# GAMBIT

Marcin Chrząszcz
mchrzasz@cern.ch

Universität Zürich,
Institute of Nuclear Physics, Polish Academy of Science

KEK meeting, KEK
October 29, 2015

## The GAMBIT Collaboration

26 Members, 15 institutions, 9 countries
8 Experiments, 4 major theory codes

| | |
|---|---|
| **Fermi-LAT** | J. Conrad, J. Edsjö, G. Martinez ¶. Scott |
| **ATLAS** | A. Buckley, P. Jackson, C. Rogan, A. Saavedra, M. White |
| **CTA** | C. Balázs, T. Bringmann, J. Conrad, M. White |
| **HESS** | J. Conrad |
| **LHCb** | M. Chrz⊠szcz, N. Serra |
| **IceCube** | J. Edsjö, C. Savage, P. Scott |
| **AMS-02** | A. Putze |
| **CDMS, DM-ICE** | L. Hsu |
| **XENON/DARWIN** | J. Conrad |
| **Theory** | P. Athron, C. Balázs, T. Bringmann, J. Cornell, L. Dal, J. Edsjö, B. Farmer, A. Krislock, A. Kvellestad, M. Pato, F. Mahmoudi, A. Raklev, C. Savage, P. Scott, C. Weniger, M. White |

## Modules

Physics Modules

- ColliderBit ATLAS and CMS likelihoods
- DarkBit Dark Matter searches
- FlavBit – flavour physics inc. $g-2$, $b \to s\gamma$, $B$ decays (new channels, theory uncerts, LHCb likelihoods)
- SpecBit – generic BSM spectrum object, providing RGE running, masses, mixings, etc via interchangeable interfaces to different RGE codes
- DecayBit – decay widths for all relevant SM & BSM particles
- EWPOBit – precision tests (mostly by interface to FeynHiggs, alt. SUSY-POPE)

+ScannerBit: manages statistics, parameter sampling and optimisation algorithms

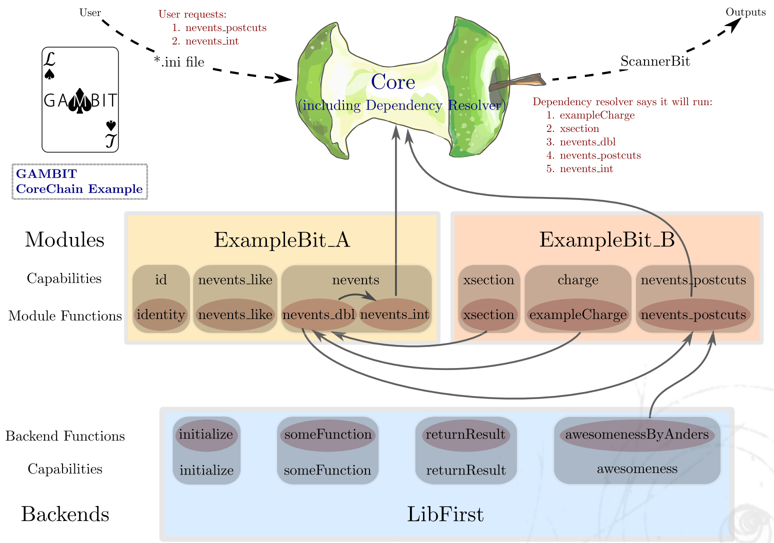
## Backends: mix and match

- GAMBIT modules consist of a number of standalone **module functions**
- Module functions can depend on each other, or they can require specific functions from **backends**
- Backends are external code libraries (DarkSUSY, FeynHiggs, etc) that include different functions
- GAMBIT automates and abstracts the interfaces to backends $\longrightarrow$ backend functions are tagged according to what they calculate
- $\longrightarrow$ with appropriate module design, different backends and their functions can be used interchangeably
- GAMBIT dynamically adapts to use whichever backends are actually present on a user's system (+ provides details of wtf it did of course)

## Backends: mix and match

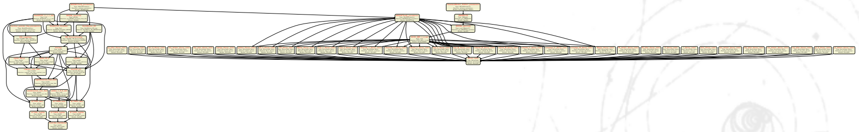
- GAMBIT modules consist of a number of standalone **module functions**
- Module functions can depend on each other, or they can require specific functions from **backends**
- Backends are external code libraries (DarkSUSY, FeynHiggs, etc) that include different functions
- GAMBIT automates and abstracts the interfaces to backends $\longrightarrow$ backend functions are tagged according to what they calculate
- $\longrightarrow$ with appropriate module design, different backends and their functions can be used interchangeably
- GAMBIT dynamically adapts to use whichever backends are actually present on a user's system (+ provides details of wtf it did of course)

