Dark matter theory
Implications and future prospects for *Fermi*

Pat Scott

Imperial College London

Finding dark matter and new physics

Even the simplest particle theories have *many* different consequences. . .

Finding dark matter and new physics

Even the simplest particle theories have *many* different consequences. . . theoretical

Finding dark matter and new physics

Even the simplest particle theories have *many* different consequences. . . theoretical, experimental

Finding dark matter and new physics

Even the simplest particle theories have *many* different consequences. . . theoretical, experimental, computational. . .

Finding dark matter and new physics

Even the simplest particle theories have *many* different consequences. . . theoretical, experimental, computational. . .

Gamma rays and *Fermi* are an important part of this, but need to be considered in the context of real models and constraints coming from complementary probes:

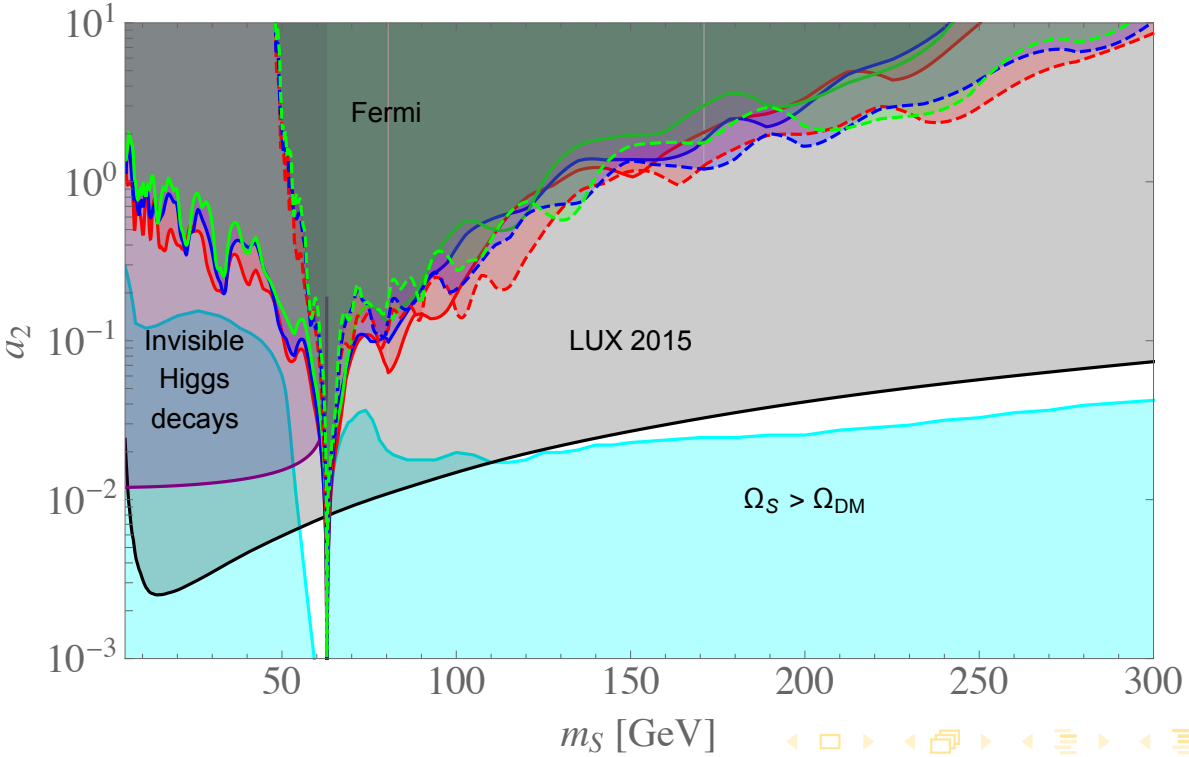
- realistic annihilation branching fractions
- direct detection
- collider searches
- relic density considerations
- related particles $\rightarrow$ precision/flavour physics

An illustrative example: lines from singlet dark matter

$$\mathcal{L}_S = -\frac{\mu_S^2}{2} S^2 - \frac{\lambda_{hs}}{2} S^2 H^\dagger H + \ldots$$

Claim: gamma-ray lines are an important constraint on scalar singlet dark matter

(e.g. arXiv:1508.04418, arXiv:1412.1105, arXiv:1009.5337)



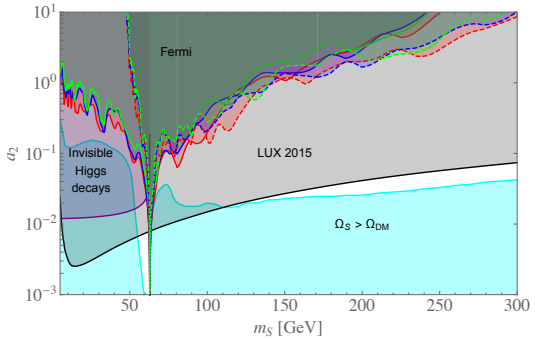
An illustrative example: lines from singlet dark matter

$$\mathcal{L}_S = -\frac{\mu_S^2}{2} S^2 - \frac{\lambda_{hs}}{2} S^2 H^\dagger H + \ldots$$

Claim: gamma-ray lines are an important constraint on scalar singlet dark matter

Problem: only true when

- singlet-Higgs coupling λ_{hs} is large
- Singlet mass $m_S \sim m_{h^0}/2$, on the resonance where S annihilates via SM Higgs h^0
- but the relic density is very low when either of these is the case $\rightarrow$ strongly suppresses the signal



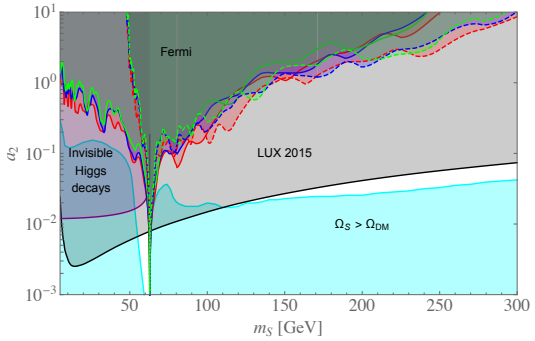
An illustrative example: lines from singlet dark matter

$$\mathcal{L}_S = -\frac{\mu_S^2}{2} S^2 - \frac{\lambda_{hs}}{2} S^2 H^\dagger H + \ldots$$

Claim: gamma-ray lines are an important constraint on scalar singlet dark matter

Problem: only true when

- singlet-Higgs coupling λ_{hs} is large
- Singlet mass $m_S \sim m_{h^0}/2$, on the resonance where S annihilates via SM Higgs h^0
- but the relic density is very low when either of these is the case $\rightarrow$ strongly suppresses the signal



$\Longrightarrow$ channel is never observable + therefore basically irrelevant for this model

A general word on how (not) to recast indirect detection limits

- Indirect limits always presented in terms of hard process final states
- Actual experiments do not measure those final states – they detect one type of SM particle produced later: γs, νs, etc
- Limits as presented cannot be combined and applied to models with mixed final states (= all non-toy models)
- Proper treatment of indirect detection for BSM searches requires full phenomenological recast abilities
  $\rightarrow$ full experimental *and* theoretical treatment at the same time

(
- Actually not so dissimilar to the LHC in this respect...
- extra complications with neutrinos from capture-annihilation balance...
)

OK OK, enough pontificating and complaining already...

OK OK, enough pontificating and complaining already...

So how to achieve all of this consistently? And how to be able to redo it all easily for many different dark matter models?

OK OK, enough pontificating and complaining already...

