loaded from /home/sanjay/GAMBIT-2.0.0-alpha-1/gum/contrib/FeynRules.
Loading model MDMSM in FeynRules, piggybacking off of SM...
Model MDMSM loaded successfully, with model name Fermion DM with scalar mediator.
Attempting to load restriction DiagonalCKM...
Found restriction file at /home/sanjay/GAMBIT-2.0.0-alpha-1/gum/contrib/FeynRules/Models/SM/DiagonalCKM.rst
Restriction DiagonalCKM loaded successfully.
Checking the Lagrangian... you have specified the following: LDM + LSM
Lagrangian seems OK...
Checking the model is Hermitian... Your Lagrangian is Hermitian.
Checking kinetic and mass terms are properly diagonalised...
Kinetic terms are diagonal... Mass terms are diagonal... All good.
Extracting particles from FeynRules model.
Found 18 particle sets.
Extracting external parameters from FeynRules model.
Found 3 parameter blocks.
Writing CalcHEP output.
Setting Feynman Gauge.
CalcHEP files written.
WSTP link closed successfully.
Finished extracting parameters from feynrules.
CalcHEP files moved to GAMBIT Backends directory.
CalcHEP model files cleaned!

Finished running external codes...
Now attempting to write proposed GAMBIT code.

The following particles (by PDG code) are missing from the particle database: [52, 99902]. GUM is now adding them to
../config/particle_database.yaml.

Adding new model MDMSM to GAMBIT.
Writing new spectrum, MDMSM_spectrum
Writing CalcHEP module functions for DecayBit
Writing new module functions for DarkBit
Writing micrOMEGAs interface for DarkBit.
Writing basic container SpecBit interface...

Now putting the new code into GAMBIT.
File ../Models/include/gambit/Models/models/MDMSM.hpp successfully created.
File ../Models/src/SpectrumContents/MDMSM.cpp successfully created.
File ../Models/include/gambit/Models/SimpleSpectra/MDMSMSimpleSpec.hpp successfully created.
File ../Models/include/gambit/Models/SpectrumContents/RegisteredSpectra.hpp successfully amended.
File ../SpecBit/src/SpecBit_MDMSM.cpp successfully created.
File ../SpecBit/include/gambit/SpecBit/SpecBit_MDMSM_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/src/DecayBit.cpp successfully amended.
File ../DarkBit/include/gambit/DarkBit/DarkBit_rollcall.hpp successfully amended.
File ../DarkBit/include/gambit/DarkBit/DarkBit_rollcall.hpp successfully amended.
File ../DarkBit/src/MDMSM.cpp successfully created.
File ../DarkBit/include/gambit/DarkBit/DarkBit_rollcall.hpp successfully amended.
Model MDMSM added to capability RD_spectrum.
Model MDMSM added to capability RD_eff_annrate.
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.
```

```
l FeynRules file

DM sector (LDM),
Lagrangian (LSM)
M.fr

plify output

didate

t in this model.
```

Model defined in a FeynRules/SARAH file

$$\mathcal{L} = \mathcal{L}_{\rm SM} + \frac{1}{2}\overline{\chi}\left(i\partial\!\!\!/ - m_\chi\right)\chi + \frac{1}{2}\partial_\mu Y \partial^\mu Y - \frac{1}{2}m_Y^2 Y^2 - \frac{g_\chi}{2}\overline{\chi}\chi Y - \frac{c_Y}{2}\sum_f y_f \overline{f} f Y\,.$$

Write a .gum file

```
math:
  # Choose FeynRules
  package: feynrules
  # Name of the model
  model: MDMSM
  # Model builds on the Standard Model FeynRules file
  base_model: SM
  # The Lagrangian is defined by the DM sector (LDM),
  # defined in MDMSM.fr, plus the SM Lagrangian (LSM)
  # imported from the 'base model', SM.fr
  Lagrangian: LDM + LSM
  # Make CKM matrix = identity to simplify output
  restriction: DiagonalCKM

