Update on GAMBIT

Pat Scott

Imperial College London

Slides at: tinyurl.com/patscott
GAMBIT: gambit.hepforge.org

Global fits for dark matter and new physics

Current global fit codes are hardcoded to deal with only a few

- theories (MSSM and/or mSUGRA+friends)
- theory calculators (often interfaced in a very ad hoc way)
- datasets and observables (often missing detailed likelihoods)
- scanning algorithms and statistical methods (generally just one)

$\Longrightarrow$ *hitting the wall on theories, data & computational methods*

Global fits for dark matter and new physics

Current global fit codes are hardcoded to deal with only a few

- theories (MSSM and/or mSUGRA+friends)
- theory calculators (often interfaced in a very ad hoc way)
- datasets and observables (often missing detailed likelihoods)
- scanning algorithms and statistical methods (generally just one)

$\Longrightarrow$ *hitting the wall on theories, data & computational methods*

How to quickly recast data, likelihood functions, scanning code 'housekeeping' and even theory predictions to new theories?

$\Longrightarrow$ a new, very general global fitting framework

Global fits for dark matter and new physics

Current global fit codes are hardcoded to deal with only a few

- theories (MSSM and/or mSUGRA+friends)
- theory calculators (often interfaced in a very ad hoc way)
- datasets and observables (often missing detailed likelihoods)
- scanning algorithms and statistical methods (generally just one)

$\Longrightarrow$ *hitting the wall on theories, data & computational methods*

How to quickly recast data, likelihood functions, scanning code 'housekeeping' and even theory predictions to new theories?

$\Longrightarrow$ a new, very general global fitting framework

$\Longrightarrow$ GAMBIT

GAMBIT: a *second-generation* global fit code

GAMBIT: The Global And Modular BSM Inference Tool

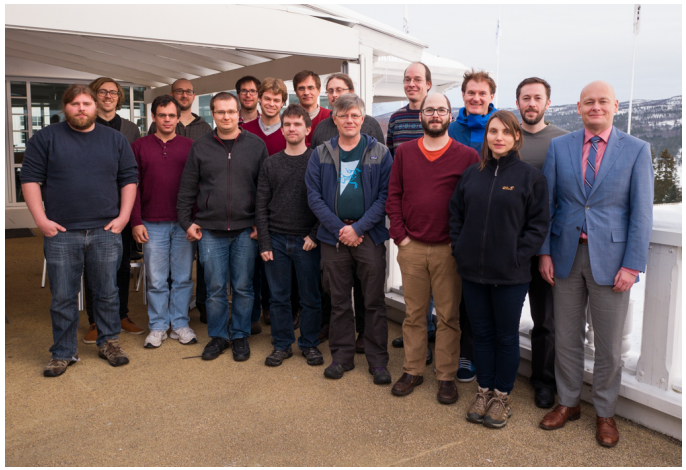
Overriding principles of GAMBIT: flexibility and modularity

- General enough to allow fast definition of new datasets and theoretical models
- Plug and play scanning, physics and likelihood packages
- Extensive model database – not just small modifications to constrained MSSM (NUHM, etc), and not just SUSY!
- Extensive observable/data libraries (likelihood modules)
- Many statistical options – Bayesian/frequentist, likelihood definitions, scanning algorithms
- A smart and *fast* LHC likelihood calculator
- Massively parallel
- Full open-source code release

The GAMBIT Collaboration

30 Members, 16 institutions, 10 countries,
11 Experiments, 4 major theory codes

Fermi-LAT	J. Conrad, J. Edsjö, G. Martinez P. Scott
ATLAS	A. Buckley, P. Jackson, C. Rogan, M. White,
CTA	C. Balázs, T. Bringmann, J. Conrad, M. White
HESS	J. Conrad
LHCb	M. Chrząszcz, N. Serra
IceCube	J. Edsjö, 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, P. Scott, C. Weniger, M. White



Also recently joined or left: T. Gonzales, J. McKay, R. Ruiz, A. Saavedra, C. Savage, R. Trotta

Modules

Physics modules

- DarkBit – dark matter observables (relic density, direct + indirect detection)
- ColliderBit – collider observables inc. Higgs + SUSY searches from ATLAS, CMS + LEP
- FlavBit – flavour physics inc. $g-2$, $b \to s\gamma$, B decays (new channels, angular obs., 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
- PrecisionBit – SM likelihoods, precision BSM tests (W mass, $\Delta\rho$ etc)