So how to achieve all of this consistently? And how to be able to redo it all easily for many different dark matter models?
$\Longrightarrow$ GAMBIT

GAMBIT: The Global And Modular BSM Inference Tool

gambit.hepforge.org

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

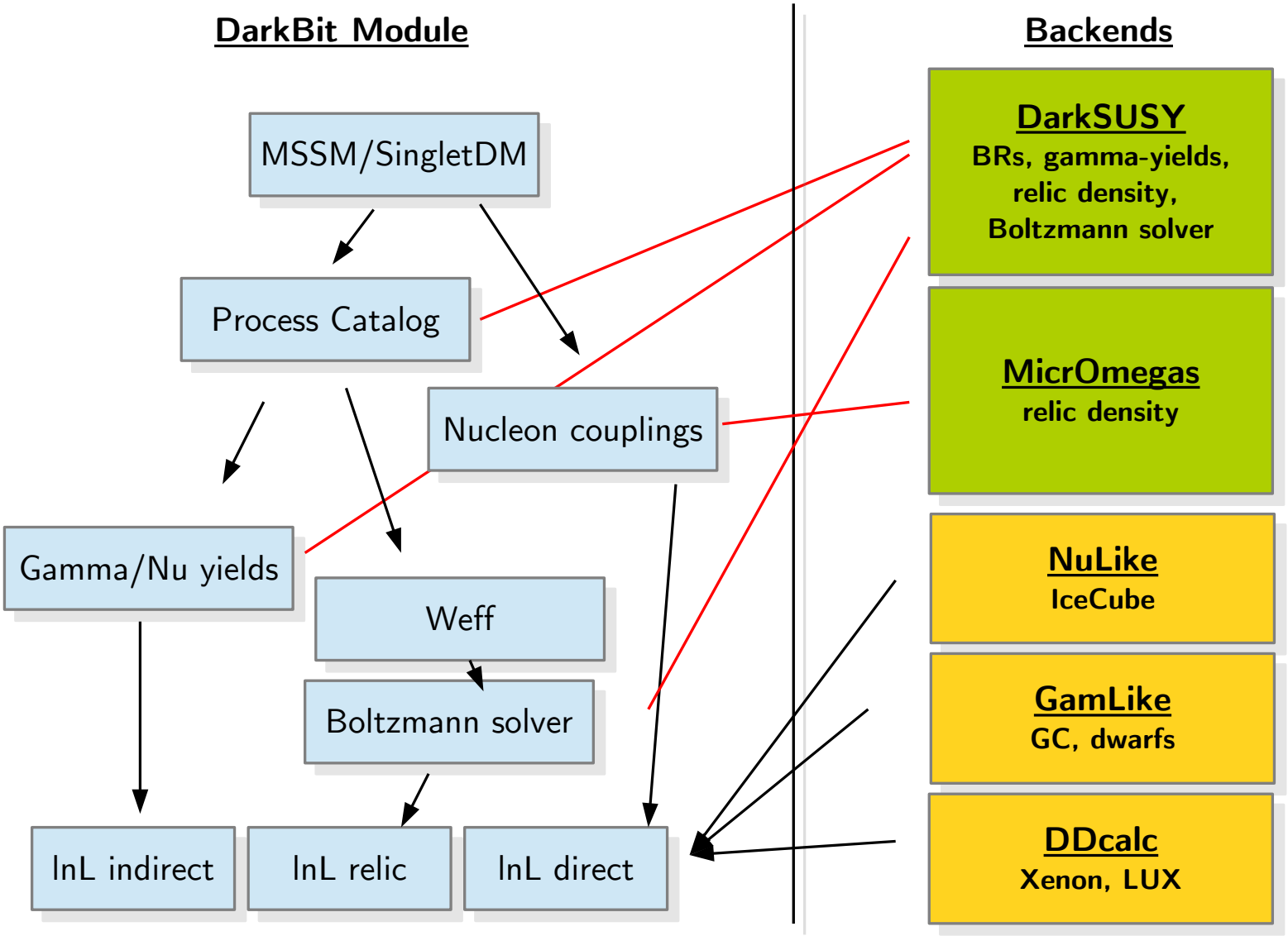
ATLAS	F. Bernlochner, A. Buckley, P. Jackson, M. White
LHCb	M. Chrząszcz, N. Serra
Belle-II	F. Bernlochner, P. Jackson
Fermi-LAT	J. Edsjö, G. Martinez, P. Scott
CTA	C. Balázs, T. Bringmann, M. White
CMS	C. Rogan
IceCube	J. Edsjö, P. Scott
XENON/DARWIN	B. Farmer, R. Trotta
Theory	P. Athron, C. Balázs, S. Bloor, T. Bringmann, J. Cornell, J. Edsjö, B. Farmer, A. Fowlie, T. Gonzalo, J. Harz, S. Hoof, F. Kahlhoefer, A. Kvellestad, F.N. Mahmoudi, J. McKay, A. Raklev, R. Ruiz, P. Scott, R. Trotta, A. Vincent, C. Weniger, M. White, S. Wild

29 Members in 9 Experiments, 12 major theory codes, 11 countries

arXiv:1705.07908 (EPJC 2017)

DarkBit

(arXiv:1705.07920)



DarkBit: gamLike

(arXiv:1705.07920)

C++ library with simple interface to most relevant likelihood functions from Fermi LAT and IACTs

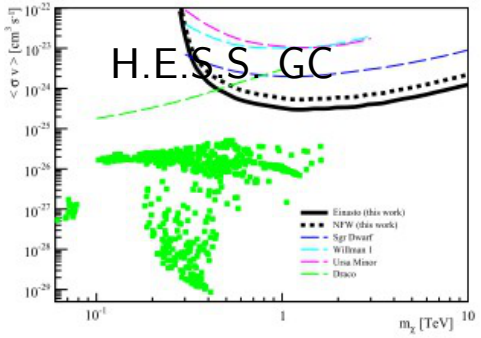
Particle physics input:

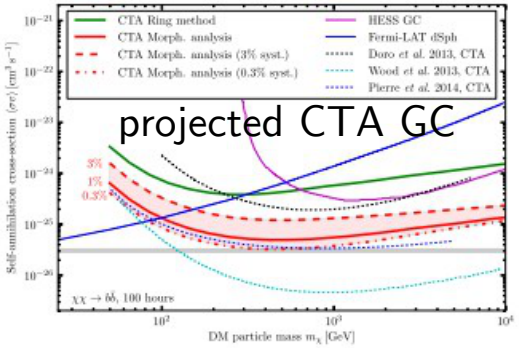
$$\frac{1}{m_\chi^2}\frac{d\sigma v}{dE}(v,E)$$

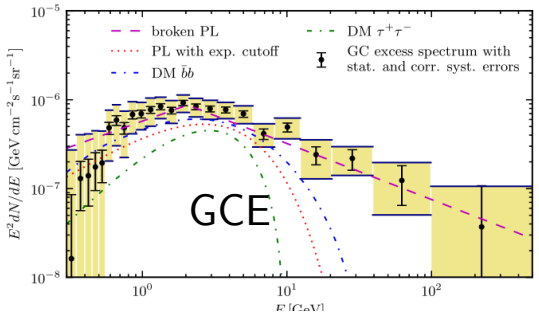
Output: lnL

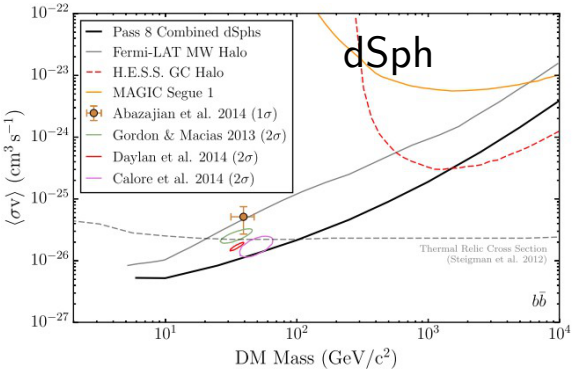
Uncertainties in the DM distribution (or astrophysical foregrounds) are internally marginalized over.

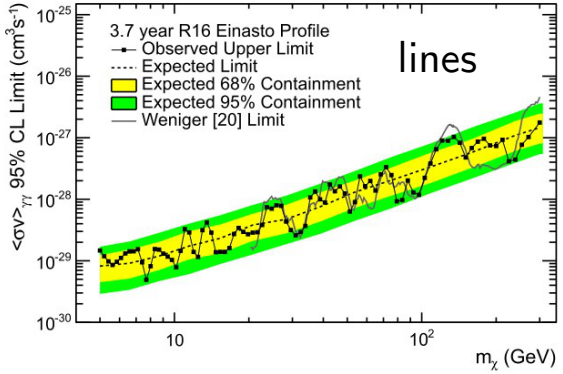
Correct treatment of energy dispersion and spectral singularities (lines, virtual internal Bremsstrahlung, boxes).