# PDG code of the annihilating DM candidate
# in the FeynRules file
wimp_candidate: 52

# Select outputs for DM physics.
# Collider physics is not as important in this model.
output:
  pythia: false
  calchep: true
  micromegas: true
```

Run GUM

```
./gum -f Tutorial/MDMSM.gum
```

Compile GAMBIT + backends

```
cd ../build
cmake ..
make micromegas_MDMSM
make calchep
make -jn gambit
```

Model defined in a FeynRules/SARAH file

$$\mathcal{L} = \mathcal{L}_{\rm SM} + \frac{1}{2}\overline{\chi}\left(i\partial\!\!\!/ - m_\chi\right)\chi + \frac{1}{2}\partial_\mu Y \partial^\mu Y - \frac{1}{2}m_Y^2 Y^2$$
$$- \frac{g_\chi}{2}\overline{\chi}\chi Y - \frac{c_Y}{2}\sum_f y_f \overline{f} f Y\,.$$

Write a .gum file

```
math:
  # Choose FeynRules
  package: feynrules
  # Name of the model
  model: MDMSM
  # Model builds on the Standard Model FeynRules file
  base_model: SM
  # The Lagrangian is defined by the DM sector (LDM),
  # defined in MDMSM.fr, plus the SM Lagrangian (LSM)
  # imported from the 'base model', SM.fr
  Lagrangian: LDM + LSM
  # Make CKM matrix = identity to simplify output
  restriction: DiagonalCKM

# PDG code of the annihilating DM candidate
# in the FeynRules file
wimp_candidate: 52

# Select outputs for DM physics.
# Collider physics is not as important in this model.
output:
  pythia: false
  calchep: true
  micromegas: true
```

Run GUM

