GAMBIT
Towards a Global And Modular Beyond-the-Standard-Model Inference Tool
15w5115

Pat Scott (Imperial College London)
Martin White (University of Adelaide)

September 27 – October 2 2015

1 Overview of the Field

The two most pressing and fundamental issues in modern physics are the identity of dark matter (DM) and the nature of physics at the TeV energy scale. A wealth of theories has been proposed for each, and an enormous amount of experimental data has been collected to test those theories. Unfortunately though, when it comes to comparing all these theories with all the relevant data, existing computational and statistical tools in particle and astroparticle physics are woefully inadequate. Our Workshop was aimed at helping remedy this situation.

DM constitutes 80% of the matter in the Universe and was discovered 80 years ago, but its composition remains a mystery. With the impending activation of the 14 TeV Large Hadron Collider (LHC), discovery of the Higgs boson and construction of high-energy astrophysics experiments like the *Fermi*-LAT, HESS-II, IceCube and SuperCDMS, we now stand on the doorstep of the TeV scale. Many popular DM candidates are intrinsically linked to the appearance of new physics beyond the Standard Model (BSM) at precisely this scale [1]. The mass of the newly-discovered Higgs itself even compels us to move beyond the Standard Model [2, 3]. We also know that the Standard Model (SM) is incomplete because it does not include gravity, nor explain the excess of matter over antimatter in our Universe, nor the fact that neutrinos have mass.

Many different experimental probes of BSM physics exist: direct and indirect searches for DM, accelerator searches, and neutrino experiments. Experiments such as CRESST [4], *Fermi*-LAT [5] and PAMELA [6] may even already show tantalizing hints of DM. To make robust conclusions about the overall level of support for different BSM scenarios from such varied sources, a simultaneous statistical fit of all the data, fully taking into account all relevant uncertainties, assumptions and correlations is an absolute necessity. This global fit approach is the one we discussed and developed at CMO-BIRS in 2015. Such holistic analyses exploit the synergy between different experimental approaches to its maximum potential, squeezing every last statistical drop of information possible from each experiment. Robust analysis of correlated signals, in a range of complementary experiments, is *essential* for claiming a credible discovery of DM or new physics at the TeV scale – and indeed, even for definitively excluding theories. This 'win-win' situation is a particular feature of a global fit analysis, as even non-detections provide crucial physical insight into which theories and parameter regions are disfavoured.

```
#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
```

Figure 1: Example of the declaration of a module function in GAMBIT, showing the assignment of a capability to a module function, as well as an example dependency and an example backend requirement. Also shown are some example conditions placed on usage of the module function, in terms of the identity of the backend used to fulfil the backend requirement, and according to the BSM model being examined.

2 Recent Developments and Open Problems

Existing global fits [7, 8, 9, 10, 11, 12, 13, 14] cover only a very small subset of interesting particle models; most have dealt with only the very simplest versions of supersymmetry. This is partly for computational reasons, as efficiently exploring the parameter spaces of more complicated models is extremely time consuming. Existing optimization and inference techniques are barely capable of dealing with even the models that have been considered so far [15, 16]. Efforts to date have rarely been truly 'global', as the full range of possible observables and datasets (e.g. indirect searches for dark matter) have not been included in a detailed way. The present generation of global analysis suites have all now essentially hit a brick wall in their abilities to deal with alternative theories, additional observables and the advanced numerical and statistical algorithms required for producing genuinely robust results.

Future progress in understanding which BSM models are favoured by experimental data will be contingent upon massively expanding the range of theories to which global fits have been applied, and the number of experimental results included in them. The only way to do this is to reconsider the computational and statistical tools used to carry them out, from the ground up. Our BIRS Workshop did exactly this.

3 Highlights and Progress

We discussed and carried out a large part of the development of a new, second-generation global fitting suite for particle and astroparticle physics: GAMBIT. GAMBIT will transform the budding field of global fits, by providing a framework in which new theories, observables, likelihoods and scanning algorithms can be quickly, easily and consistently combined in order to completely and rigorously test essentially *any* proposed extension of particle physics beyond the SM.