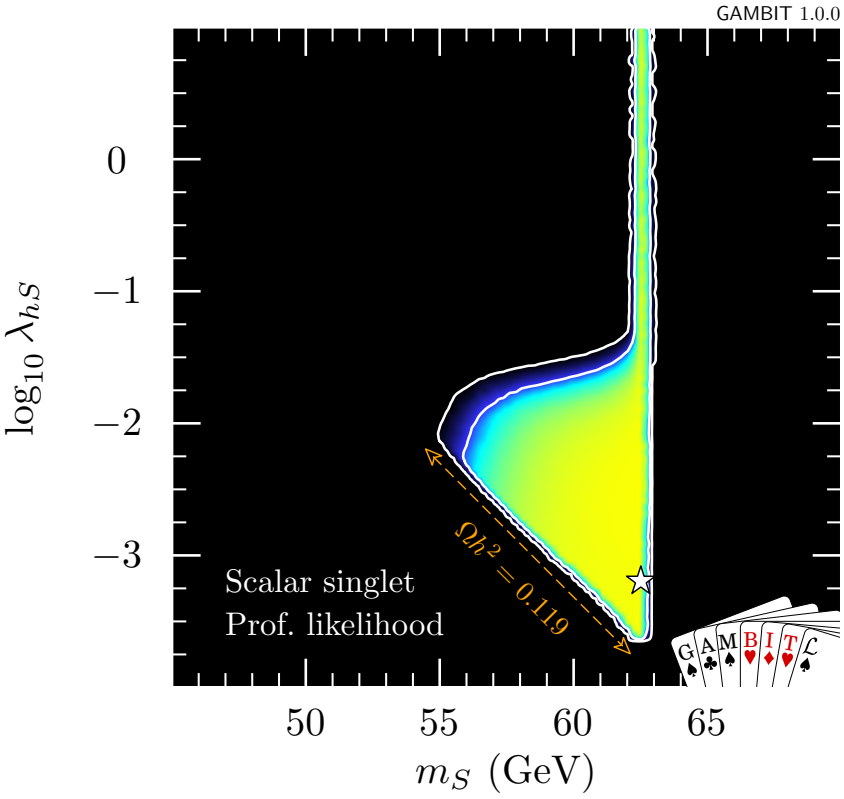


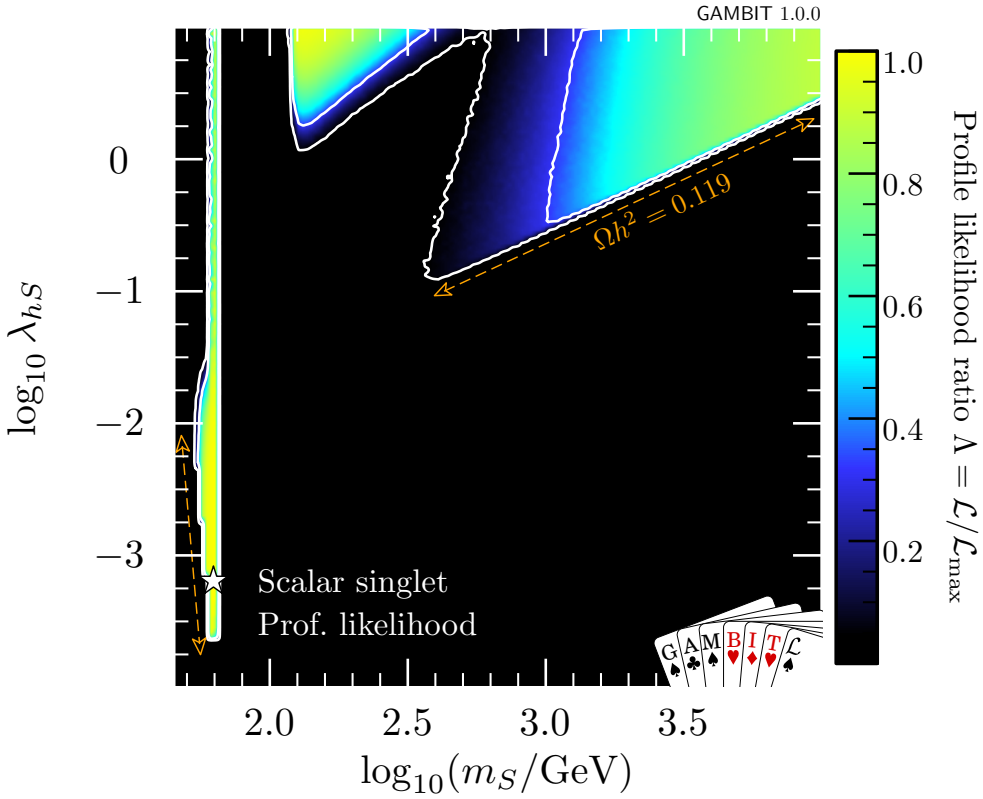






Scalar singlet DM (m_S, λ_{hS} + 13 nuisances) (arXiv:1705.07931)

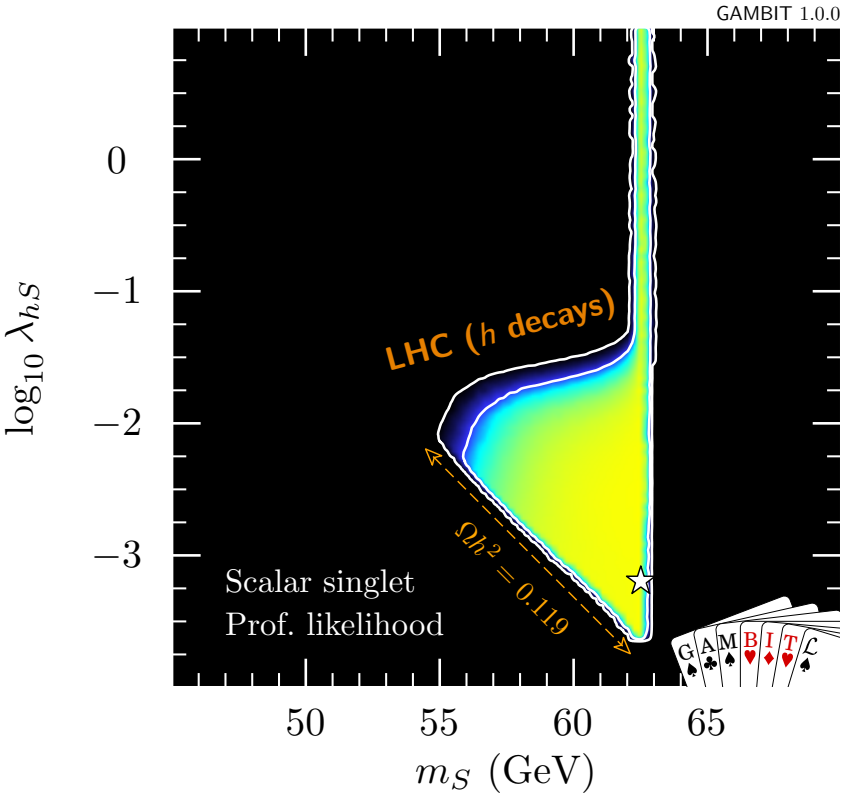


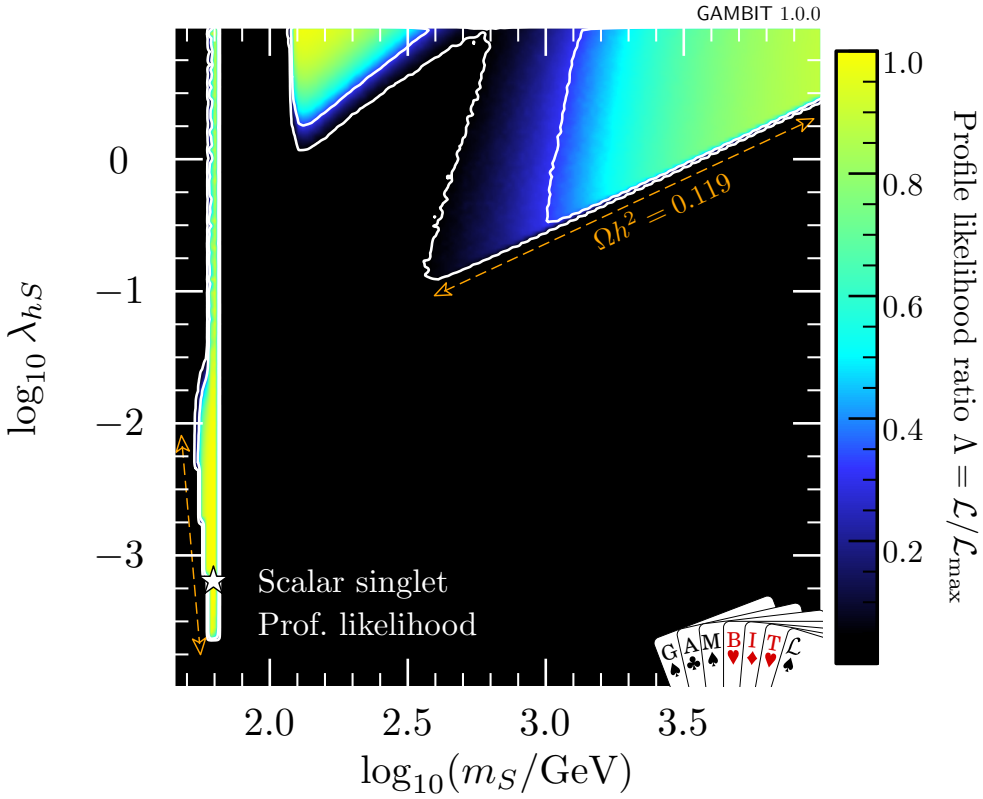


Simplest BSM example: $\mathcal{L}_S = -\frac{\mu_S^2}{2} S^2 - \frac{\lambda_{hs}}{2} S^2 H^\dagger H + \ldots$