```
./gum -f Tutorial/MDMSM.gum
```

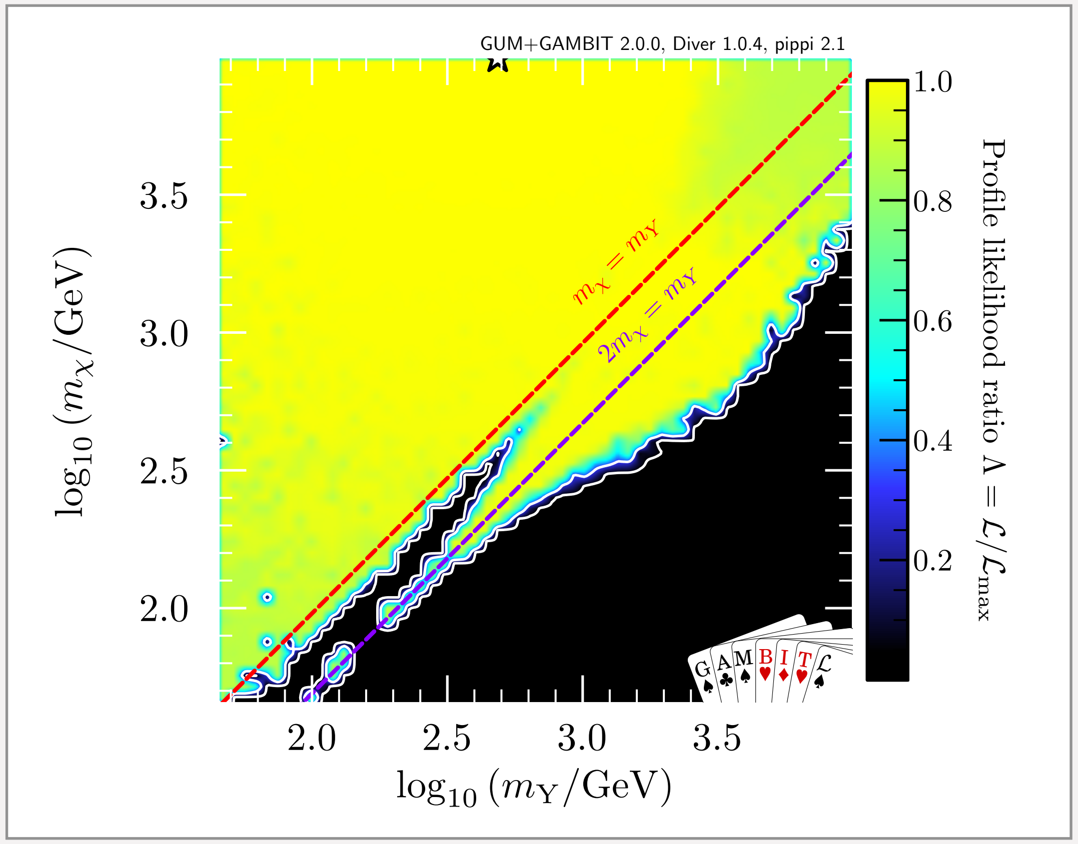
Compile GAMBIT + backends

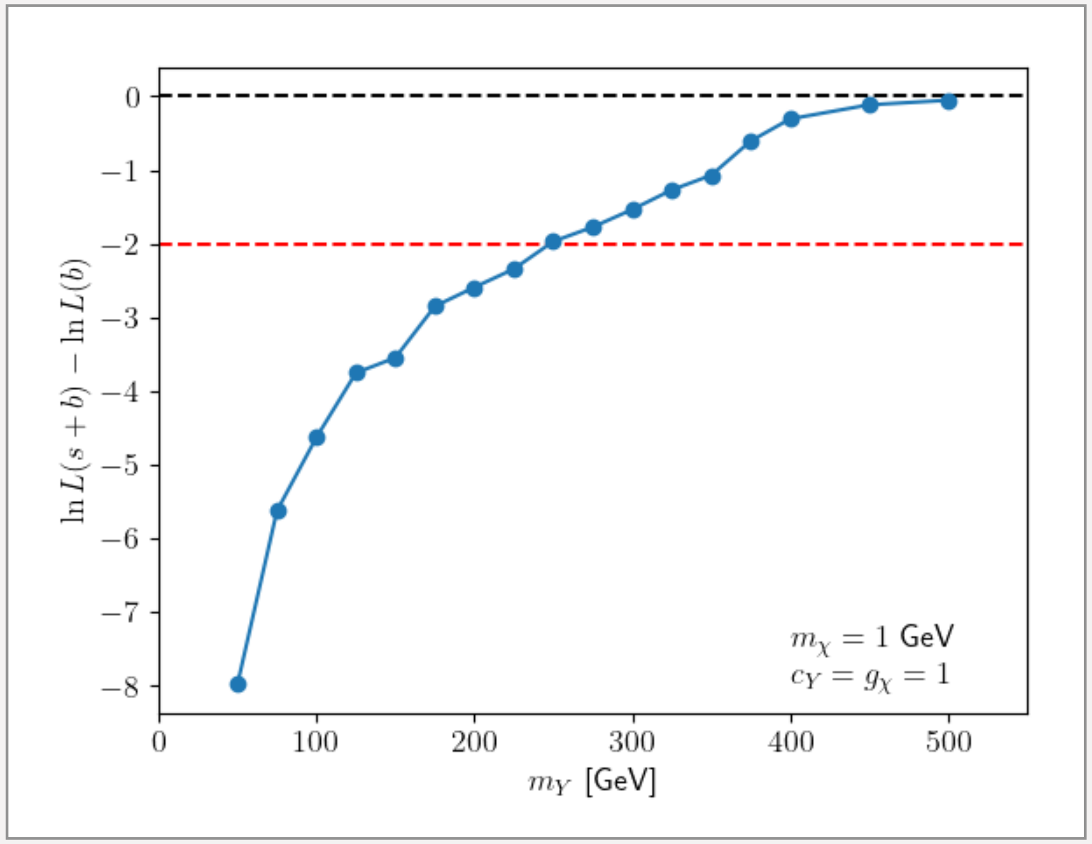
```
cd ../build
cmake ..
make micromegas_MDMSM
make calchep
make -jn gambit
```

Adjust GAMBIT input file