Below we list a few particular highlights in this development.

3.1 Physics and Scanner Modules

GAMBIT consists of a series of core components, a scanning module (ScannerBit) and 6 specific physics modules. These are:

- DarkBit – consisting of dark matter observables and associated likelihood functions. This includes relic density calculations, direct detection and indirect detection with gamma rays, neutrinos and the cosmic microwave background radiation. Likelihood functions for the Fermi, CTA, HESS, IceCube, LUX, SuperCDMS, SIMPLE, XENON100 and Plack experiments are all included.

- ColliderBit – consisting of LHC and LEP observables and associated likelihoods. This includes sparticle production at the LHC and LEP, Monte Carlo simulation of such signals, and LHC Higgs bounds.

- DecayBit – consisting of routines to calculate decay rates of all particles from the SM and beyond, as well as LHC likelihoods for Higgs decays.

- SpecBit – consisting of routines to perform renormalisation group evolution and calculate particle masses.
- FlavBit – consisting of observables and likelihood functions from flavour physics. This includes extensive rare process predictions and corresponding LHCb likelihoods.
- PrecisionBit – consisting of precision observables and likelihoods. These include uncertainties on SM quantities like the top and bottom quark masses and the SM gauge couplings, as well as things like electroweak precision observables and the anomalous magnetic moment of the muon.

The physics modules each consist of a series of standalone functions. These functions are each assigned a **capability** describing what they calculate, and may also optionally be given **dependencies** and/or **backend requirements** (Fig. 3.1). GAMBIT then connects different functions to each other according to these capabilities, dependencies and backend requirements.

3.2 Backend system

Many excellent codes exist for performing small numbers of specialised calculations in specific BSM theories. Rather than repeat the many years of work that went into these codes, we created a system whereby any existing code can be used to perform calculations from within GAMBIT. These 'backend' codes must be compiled as shared libraries, and some basic code written in GAMBIT in order to inform it of which functions might be required from those shared libraries. The actual libraries are then dynamically loaded at runtime, making it possible to use GAMBIT with or without the presence of any particular backend on a user's machine. If the required backend is present, any backend functions required by a module function are then provided as function pointers to the module function at runtime. When the necessary backend is missing, GAMBIT attempts to compensate by using any other compatible functions available from backends that *are* present, or provides a detailed error message informing the user which backend they need to install in order to complete a given scan.

3.3 BOSS and the new C++ 'ladder' pattern

To be able to dynamically load other C++ libraries which define classes necessary for the code's use by an external program such as GAMBIT, we needed to come up with a system for dynamically loading C++ classes at runtime. This does not exist as a possibility in the language, as the dynamic loader `dlopen` is a pure C utility. To solve this problem, we created the Backend on a Stick Script (BOSS). BOSS utilises a 'ladder' pattern, which is a generalisation of the well-know 'factory' pattern for external classloading. BOSS parses the code of the external library, creating additional interface and factory classes. It then causes every class in the hierarchy in which a class of interest is involved to descend from its interface class. The headers declaring the interface and factory classes are compiled into both GAMBIT and the external code, allowing the class to be instantiated from within GAMBIT, but actually constructed from the external code. GAMBIT also tracks which BOSSed libraries have actually been loaded or not at runtime, and only allows module functions using classes for which a constructor is available to be activated in a scan.

Some example diagnostic output of the backend system is shown in Fig. 3.2. This indicates which backends GAMBIT has been configured to work with, their library locations, status and the number of available functions, classes and constructors that each library would provide.

3.4 Scan initialisation by graph methods

One of the problems with existing BSM global fit codes is their inability to adapt efficiently to new models, by running only the calculations required for the specific model under investigation, and identifying gaps where new code must be written in order to complete a particular calculation in a new model. To deal with this, we designed GAMBIT such that module functions are placed automatically into a dependency tree at runtime, according to their capabilities and dependencies on other module functions' results. GAMBIT takes input from the user about what observables or likelihoods need to be calculated for each parameter combination, and then uses the reported capabilities and dependencies of the various module functions to work out which

pat@xpspedition: ~/gambit 163x45

All relative paths are given with reference to /home/pat/gambit.