All dark matter signals consistently scaled for predicted abundance

Scalar singlet DM (m_S, λ_{hS} + 13 nuisances) (arXiv:1705.07931)

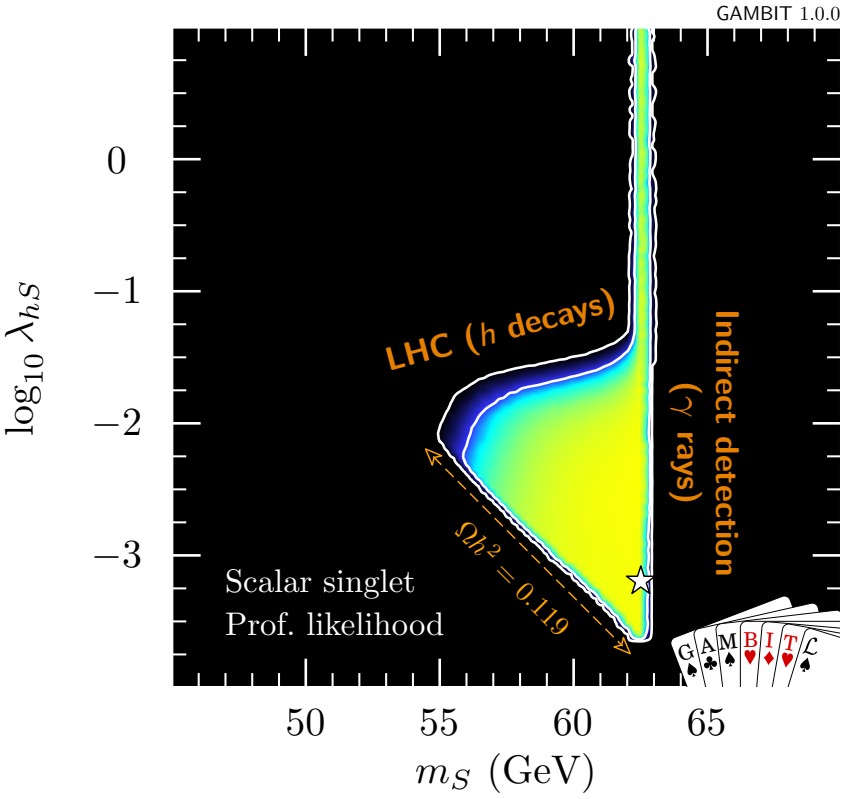


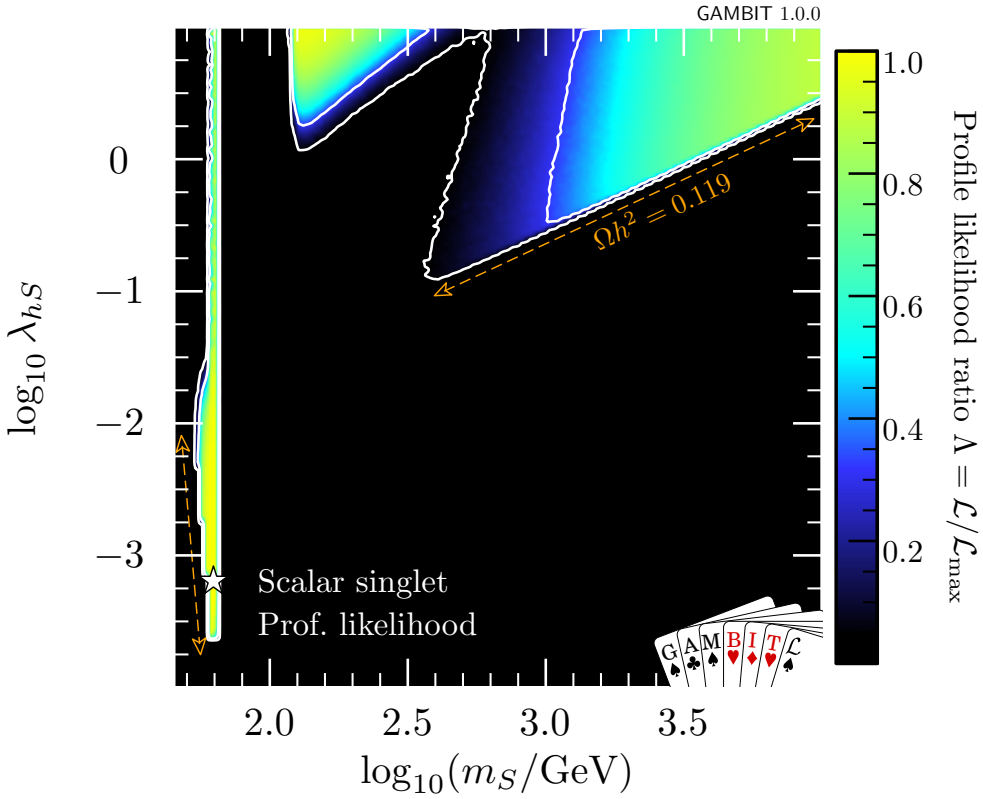


Simplest BSM example: $\mathcal{L}_S = -\frac{\mu_S^2}{2} S^2 - \frac{\lambda_{hs}}{2} S^2 H^\dagger H + \ldots$

All dark matter signals consistently scaled for predicted abundance

Scalar singlet DM (m_S, λ_{hS} + 13 nuisances) (arXiv:1705.07931)