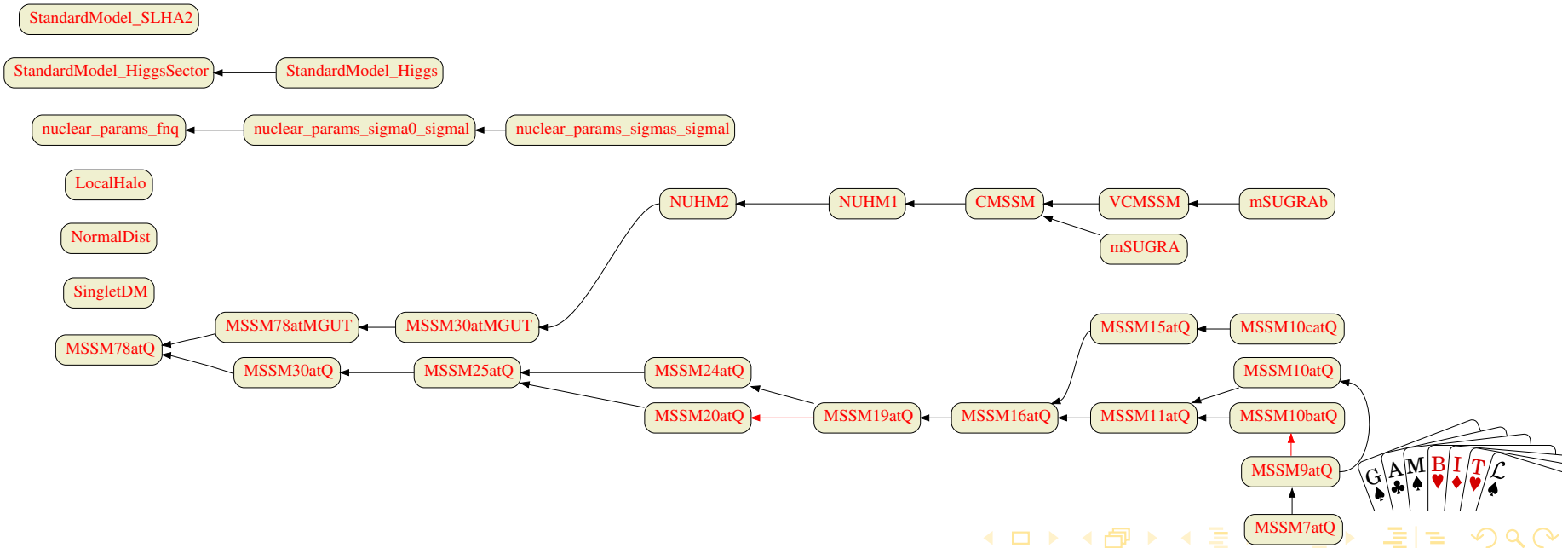
Each consists of a number of **module functions** that can have **dependencies** on each other

+ScannerBit: manages stats, sampling and optimisation

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 function/observable is tracked

$\Longrightarrow$ maximum safety, maximum reuse



Backends: mix and match

- Module functions 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 $\rightarrow$ backend functions are tagged according to what they calculate
- $\rightarrow$ 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 what it decided to do of course)

Backends: mix and match

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/MSSM/libmicromegas.so	OK	15	0	0
MicrOmegasSingletDM	3.5.5	Backends/installed/micromegas/3.5.5/SingletDM/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)

Backends: mix and match

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/MSSM/libmicromegas.so	OK	15	0	0
MicrOmegasSingletDM	3.5.5	Backends/installed/micromegas/3.5.5/SingletDM/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