BACKENDS	VERSION	PATH TO LIB	STATUS	#FUNC	#TYPES	#CTORS
DDCalc0	0.0	Backends/installed/DDCalc/0.0/libDDCalc0.so	OK	62	0	0
DarkSUSY	5.1.1	Backends/installed/DarkSUSY/5.1.1/lib/libdarksusy.so	OK	68	0	0
FastSim	1.0	Backends/installed/fastsim/1.0/libfastsim.so	absent/broken	1	0	0
FeynHiggs	2.11	Backends/installed/FeynHiggs/2.11.2/lib/libFH.so	OK	14	0	0
HiggsBounds	4.2.1	Backends/installed/HiggsBounds/4.2.1/lib/libhiggsbounds.so	OK	10	0	0
HiggsSignals	1.4	Backends/installed/HiggsSignals/1.4.0/lib/libhiggssignals.so	OK	11	0	0
LibFarrayTest	1.0	Backends/examples/libFarrayTest.so	OK	9	0	0
LibFirst	1.0	Backends/examples/libfirst.so	OK	8	0	0
	1.1	Backends/examples/libfirst.so	OK	15	0	0
LibFortran	1.0	Backends/examples/libfortran.so	OK	6	0	0
MicrOmegas	3.5.5	Backends/installed/micromegas/3.5.5/MSSM/libmicromegas.so	OK	15	0	0
MicrOmegasSingletDM	3.5.5	Backends/installed/micromegas/3.5.5/SingletDM/libmicromegas.so	OK	13	0	0
Pythia	8.186	Backends/installed/Pythia/8.186/lib/libpythia8.so	absent/broken	0	27	105
	8.209	Backends/installed/Pythia/8.209/lib/libpythia8.so	OK	0	28	107
SUSYPOPE	0.2	no path in config/backend_locations.yaml	absent/broken	3	0	0
SUSY_HIT	1.5	Backends/installed/SUSY-HIT/1.5/libsusyhit.so	OK	55	0	0
Superiso	3.4	Backends/installed/Superiso/3.4/libsuperiso.so	OK	32	0	0
gamLike	1.0.0	Backends/installed/gamLike/1.0.0/lib/gamLike.so	OK	3	0	0
nulike	1.0.0	Backends/installed/nulike/1.0.0/lib/libnulike.so	OK	4	0	0

Gambit diagnostic backend line 1 (press h for help or q to quit)

Figure 2: Example diagnostic output of the GAMBIT backend system, showing which backends are configured within GAMBIT, which ones are actually detected and working, where they reside, and how many functions, constructors and classes they each provide.