```
# Our dark matter model, implemented by GUM
MDMSM:
  mchi:
    range: [45, 10000]
    prior_type: log
  mY:
    range: [45, 10000]
    prior_type: log
  gchi:
    range: [1e-4, 12.566]
    prior_type: log
  cY:
    range: [1e-4, 12.566]
    prior_type: log

# Default halo parameters for the example
Halo_gNFW_rho0:
  rho0: 0.3
  v0: 240
  vesc: 533
  vrot: 240
  rs: 20.0
  r_sun: 8.5
  alpha: 1
  beta: 3
  gamma: 1
```

Run GAMBIT!





- **4D** scan (m_X, m_Y, g_X, c_Y)
- **Relic abundance** (as upper bound) [micrOMEGAs]
- **Direct detection**: XENON1T 2018, LUX 2016 [micrOMEGAs, DDCalc]
- **Indirect detection**: Fermi-LAT dwarf galaxies [CalcHEP, DarkSUSY, gamLike]
- **~11 hours on 4-core laptop**, sampling ~300k parameter points [Diver]

- Same model
- **1D** scan of m_Y
- $m_X = 1$ GeV, $g_X = 1$, $c_Y = 1$
- **Collider:** ATLAS 2lep+jets+MET, 139 fb^{-1} [Pythia, ColliderBit]
- Light m_Y disfavoured, but can easily be accommodated in the larger 4D parameter space