# GAMBIT: a toy example



## Dependency Resolution

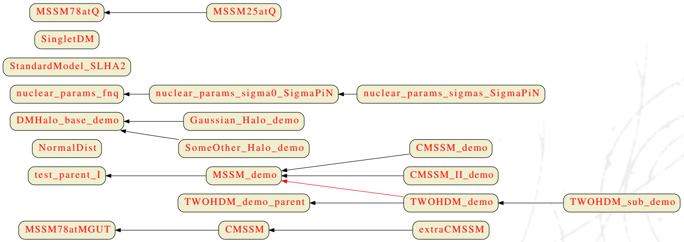
- Module functions and backend functions get arranged into a **dependency tree**
- Starting with requested observables and likelihoods, fills each dependency and backend requirement
- Obeys rules at each step: allowed models, allowed backends, constraints from input file, etc
- $\rightarrow$ tree constitutes a directed acyclic graph
- $\rightarrow$ GAMBIT uses graph-theoretic methods to 'solve' the graph to determine function evaluation order



## Dependency Resolution

## Hierarchical Model Database

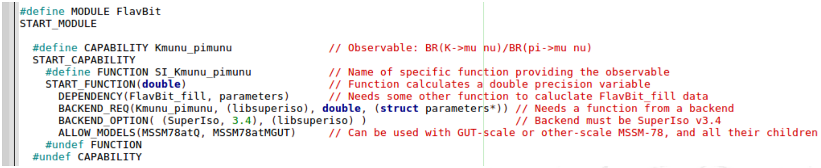
- Models are defined by their parameters and relations to each other
- Models can inherit from **parent models**
- Points in child models can be **automatically translated** to ancestor models
- **Friend models** also allowed (cross-family translation)
- Model dependence of every module/backend function is tracked
  $\Longrightarrow$ maximum safety, maximum reuse



## Expansion: adding new functions

Adding a new module function is easy:

1. Declare the function to GAMBIT in a module's **rollcall header**
    - Choose a capability
    - Declare any **dependencies**
    - Declare any **backend requirements**
    - Declare any specific **allowed models**
    - other more advanced declarations also available



```
#define MODULE FlavBit
START_MODULE

  #define CAPABILITY Kmunu_pimunu                 // Observable: BR(K->mu nu)/BR(pi->mu nu)
  START_CAPABILITY
    #define FUNCTION SI_Kmunu_pimunu              // Name of specific function providing the observable
    START_FUNCTION(double)                        // Function calculates a double precision variable
      DEPENDENCY(FlavBit_fill, parameters)        // Needs some other function to caluclate FlavBit_fill data
      BACKEND_REQ(Kmunu_pimunu, (libsuperiso), double, (struct parameters*)) // Needs a function from a backend
      BACKEND_OPTION( (SuperIso, 3.4), (libsuperiso) )                       // Backend must be SuperIso v3.4
      ALLOW_MODELS(MSSM78atQ, MSSM78atMGUT)       // Can be used with GUT-scale or other-scale MSSM-78, and all their children
    #undef FUNCTION
  #undef CAPABILITY
```

2. Write the function as a simple C++ function
   (one argument: the result)

## Other nice technical features

- **Scanners**: MultiNest, Diver (diff. evolution), PIKAIA (genetic algorithms), GreAT (MCMC)
- **Statistics**: Bayesian, Profile Likelihood, later full Neyman
- Mixed-mode **MPI + openMP**, mostly automated
- diskless generalisation of various Les Houches Accords
- **BOSS**: dynamic loading of C++ classes from backends (!)
- **all-in or module standalone** modes – easily implemented from single cmake script
- **automatic getters** for obtaining, configuring + compiling backends[1]
- **flexible output streams** (ASCII, databases, binary, ...)
- more more more...

[1] if a backend breaks, won't compile and/or kills your dog, blame the authors (not us...unless we **are** the authors...)

## Closing remarks

- Robust analysis of dark matter and BSM physics requires multi-messenger global fits
- GAMBIT is coming:
  - $\longrightarrow$ Global fits to many models for the first time
  - $\longrightarrow$ Better global fits to familiar ones
  - $\longrightarrow$ Highly modular, usable and extendable public code
  - $\longrightarrow$ Faster, more complete and more consistent theory explorations + experimental analysis prototyping

## Backup