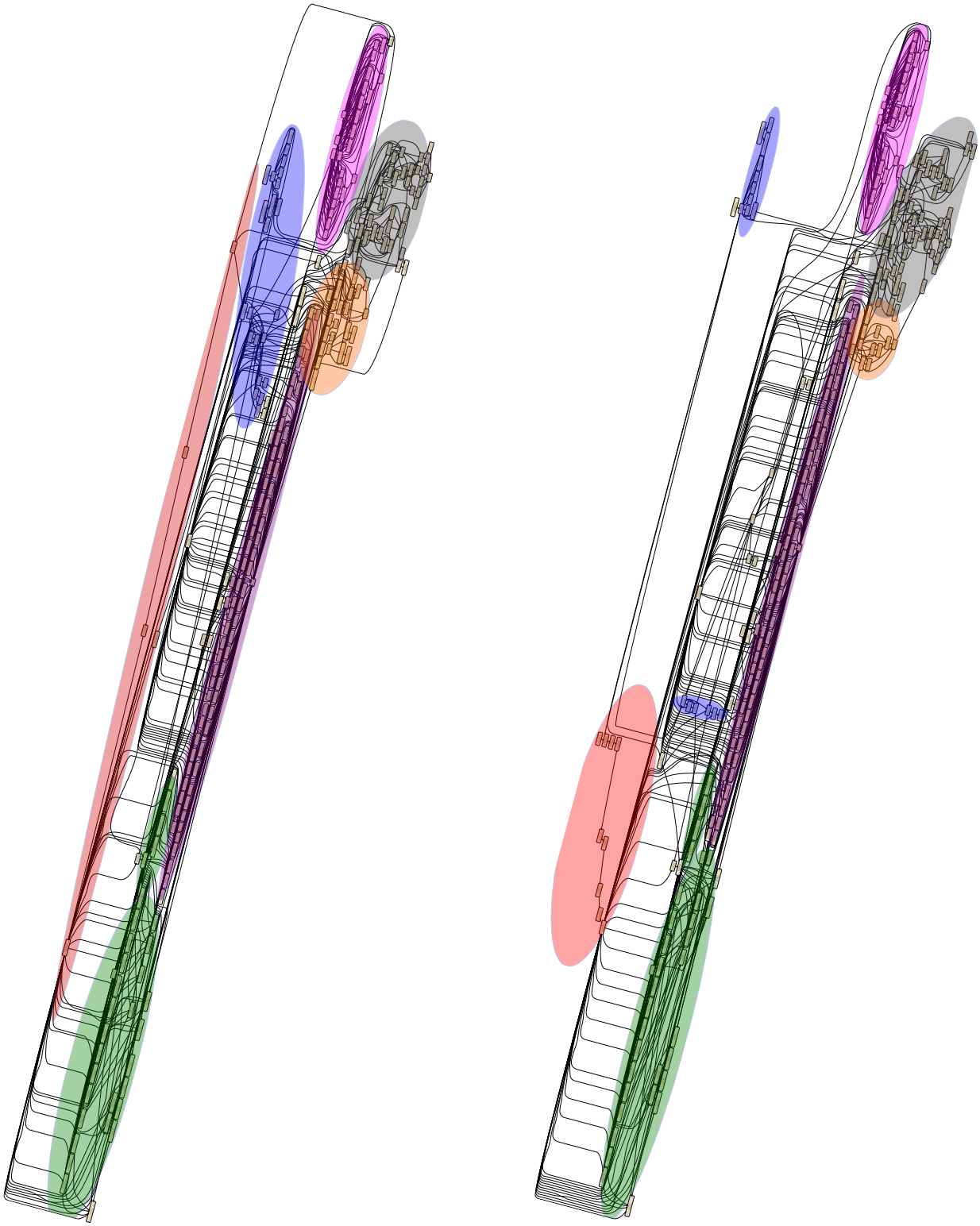

Figure 3: Example dependency trees for GAMBIT scans of the CMSSM (left) and MSSM-7 (right). Quantities calculated earlier typically appear near the top of the plots (left in portrait orientation), whereas later calculations generally appear further down the page (right in portrait orientation). Red shading indicates the approximate region where module functions are doing model parameter translation (core and SpecBit functions), blue corresponds to PrecisionBit, green to LEP likelihoods (ColliderBit), purple to DecayBit, grey to DarkBit, orange to LHC likelihoods (ColliderBit), and pink to FlavBit.