Understanding the full implications of [experimental] searches requires the interpretation of the experimental results in the context of many more theoretical models than are currently explored at the time of publication.

HEP Software Foundation [arxiv:1712.06982]

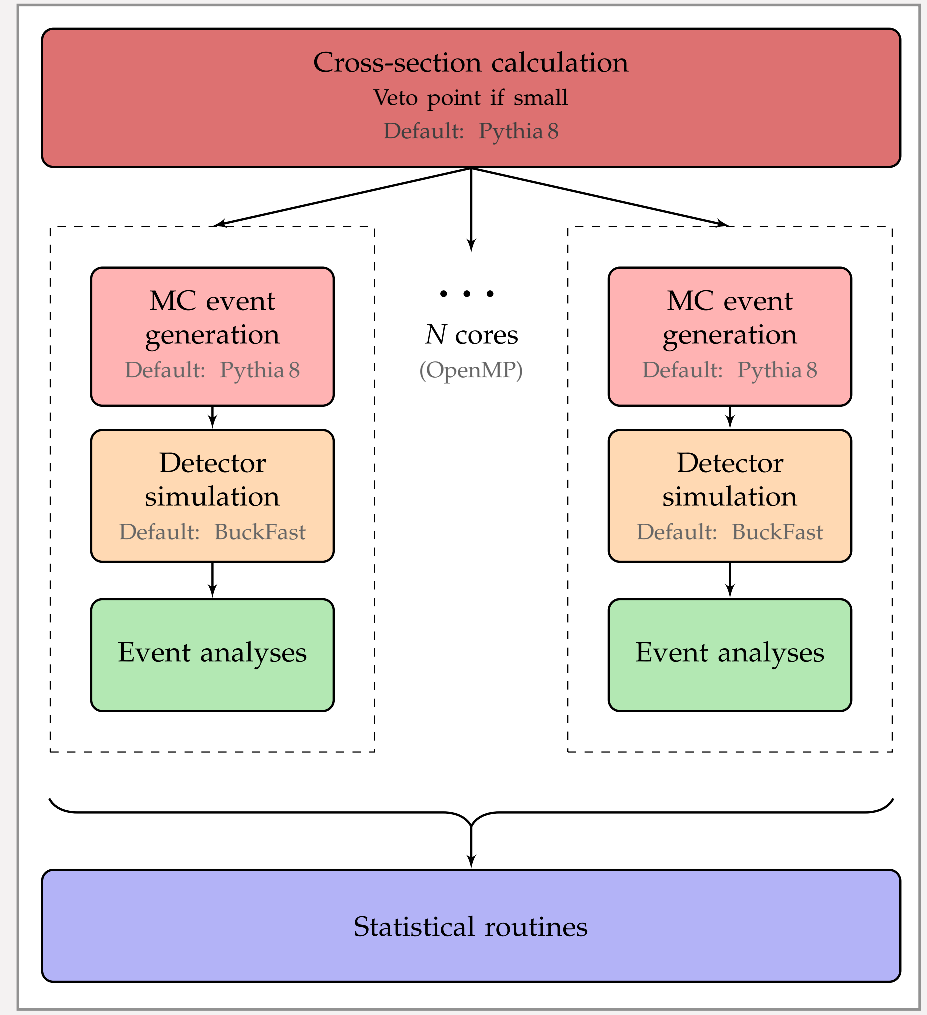
See also:

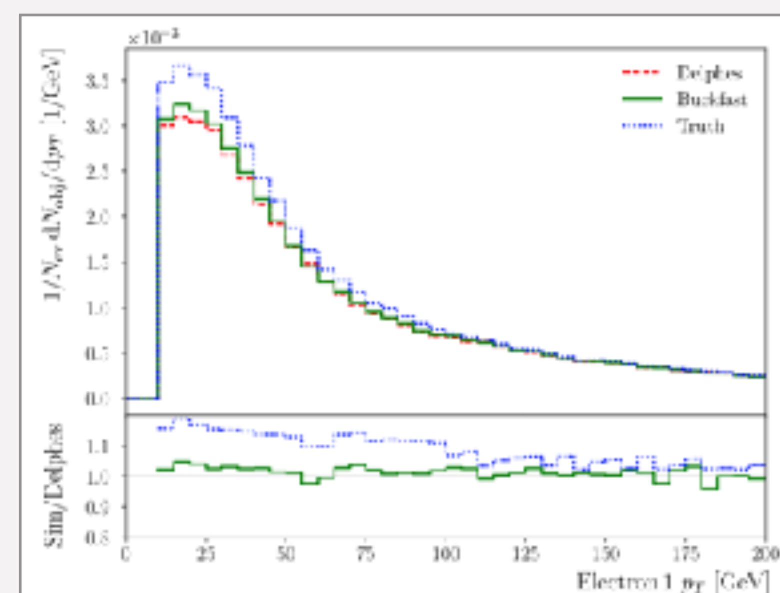
- *Publishing statistical models: Getting the most out of particle physics experiments* **[arxiv:2109.04981]**
- *Reinterpretation of LHC Results for New Physics: Status and Recommendations after Run 2* **[arxiv:2003.07868]**
- *Simple and statistically sound strategies for analysing physical theories* **[arxiv:2012.09874]**

Example: ColliderBit

Full Poisson likelihood from fast MC simulation of searches

- **Focus on speed**
- **MC generation:** Pythia8 parallelised with OpenMP + other speed tweaks
- **Detector simulation:** Fast simulation based on 4-vector smearing
- **Cross-sections:** LO+LL from Pythia8
  Coming soon: fast NLO cross-sections for SUSY