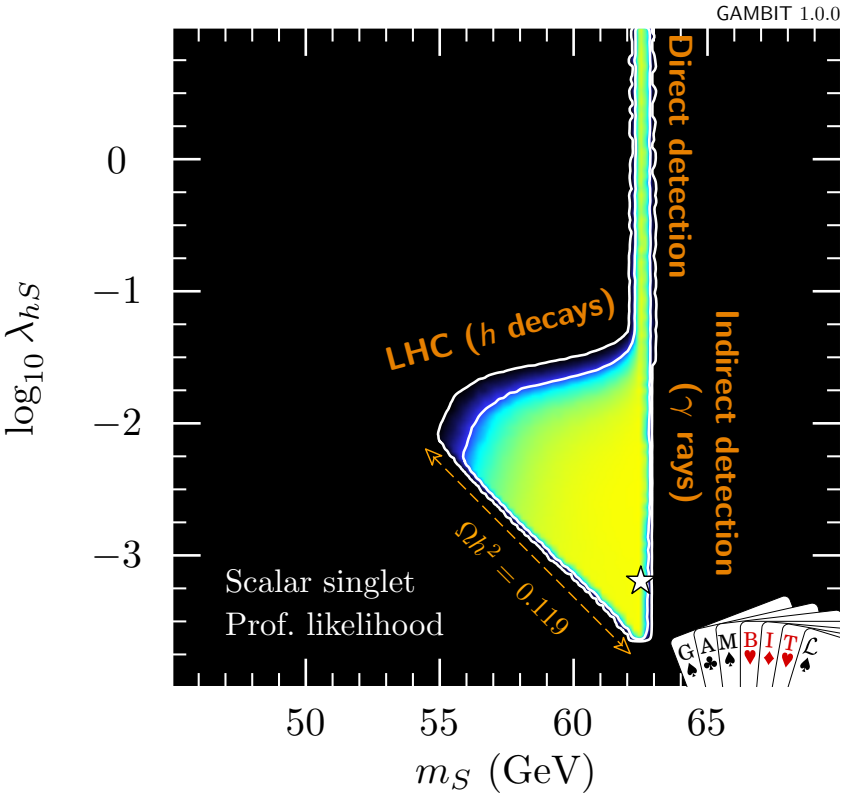


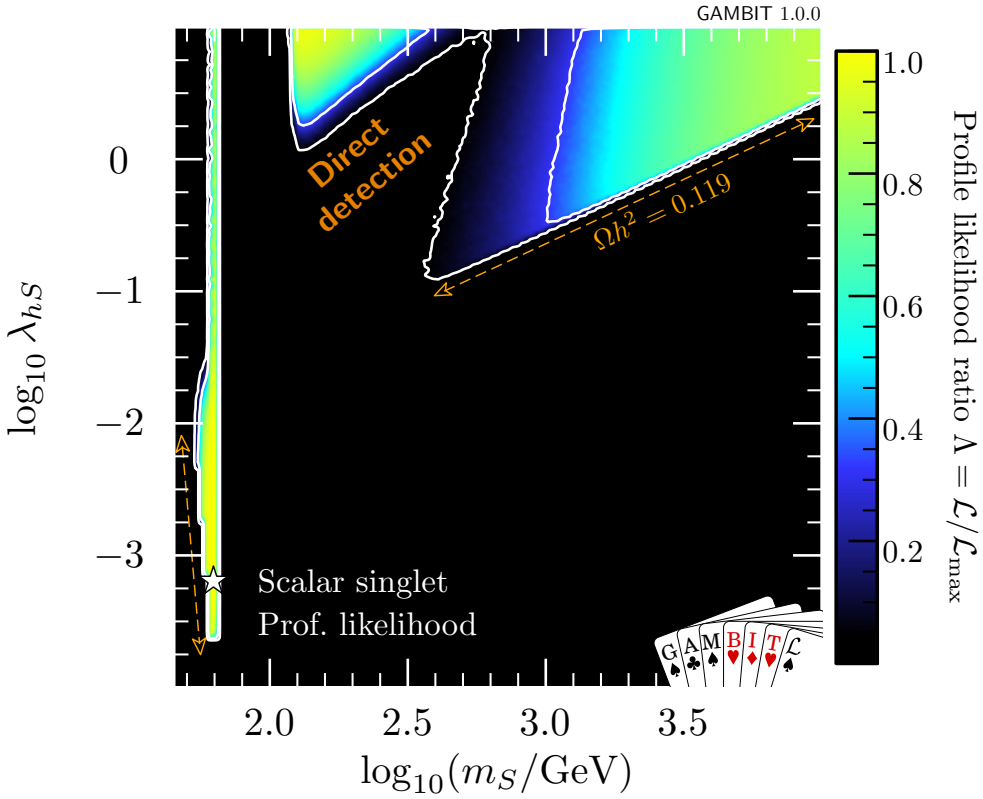


Simplest BSM example: $\mathcal{L}_S = -\frac{\mu_S^2}{2} S^2 - \frac{\lambda_{hs}}{2} S^2 H^\dagger H + \ldots$

All dark matter signals consistently scaled for predicted abundance

Scalar singlet DM (m_S, λ_{hS} + 13 nuisances) (arXiv:1705.07931)

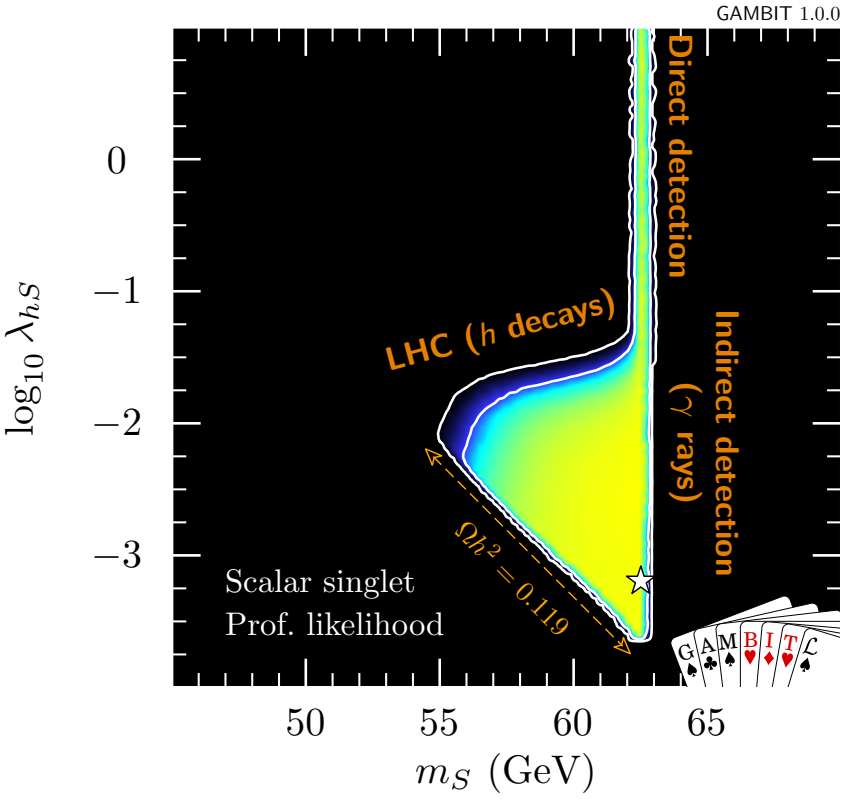


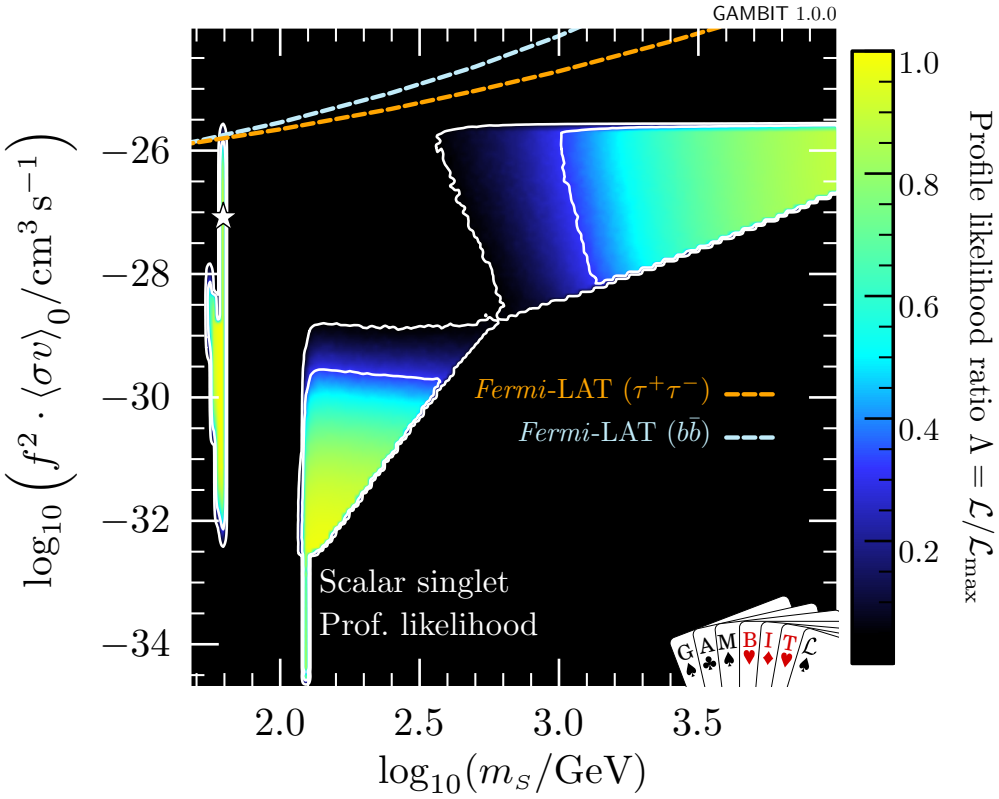


Simplest BSM example: $\mathcal{L}_S = -\frac{\mu_S^2}{2} S^2 - \frac{\lambda_{hs}}{2} S^2 H^\dagger H + \ldots$

All dark matter signals consistently scaled for predicted abundance

Scalar singlet DM (m_S, λ_{hS} + 13 nuisances) (arXiv:1705.07931)