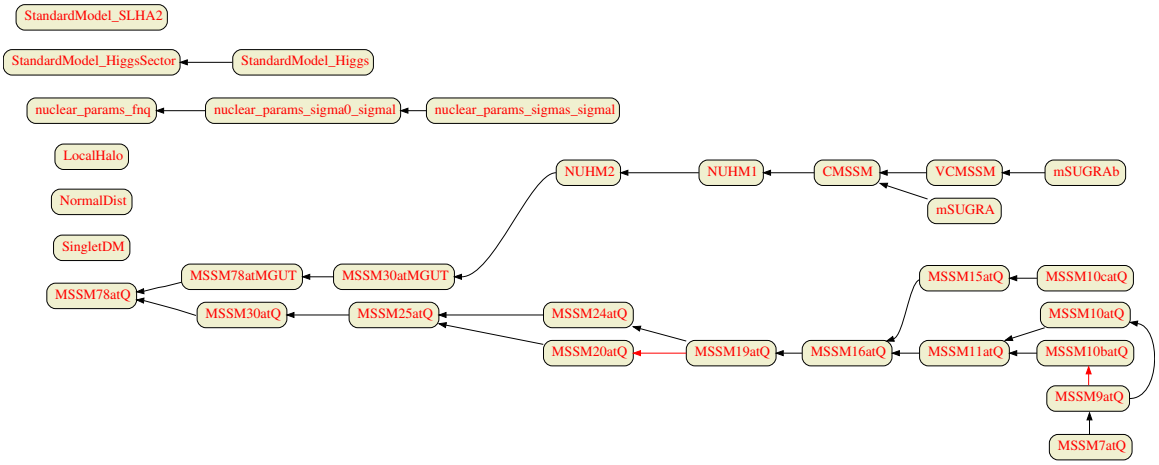

Figure 4: The GAMBIT model hierarchy as it currently stands. Here one can see 'nuisance models' (SM, nuclear parameters), the scalar singlet DM model, and two branches of supersymmetry: the high-scale branch (CMSSM, NUHM, etc) and the weak-scale branch (MSSM-7, -11, -15, etc). Parameter translations from child to parent models are shown with black arrows, and translations to friend models with red arrows.

functions it needs to place into the dependency tree in order to complete this calculation. Assuming that no closed loops are formed (which GAMBIT treats as an error), the dependency tree therefore constitutes a directed acyclic graph. GAMBIT uses methods from graph theory to solve this graph and determine the correct order in which to evaluate the different module functions such that every module function runs only *after* all functions that fulfil its dependencies have run.

In this manner, large parts of likelihood calculation become reusable for multiple models, as the further 'downstream' in the dependency tree a calculation goes, the more model-independent it typically becomes. Examples of this are shown in Fig. 3.4 for the two example supersymmetric scans that we started working with in Oaxaca: the high-scale constrained minimal supersymmetric model (CMSSM) and the 7-parameter weak-scale MSSM (MSSM-7). In these examples, the earlier parts of the calculations show obvious differences at the graph level, e.g. in module functions implementing model parameter translations and precision observables, but more derived quantities – like dark matter observables – are essentially identical for the two models.

3.5 Model hierarchy

In order to track the model-dependence of each step in every observable and likelihood calculation, GAMBIT demands that each module function either have a series of models explicitly declared as allowed for use with that module function (e.g. Fig. 3.1), or that a module function can be used with *any* model. BSM models are declared to GAMBIT simply as a group of parameters. Models can be declared to descend from other models, which implies that a parameter point in the child model must be able to be interpreted – via an appropriate translation function – as a point in its parent model (and therefore by extension, in any of its parent's ancestors). The model definition therefore requires that any child model also comes with a declaration of that translation function. It is also possible to declare translation functions from one model to a 'friend' model, allowing translational links to be established *across* model families. The resulting model hierarchy presently declared within GAMBIT can be seen in Fig. 3.5.

Together, this arrangement helps ensure that module functions written for one scan can be easily and safely re-used automatically in scans of new models, if and only if they are appropriate for doing so. GAMBIT makes use of this by automatically translating a model point in a given scan to the corresponding point in the model(s) that each module function declares that it can work with. GAMBIT is not limited to scanning just one model at a time, so joint models (e.g. a DM particle model and a model for the Galaxy's DM halo) need not be declared as joint models, but merely scanned over simultaneously – any module function that needs

```yaml
Parameters:
  StandardModel_SLHA2: !import StandardModel_SLHA2_defaults.yaml
  MSSM25atQ: !import LesHouches.in.MSSM_1.yaml

Priors:
    # none: all parameters fixed in this example.

Scanner:

  use_scanner: toy_mcmc

  scanners:
    toy_mcmc:
      plugin: toy_mcmc
      point_number: 2000
      output_file:  output
      like:  Likelihood

ObsLikes:

  # Test DecayBit
  - purpose:      Test
    capability:   decay_rates
    type:         DecayTable

  # 79-string IceCube likelihood
  - capability: IceCube_likelihood
    purpose: Likelihood
    function: IC79_loglike

Rules:

  - capability: MSSM_spectrum
    function: get_MSSMatQ_spectrum
    options:
      invalid_point_fatal: true
```