- **Analysis system:** Event-level, independent of simulation
- **Extensive list of ATLAS/CMS searches**
  - ~40 searches, most at 13 TeV
  - mainly SUSY + some monojet DM searches
- **Likelihoods:**
  - marginalise/profile correlated bkg uncertainties; or
  - use «best expected» SR
- **ColliderBit Solo (coming soon):**
  - standalone tool
  - only the analyses + likelihood evaluation (fast)
  - takes HepMC events as input





p_T distribution of leading electrons.

[arXiv:1705.07919]

Dependency resolution

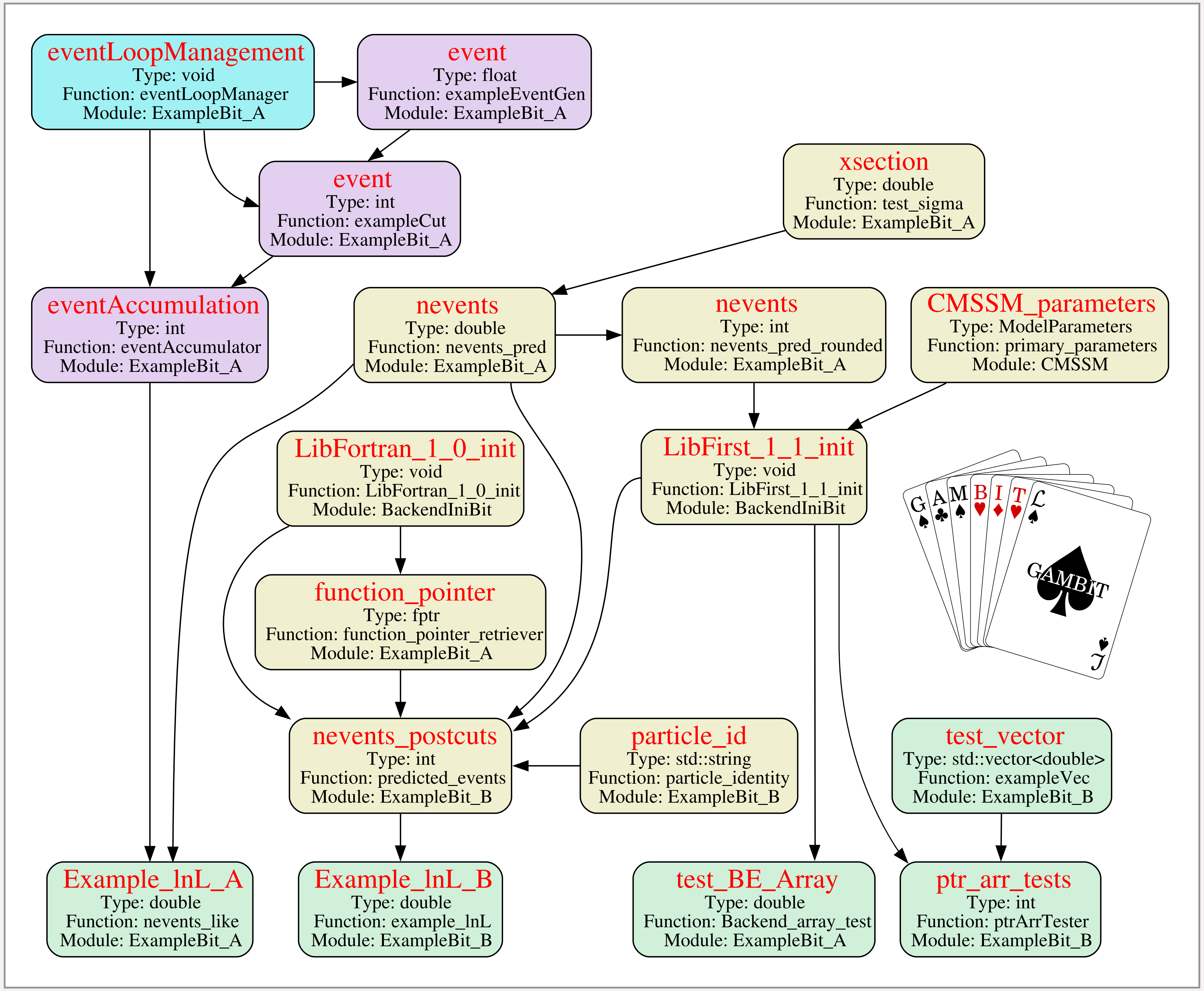
- Basic building blocks: **module functions**
- A physics module: **a collection of module functions** related to the same physics topic
- Each module function has a single **capability** (what it calculates)
- A module function can have **dependencies** on the results of other module functions
- A module function can declare which **models** it can work with
- GAMBIT determines which module functions should be run in which order for a given scan (**dependency resolution**)