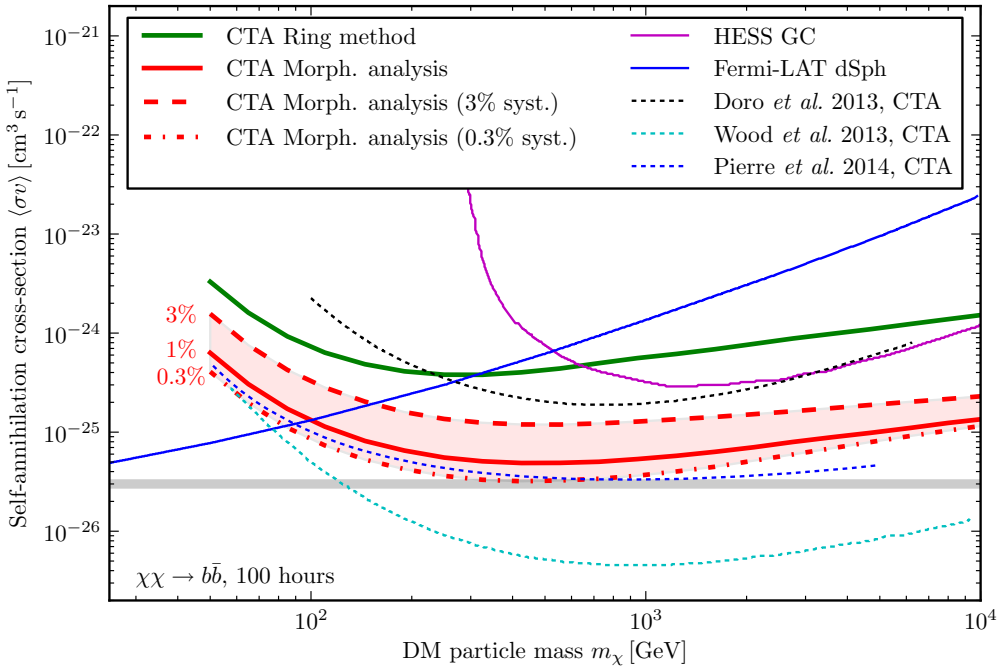




Simplest BSM example: $\mathcal{L}_S = -\frac{\mu_S^2}{2} S^2 - \frac{\lambda_{hs}}{2} S^2 H^\dagger H + \ldots$

All dark matter signals consistently scaled for predicted abundance

An advance health warning about CTA projections



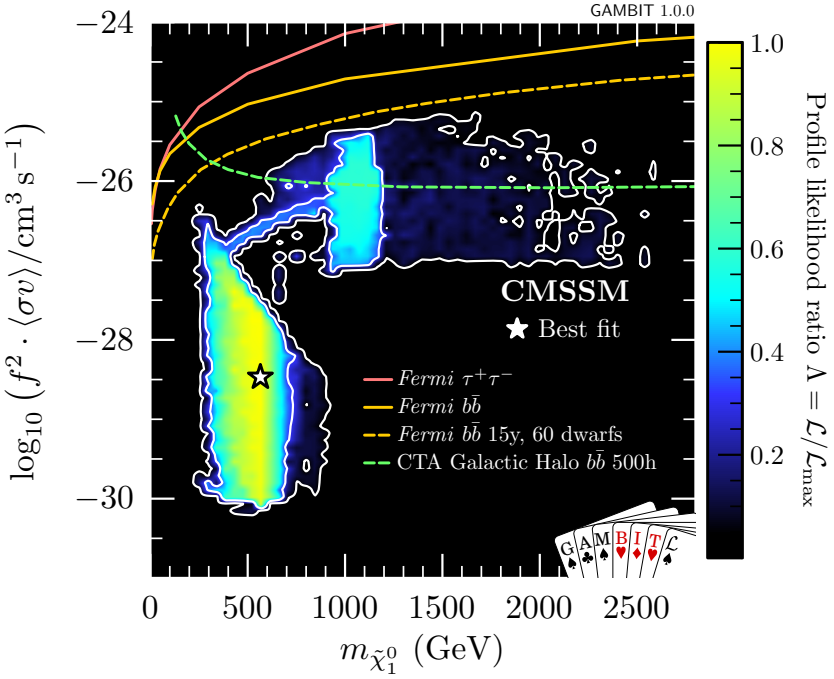
Silverwood, Weniger, PS, Bertone, arXiv:1408.4131 (JCAP 2015)

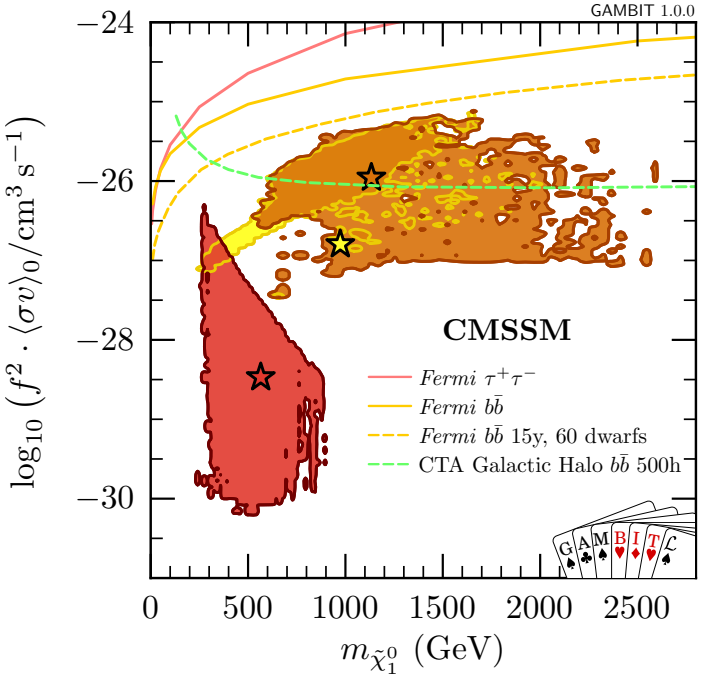
CTA sensitivity will be dominantly determined by:

- statistical variation of diffuse emission
- systematics (differential error over FoV on effective area, event reconstruction, atmospheric conditions)

Official CTA projections assume 0% systematic errors & 500 hr exp.
. . . with 0.3% sys err, CTA is already systematic-limited @200 hr. . .

GUT-scale MSSM (CMSSM) (arXiv:1705.07935)