Figure 5: An extract from an example GAMBIT yaml input file, used for defining a scan. Simple text-based input sections exist for defining the model (Parameters), priors on its parameters (Priors), the optimiser or integrator to be employed for exploring the paramter space (Scanner), the observables and likelihoods to include in the scan (ObsLikes) and specific conditions to be imposed on how they are calculated (Rules).

parameters from both such models need only say so in its declaration (Fig. 3.1).

3.6 Improved scanning algorithms for BSM global fits

Existing codes rely almost exclusively on nested sampling, a technique designed to calculated the Bayesian evidence. So far, only a small amount of attention has been paid to methods designed for optimisation rather than integration [15]. This is problematic when one is interested in calculating a profile likelihood, as that requires an accurate characterisation of the global maximum likeihood point, as well as dense sampling around it. We investigated and implmented a number of improved methods in Oaxaca. These included the evolutionary algorithms known as differential evolution and genetic algorithms, as well as improved Bayesian methods that make use of multiple interacting MCMC-type chains (polycord and the so-called t-walk). It is apparent that the most efficient algorithm for any given scan will depend strongly on the model being scanned, the observables included in the likelihood function, and whether one is interested in drawing conclusions with Bayesian or frequentist inference techniques. For this reason, we plan to offer all these methods as options in the public release of the GAMBIT code.

3.7 Simple input format

Practically driving a GAMBIT scan is done by writing a simple YAML-format text file, and using it as input to the GAMBIT executable. An excerpt from an example GAMBIT yaml file is shown in Fig. 3.7. The file consists of sections for defining the model(s) to be scanned over and the allowed ranges for the model parameters (‘Parameters’), a section for specifying what priors are to be applied to them (‘Priors’), a section for choosing which scanning algorithm to employ and what settings it should be run with (‘Scanner’), and sections for choosing which functions actually end up in the dependency tree (’ObsLikes’ and ’Rules’). The ObsLikes section is for specifying likelihood components that should be used for driving the scanner’s exploration of the parameter space, as well as additional observables that the user is interested in having calculated for each parameter combination. The Rules section provides a way to define rules about which module and backend functions should be used for resolving different dependencies and backend requirements.

This gives the user complete control over exactly what is connected to what in the dependency tree, allowing him/her to override automatic decisions made by GAMBIT or break deadlocks that GAMBIT has no way to solve (if e.g. many functions exist that are all equally valid ways of fulfilling a dependency). The Rules section also allows the user to provide options to be passed to module functions. There is also an additional section not shown, where the user can set global options relating to how GAMBIT itself (rather than individual module functions) operates.

4 Outcomes of the Meeting

We made quite substantial progress on the common project of writing the GAMBIT code, defining the physics analyses that we will do with it, and carrying them out.

1. We re-evaluated the computational and statistical techniques used to carry out global fits in particle and astroparticle physics.
2. We completed the majority of the development of a second generation global-fitting package for particle and astroparticle physics: GAMBIT, the Global And Modular Beyond-the-Standard-Model Inference Tool.
3. We determined the statistical approaches to be supported by GAMBIT: parameter estimation by both Bayesian posterior and profile likelihood, model comparison by Bayesian model comparison and direct p-value computation, and systematic error treatment by inline marginalisation and profiling.
4. We identified new computational strategies that optimise the efficiency and accuracy of GAMBIT relative to first-generation codes: differential evolution, the t-walk algorithm, graph-theoretic techniques for scan initialisation, model hierarchies and the new C++ 'ladder pattern' for dynamic classloading.
5. We determined a public release plan for the GAMBIT software.
6. We prioritised and discussed different physics analyses to be carried out with GAMBIT. The first of these will be centred on supersymmetry and scalar singlet models for dark matter.
7. We began drafting a series of 9 initial papers to be submitted to the European Journal of Physics C in the next few months, describing GAMBIT, its component physics and statistical packages, and its first physics results. Many aspects of this report will be expanded on in those papers.