```
void function_name(double &result)
{
  ...
  result = ... // something useful
}
```

```
// Observable: BR(B -> tau nu)
#define CAPABILITY Btaunu
START_CAPABILITY
  #define FUNCTION SI_Btaunu
  START_FUNCTION(double)
  DEPENDENCY(SuperIso_modelinfo, parameters)
  BACKEND_REQ(Btaunu, (libsuperiso), double, (const parameters*))
  BACKEND_OPTION( (SuperIso, 3.6), (libsuperiso) )
  #undef FUNCTION
#undef CAPABILITY
```

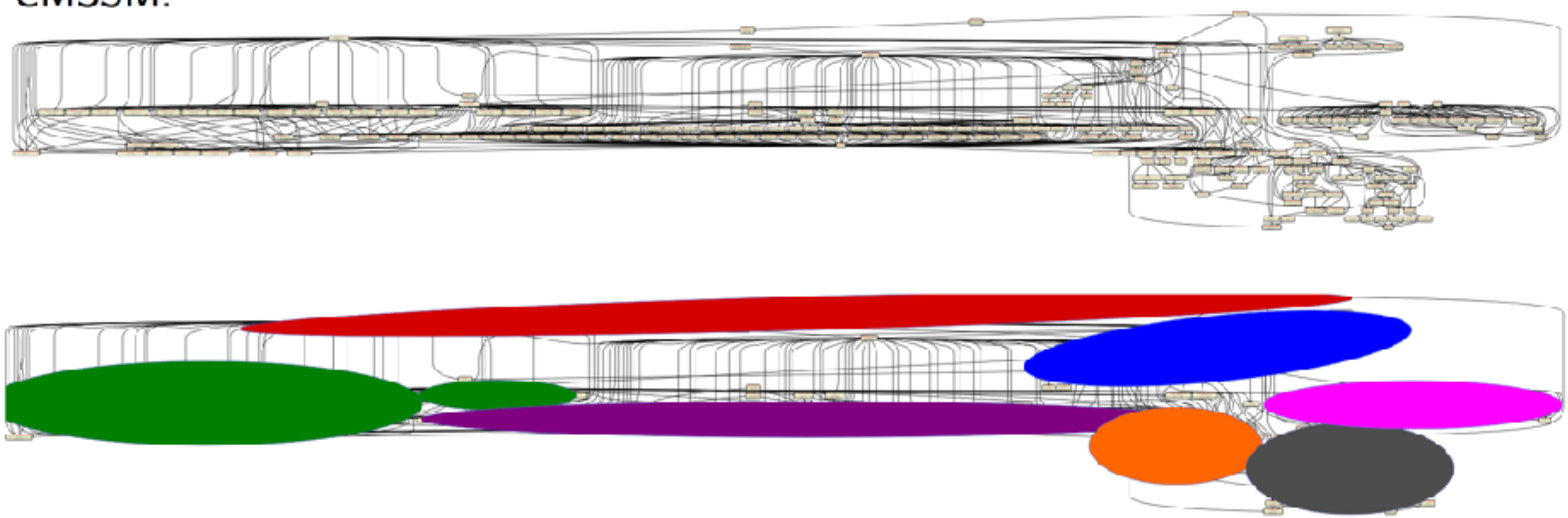
```
/// Br B->tau nu_tau decays
void SI_Btaunu(double &result)
{
  using namespace Pipes::SI_Btaunu;

  parameters const& param = *Dep::SuperIso_modelinfo;
  result = BEreq::Btaunu(&param);
}
```

Dependency resolution



Dependency resolution

CMSSM:



Red: Model parameter translations
Blue: Precision calculations
Green: LEP rates+likelihoods
Purple: Decays
Orange: LHC observables and likelihoods
Grey: DM direct, indirect and relic density
Pink: Flavour physics