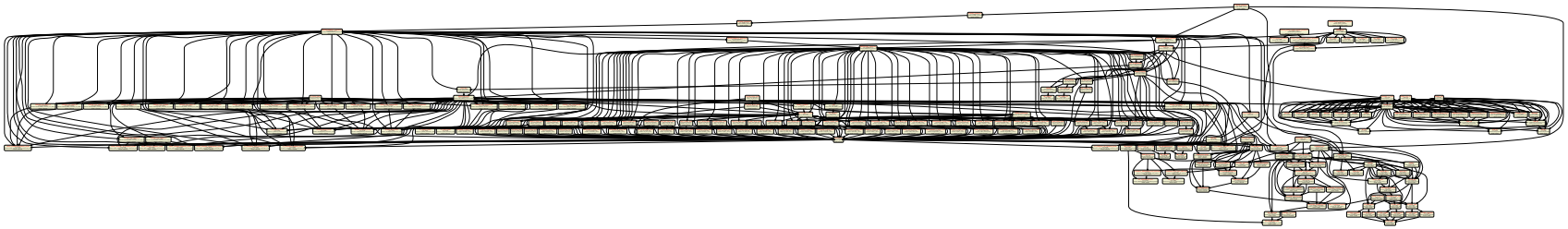
Savage

Weniger

Scott

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

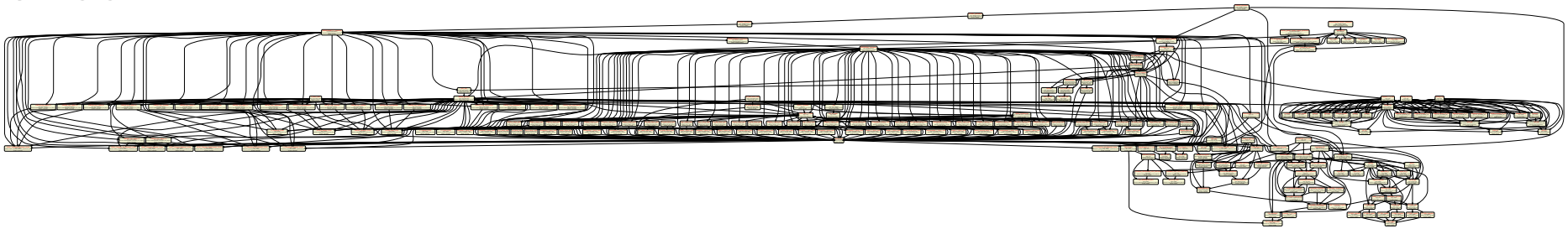
Dependency Resolution



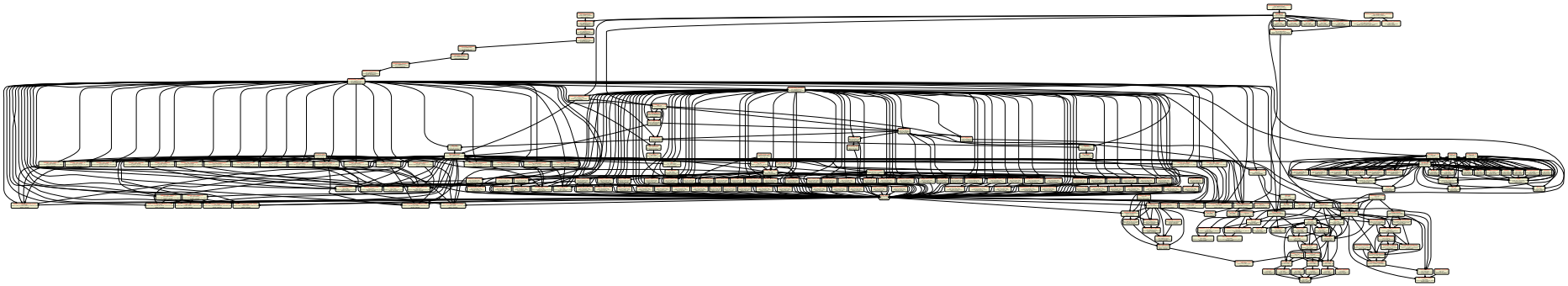
- Module functions and backend functions get arranged into a **dependency tree**
- Starting with requested observables and likelihoods, GAMBIT fills each dependency and backend requirement
- Obeys **rules** at each step: allowed models, allowed backends, constraints from input file, etc
- $\longrightarrow$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

Dependency Resolution

CMSSM:

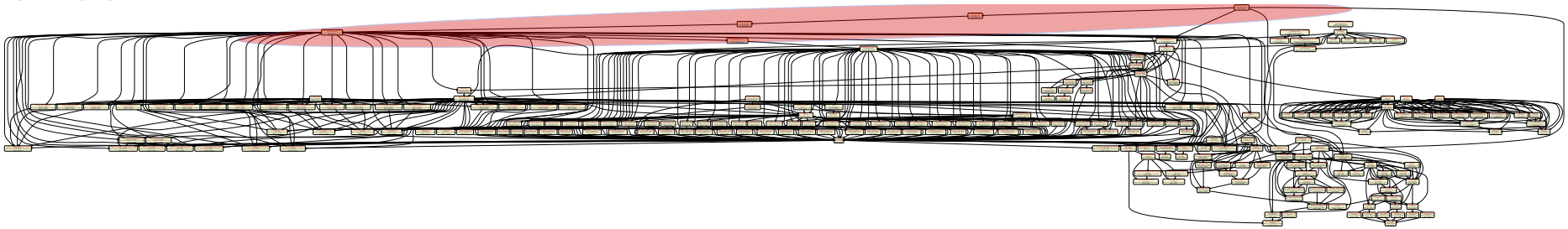


MSSM7:

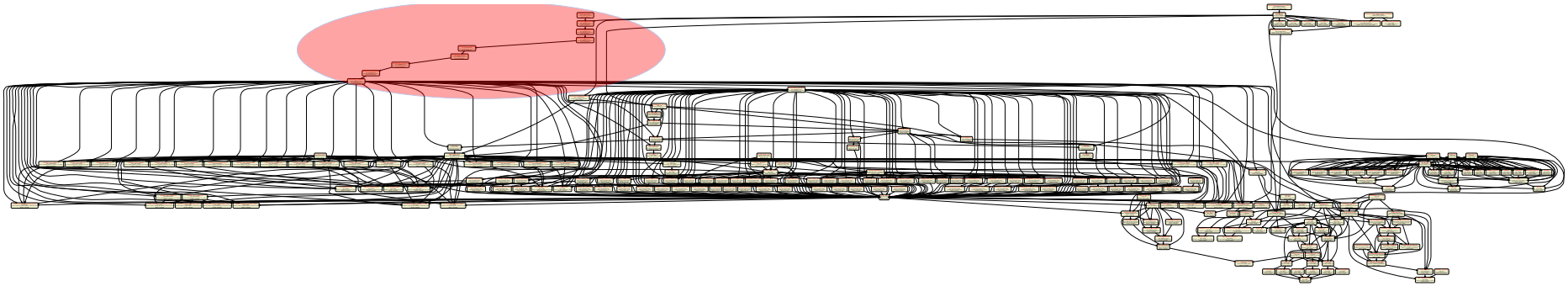


Dependency Resolution

CMSSM:



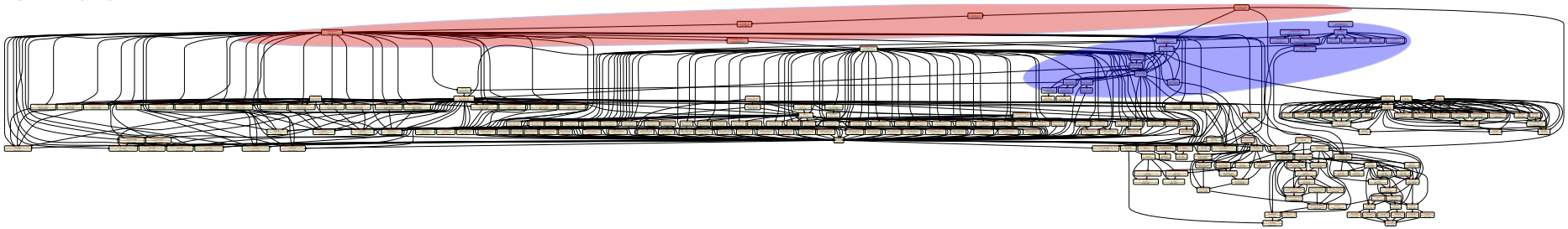
MSSM7:



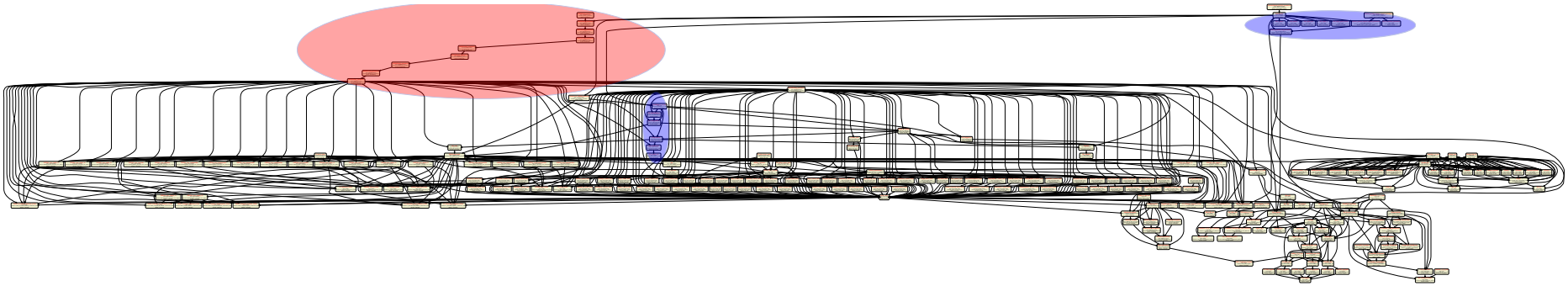
Red: Model parameter translations

Dependency Resolution

CMSSM:



MSSM7:

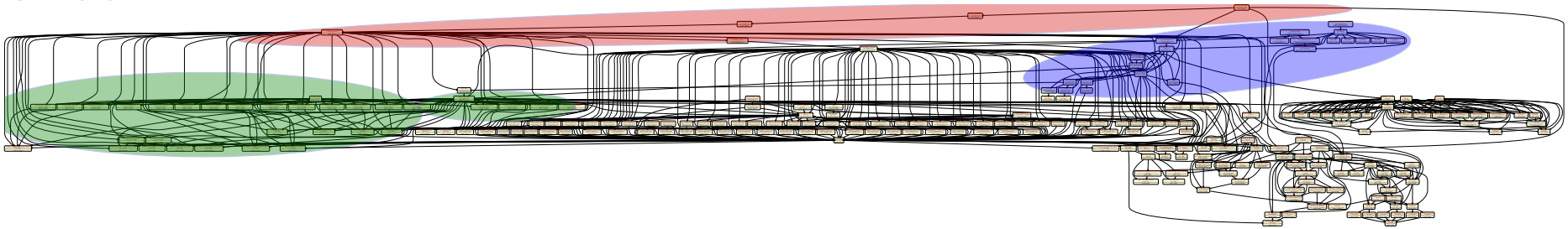


Red: Model parameter translations

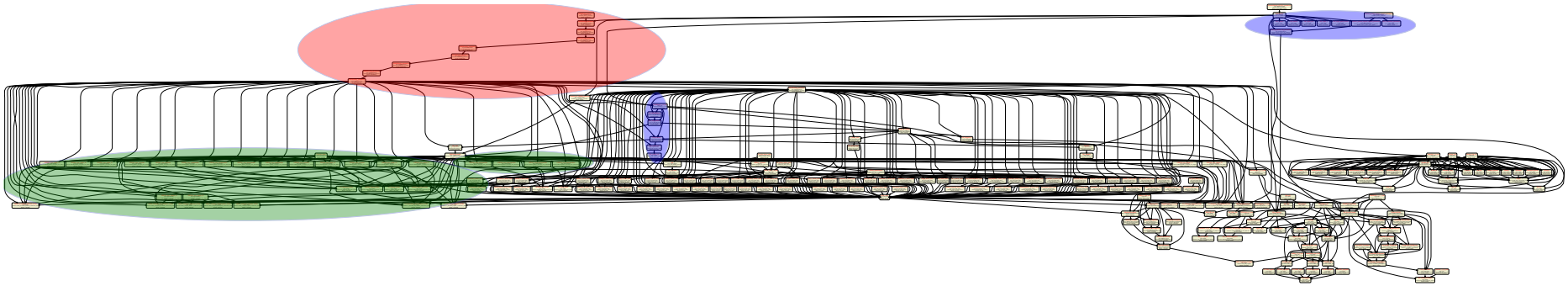
Blue: Precision calculations

Dependency Resolution

CMSSM:



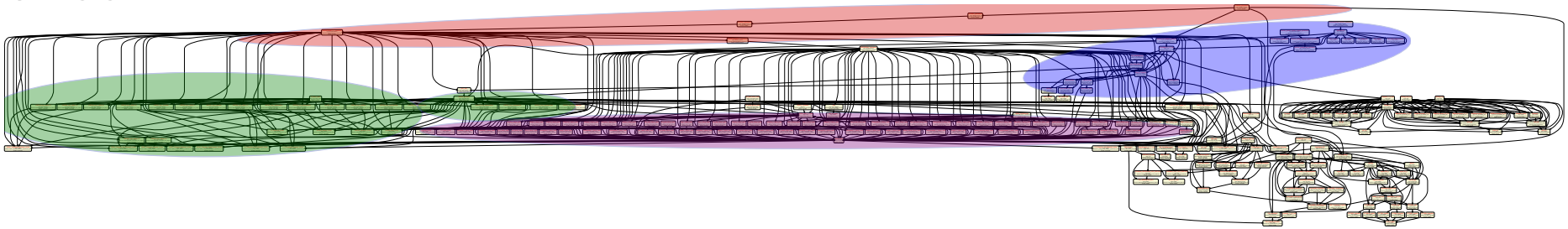
MSSM7:



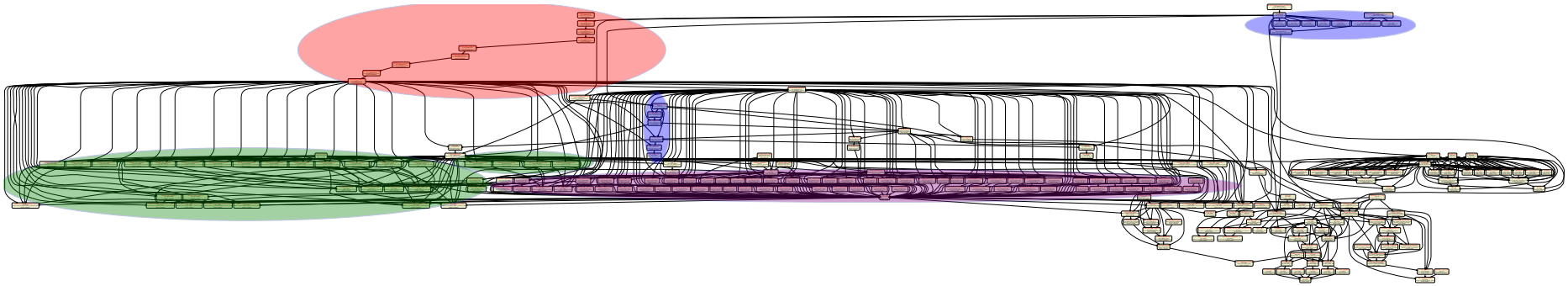
Red: Model parameter translations
Blue: Precision calculations
Green: LEP rates+likelihoods

Dependency Resolution

CMSSM:



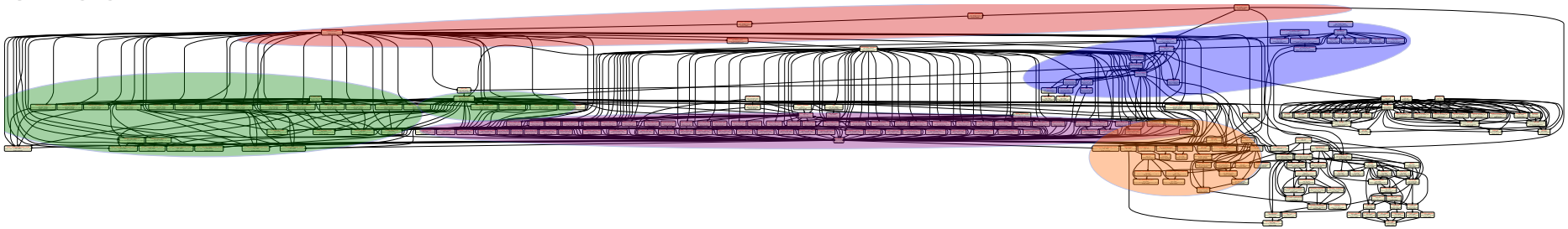
MSSM7:



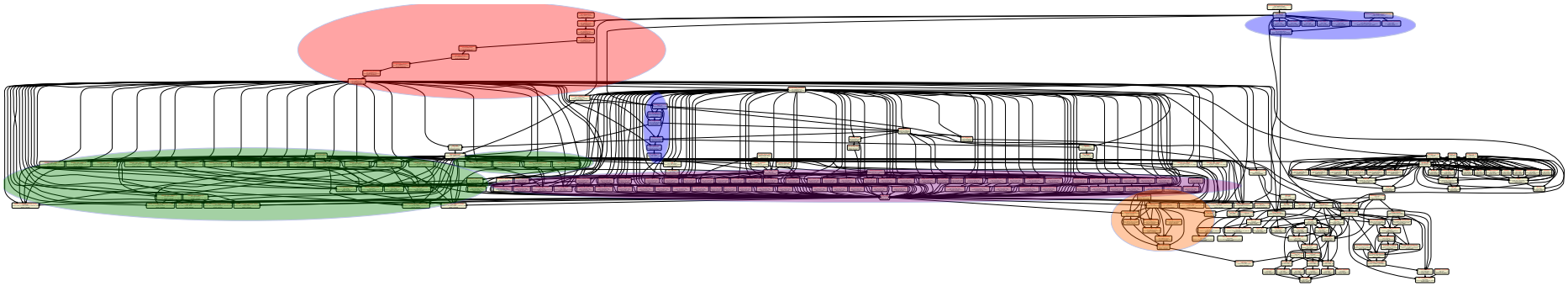
Red: Model parameter translations
Blue: Precision calculations
Green: LEP rates+likelihoods
Purple: Decays

Dependency Resolution

CMSSM:



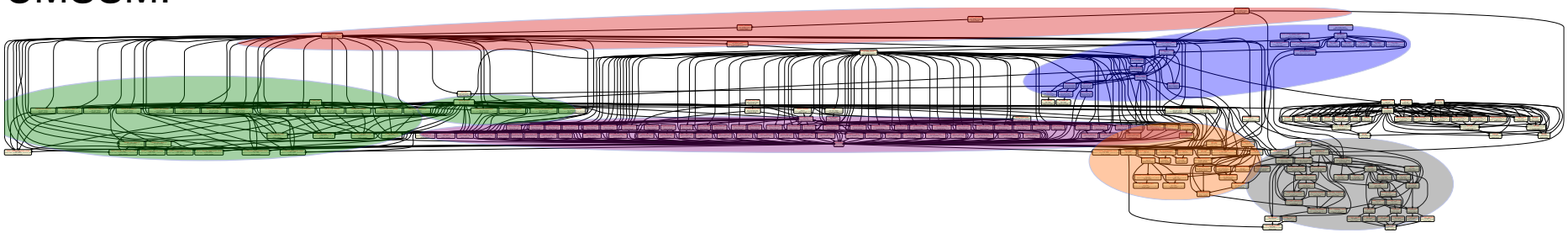
MSSM7:



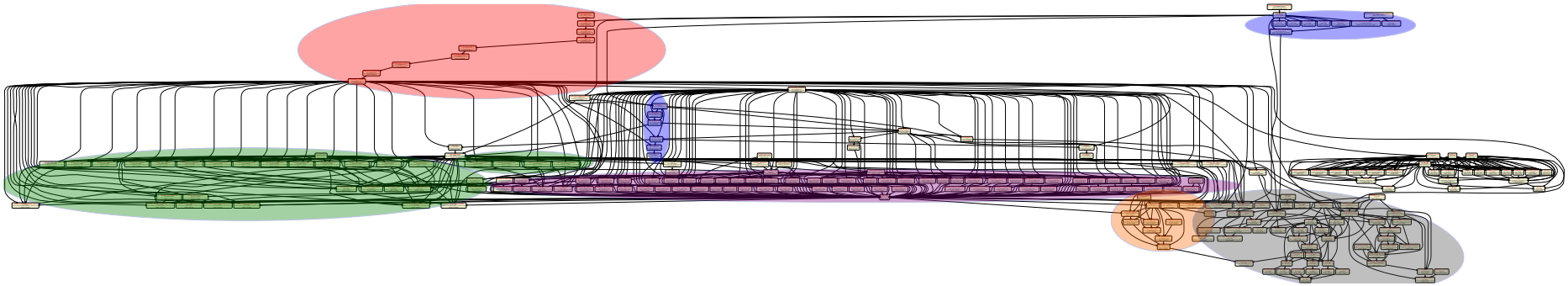
Red: Model parameter translations
Blue: Precision calculations
Green: LEP rates+likelihoods
Purple: Decays
Orange: LHC observables and likelihoods

Dependency Resolution

CMSSM:



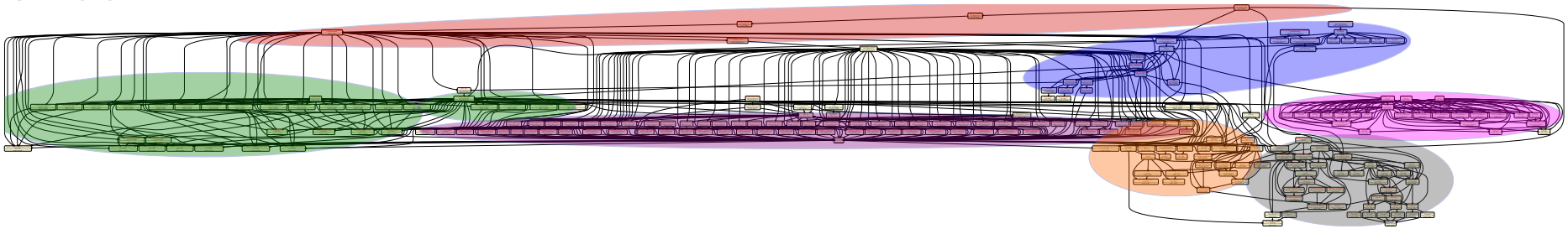
MSSM7:



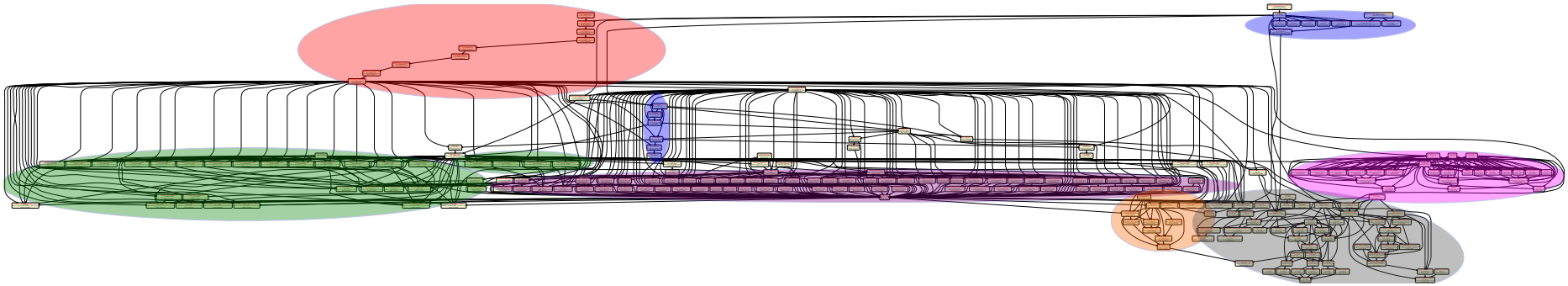
Red: Model parameter translations
Blue: Precision calculations
Green: LEP rates+likelihoods
Purple: Decays
Orange: LHC observables and likelihoods
Grey: DM direct, indirect and relic density

Dependency Resolution

CMSSM:

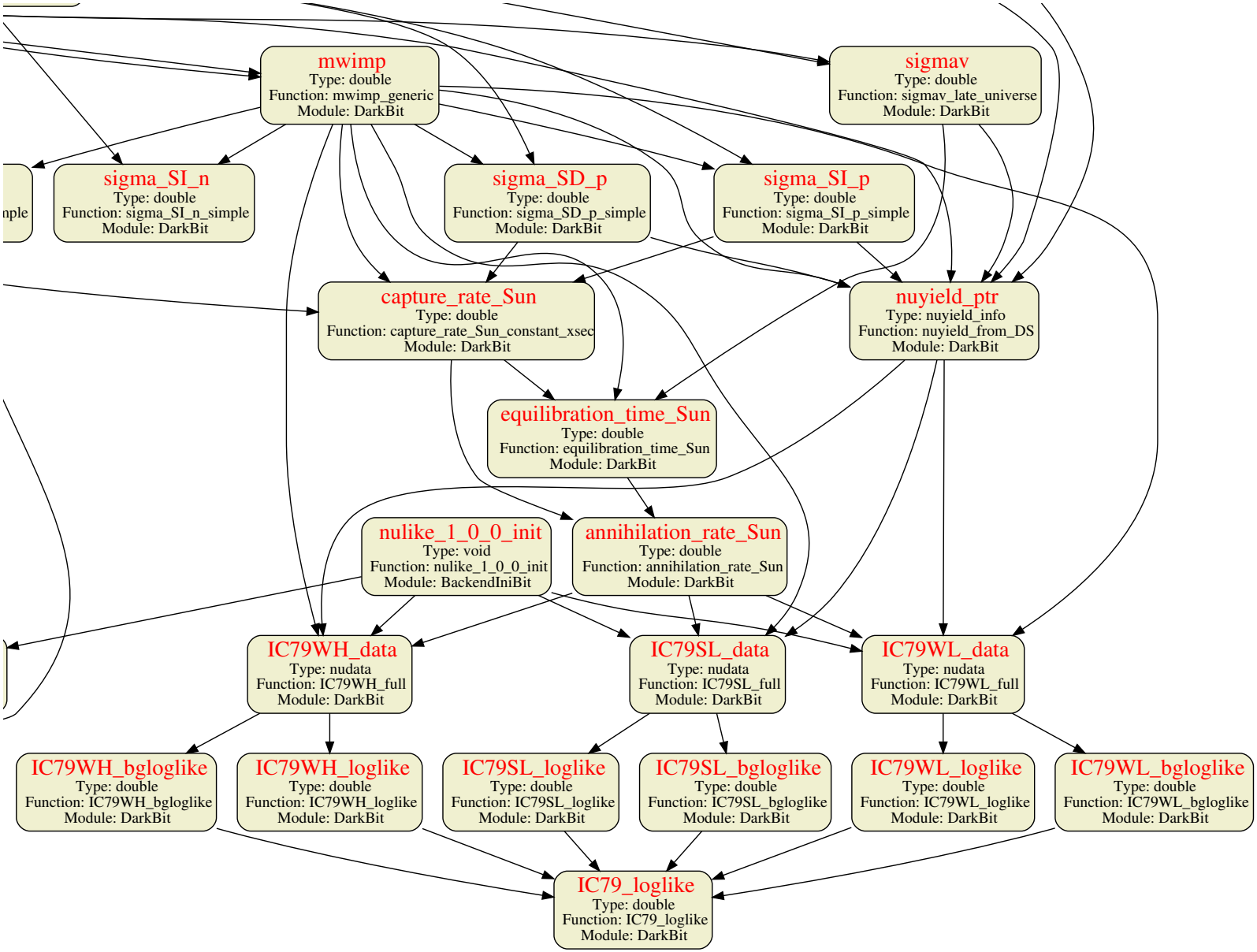


MSSM7:



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

Dependency Resolution: a closer look



Other nice technical features

- **Interface**: Simple, clean, intuitive and powerful YAML text file
- **Scanners**: Nested sampling, differential evolution, MCMC, genetic algorithm, t-walk...
- Mixed-mode **MPI + openMP** parallelisation, mostly automated $\rightarrow$ scales to 10k+ cores
- 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, HDF5, ...)
- more more more...

[1] if a backend won't compile/crashes/steals your credit card info, blame the authors (not us... except where we **are** the authors...)

Outlook

- GAMBIT is almost here:
  - $\rightarrow$ Global fits to many models for the first time
  - $\rightarrow$ Better global fits to familiar ones
  - $\rightarrow$ Highly modular, usable and extendable public code
  - $\rightarrow$ Faster, more complete and more consistent theory explorations + experimental analysis prototyping
- Series of 9 papers in preparation:
  - EW-scale MSSMs, CMSSM$\pm\epsilon$ (NUHM, etc), Scalar Singlet
  - DarkBit, ColliderBit, FlavBit, Spec+Decay+PrecisionBits
  - GAMBIT framework, ScannerBit
- After that:
  more models, more observables, more data!
  $\rightarrow\rightarrow\rightarrow$ LHC Run 2!

Backup Slides

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)

Interface: yaml file

Basic interface for a scan is a YAML initialisation file

- specify parameters, ranges, priors
- select likelihood components
- select other observables to calculate
- define generic rules for how to fill dependencies
- define generic rules for options to be passed to module functions
- set global options (scanner, errors/warnings, logging behaviour, etc)

```yaml
Parameters:
  StandardModel_SLHA2: !import StandardModel_SLHA2_default
  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
```

Interface: yaml file

Basic interface for a scan is a YAML initialisation file

- specify parameters, ranges, priors
- select likelihood components
- select other observables to calculate
- define generic rules for how to fill dependencies
- define generic rules for options to be passed to module functions
- set global options (scanner, errors/warnings, logging behaviour, etc)

```
Parameters:
  StandardModel_SLHA2: !import StandardModel_SLHA2_default
  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
```

Interface: yaml file

Basic interface for a scan is a YAML initialisation file

- specify parameters, ranges, priors
- select likelihood components
- select other observables to calculate
- define generic rules for how to fill dependencies
- define generic rules for options to be passed to module functions
- set global options (scanner, errors/warnings, logging behaviour, etc)

```yaml
Parameters:
  StandardModel_SLHA2: !import StandardModel_SLHA2_default
  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
```

Interface: yaml file

Basic interface for a scan is a YAML initialisation file

- specify parameters, ranges, priors
- select likelihood components
- select other observables to calculate
- define generic rules for how to fill dependencies
- define generic rules for options to be passed to module functions
- set global options (scanner, errors/warnings, logging behaviour, etc)

```yaml
Parameters:
  StandardModel_SLHA2: !import StandardModel_SLHA2_default
  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
```

Interface: yaml file

Basic interface for a scan is a YAML initialisation file

- specify parameters, ranges, priors
- select likelihood components
- select other observables to calculate
- define generic rules for how to fill dependencies
- define generic rules for options to be passed to module functions
- set global options (scanner, errors/warnings, logging behaviour, etc)

```
Parameters:
  StandardModel_SLHA2: !import StandardModel_SLHA2_default
  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
```