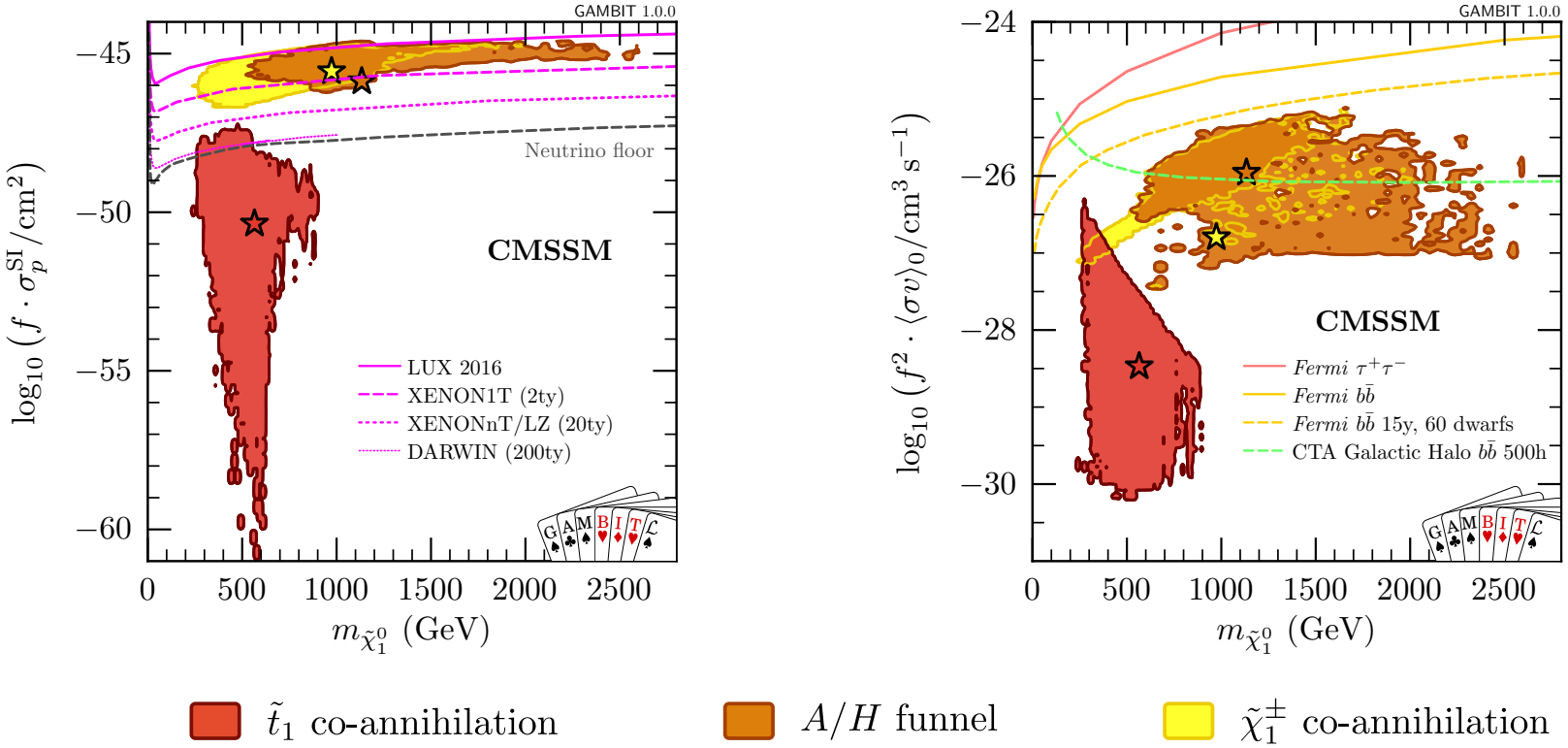




$\tilde{t}_1$ co-annihilation A/H funnel $\tilde{\chi}_1^\pm$ co-annihilation

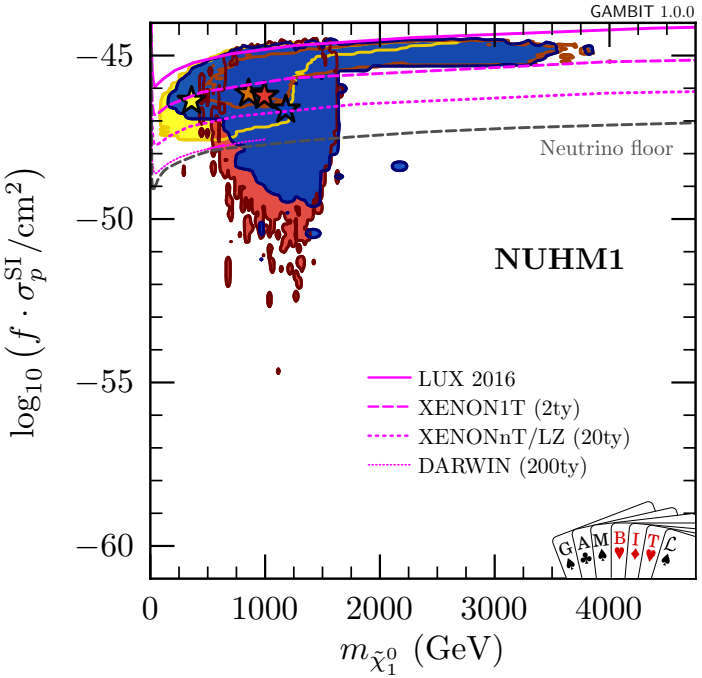
- m_0, $m_{\frac{1}{2}}$, A_0, $\tan\beta$ + 5 nuisances
- H/A^0 funnel, $\chi^\pm$ co-annihilation, $\tilde{t}$ co-annihilation
- $\tilde{\tau}$ **co-annihilation now ruled out**
- Includes LUX 2016, Panda-X + direct simulation of LHC Run 1 & Run 2 limits.

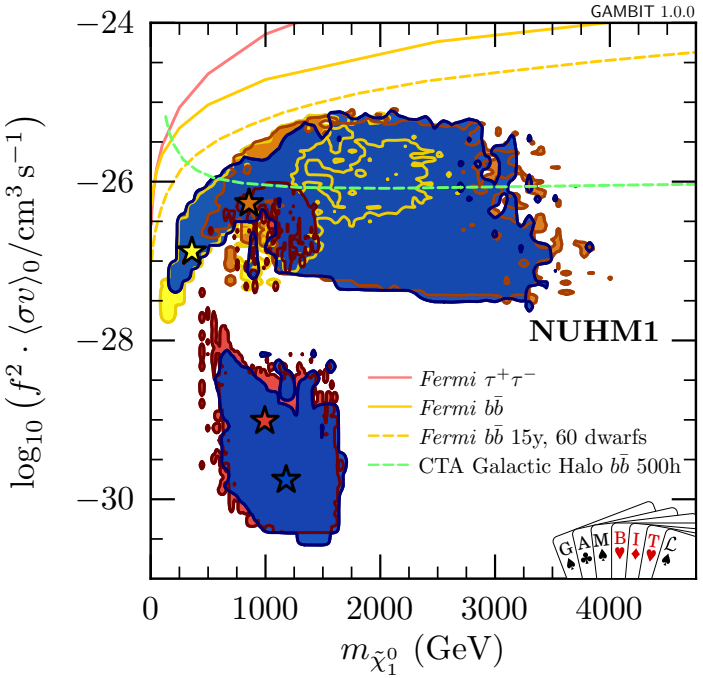
GUT-scale MSSM (CMSSM) (arXiv:1705.07935)



- m_0, $m_{\frac{1}{2}}$, A_0, $\tan\beta$ + 5 nuisances
- H/A^0 funnel, $\chi^\pm$ co-annihilation, $\tilde{t}$ co-annihilation
- $\tilde{\tau}$ **co-annihilation now ruled out**
- $\tilde{t}$ co-annihilation tough to get at with DD, ID or LHC – but some hope with LHC for seeing light stops at the 'tip'

GUT-scale MSSM (NUHM1) (arXiv:1705.07935)

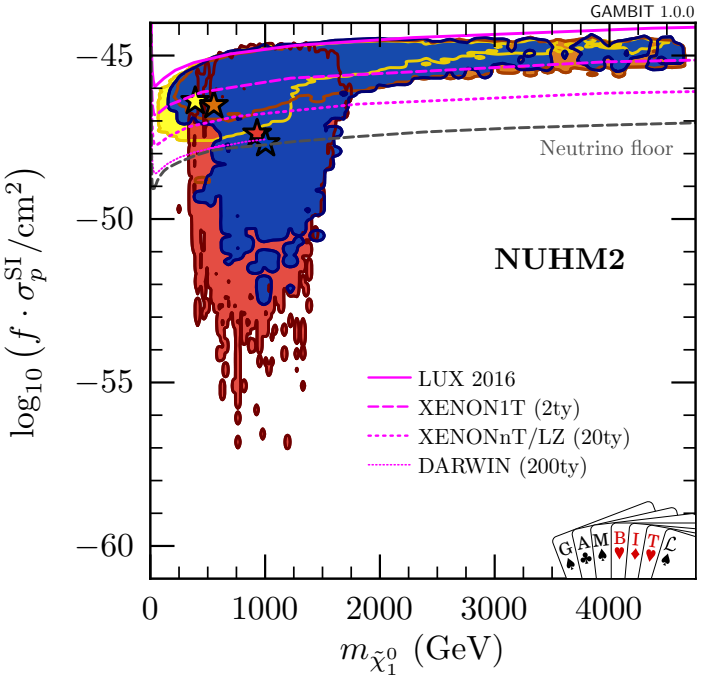


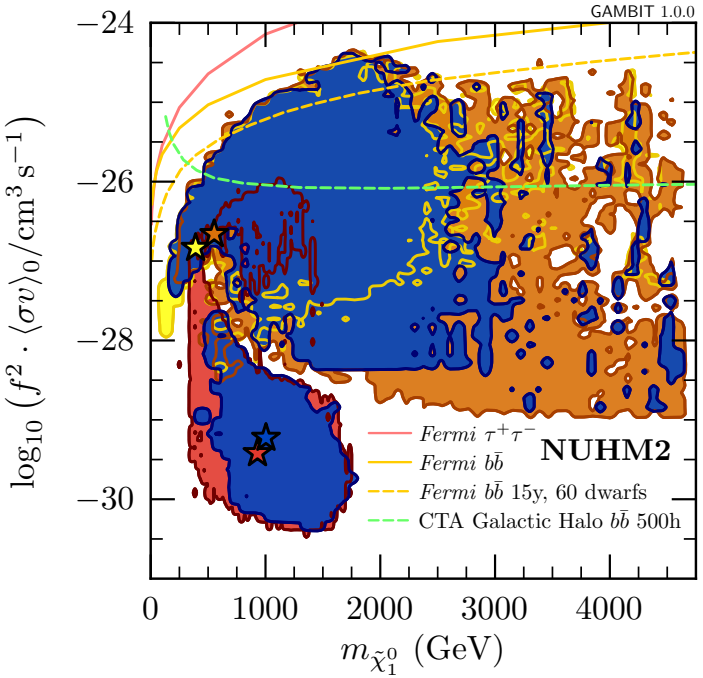


$\tilde{t}_1$ co-annihilation A/H funnel $\tilde{\chi}_1^\pm$ co-annihilation $\tilde{\tau}_1$ co-annihilation

- m_0, m_H, $m_{\frac{1}{2}}$, A_0, $\tan\beta$ + 5 nuisances
- H/A^0 funnel, $\chi^\pm$ co-annihilation, $\tilde{t}$ co-annihilation + $\tilde{\tau}$ co-annihilation due to additional freedom to fit m_h
- Includes LUX 2016, Panda-X + direct simulation of LHC Run 1 & Run 2 limits.