References

[1] G. Bertone, D. Hooper, and J. Silk, *Particle dark matter: evidence, candidates and constraints*, *Phys. Rep.* **405** (2005) 279–390, [hep-ph/0404175].

[2] H. Baer and X. Tata, *Weak Scale Supersymmetry*. Cambridge University Press, 2006.

[3] G. Degrassi, S. Di Vita, *et. al.*, *Higgs mass and vacuum stability in the Standard Model at NNLO*, *JHEP* **8** (2012) 98, [arXiv:1205.6497].

[4] G. Angloher, M. Bauer, *et. al.*, *Results from 730 kg days of the CRESST-II Dark Matter search*, *Eur. Phys. J. C* **72** (2012) 1971, [arXiv:1109.0702].

[5] T. Bringmann, X. Huang, A. Ibarra, S. Vogl, and C. Weniger, *Fermi LAT search for internal bremsstrahlung signatures from dark matter annihilation*, *JCAP* **7** (2012) 54, [arXiv:1203.1312].

[6] O. Adriani, G. C. Barbarino, *et. al.*, *An anomalous positron abundance in cosmic rays with energies 1.5–100 GeV*, *Nature* **458** (2009) 607–609, [arXiv:0810.4995].

[7] P. Scott, C. Savage, J. Edsjö, and the IceCube Collaboration: R. Abbasi et al., *Use of event-level neutrino telescope data in global fits for theories of new physics*, *JCAP* **11** (2012) 57, [arXiv:1207.0810].

[8] P. Bechtle, T. Bringmann, *et. al.*, *Constrained supersymmetry after two years of LHC data: a global view with Fittino*, *JHEP* **6** (2012) 98, [arXiv:1204.4199].

[9] O. Buchmueller, R. Cavanaugh, *et. al.*, *The CMSSM and NUHM1 in light of 7 TeV LHC, $B_s \to \mu^+ \mu^-$ and XENON100 data*, *Eur. Phys. J. C* **72** (2012) 2243, [arXiv:1207.7315].

[10] L. Roszkowski, E. M. Sessolo, and Y.-L. S. Tsai, *Bayesian implications of current LHC supersymmetry and dark matter detection searches for the constrained MSSM*, *Phys. Rev. D* **86** (2012) 095005, [arXiv:1202.1503].

[11] C. Strege, G. Bertone, *et. al.*, *Global fits of the cMSSM and NUHM including the LHC Higgs discovery and new XENON100 constraints*, *JCAP* **4** (2013) 13, [arXiv:1212.2636].

[12] A. Achterberg, S. Amoroso, S. Caron, L. Hendriks, R. Ruiz de Austri and C. Weniger, *JCAP* **1508** (2015) 08, 006, [arXiv:1502.05703].

[13] K. J. de Vries *et al.*, *Eur. Phys. J. C* **75** (2015) 9, 422, [arXiv:1504.03260].

[14] P. Bechtle *et al.* (2015) arXiv:1508.05951.

[15] Y. Akrami, P. Scott, J. Edsjö, J. Conrad, and L. Bergström, *A profile likelihood analysis of the Constrained MSSM with genetic algorithms*, *JHEP* **4** (2010) 57, [arXiv:0910.3950].

[16] F. Feroz, K. Cranmer, M. Hobson, R. Ruiz de Austri, and R. Trotta, *Challenges of profile likelihood evaluation in multi-dimensional SUSY scans*, *JHEP* **6** (2011) 42, [arXiv:1101.3296].