GUT-scale MSSM (NUHM2) (arXiv:1705.07935)

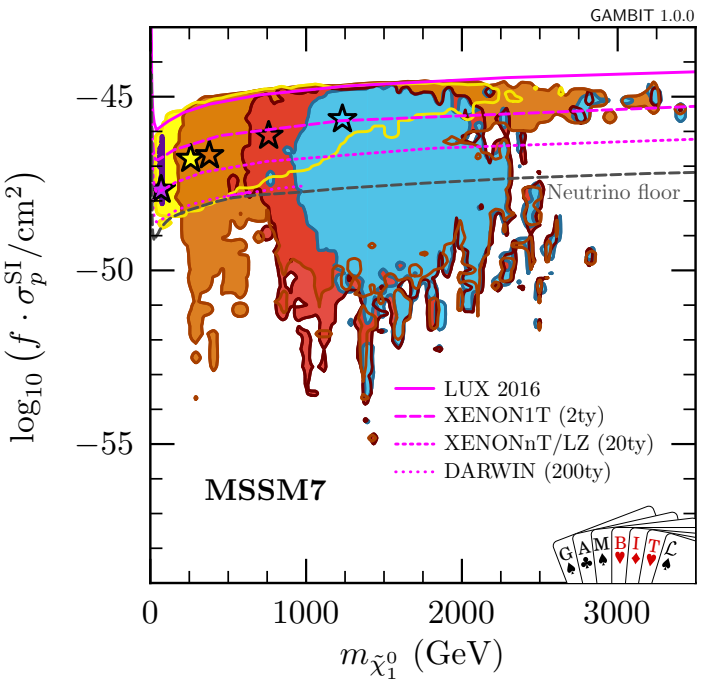


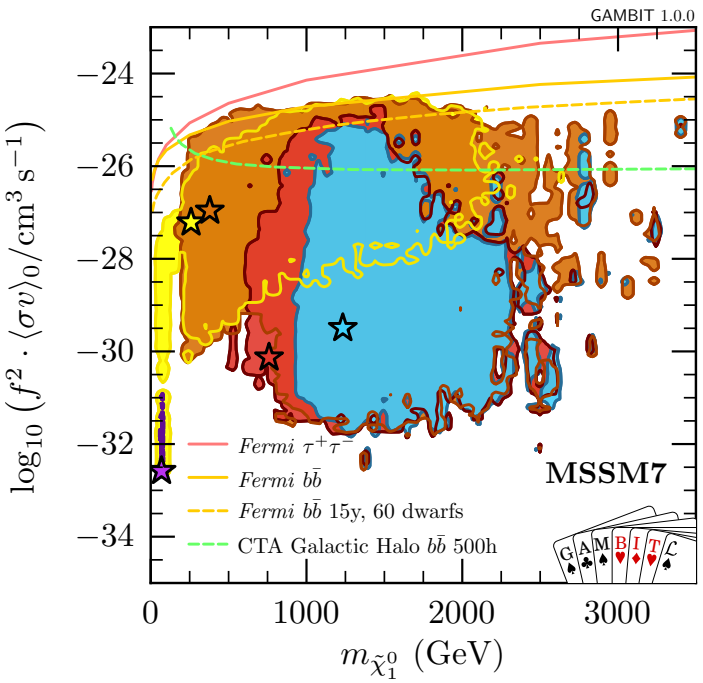


$\tilde{t}_1$ co-annihilation A/H funnel $\tilde{\chi}_1^\pm$ co-annihilation $\tilde{\tau}_1$ co-annihilation

- m_0, m_{H_u}, m_{H_d}, $m_{\frac{1}{2}}$, A_0, $\tan\beta$ + 5 nuisances
- H/A^0 funnel, $\chi^\pm$ co-annihilation, $\tilde{t}$ co-annihilation, $\tilde{\tau}$ co-annihilation
- Includes LUX 2016, Panda-X + direct simulation of LHC Run 1 & Run 2 limits.

Weak-scale MSSM (MSSM-7) (arXiv:1705.07917)





$\tilde{t}_1$ co-annihilation | A/H funnel | $\tilde{\chi}_1^\pm$ co-annihilation | $\tilde{b}_1$ co-annihilation | h/Z funnel

- $m_{\tilde{f}}$, M_2, A_u, A_d, m_{Hu}, m_{Hd}, $\tan\beta$ + 5 nuisances
- H/A^0 funnel, h/Z funnel, $\chi^\pm$ co-annihilation, $\tilde{t}/\tilde{b}$ co-annihilation
- Includes LUX 2016, Panda-X + direct simulation of LHC Run 1 & Run 2 limits.

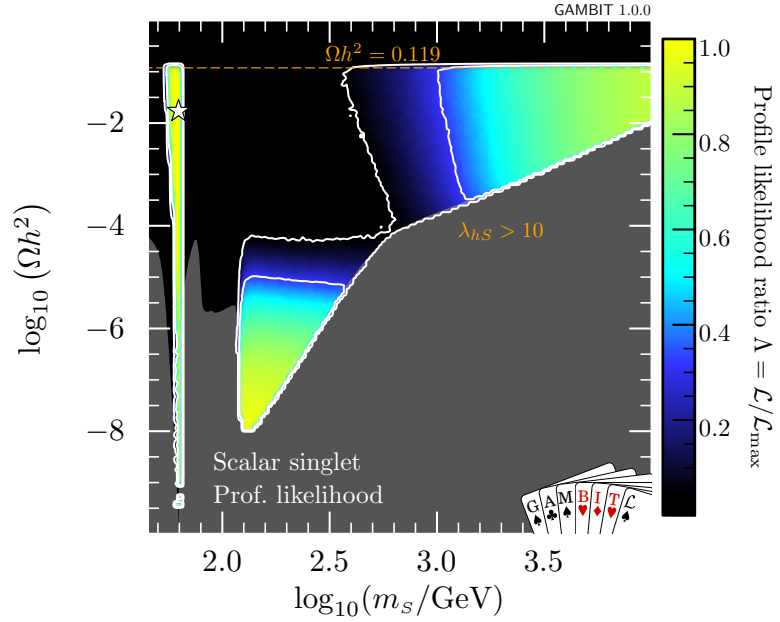
Summary

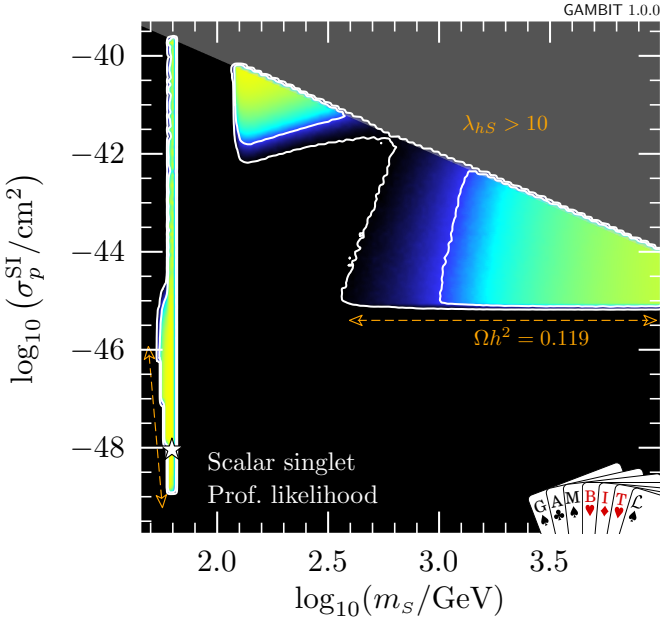
- Scalar singlet restricted to 'resonant triangle' and high-coupling/high-mass solutions
- Fermi dwarfs place limits in the region of the Higgs resonance
- Dwarfs are providing significant constraints on some versions of SUSY + will do more
- Heavy Higgs funnel models are especially promising targets for Fermi going forwards
- CTA will also probe many of the same models
- γ-rays are still $\sim$complementary with XENONnT/LZ/Panda-X, but analyses will need to be fast!

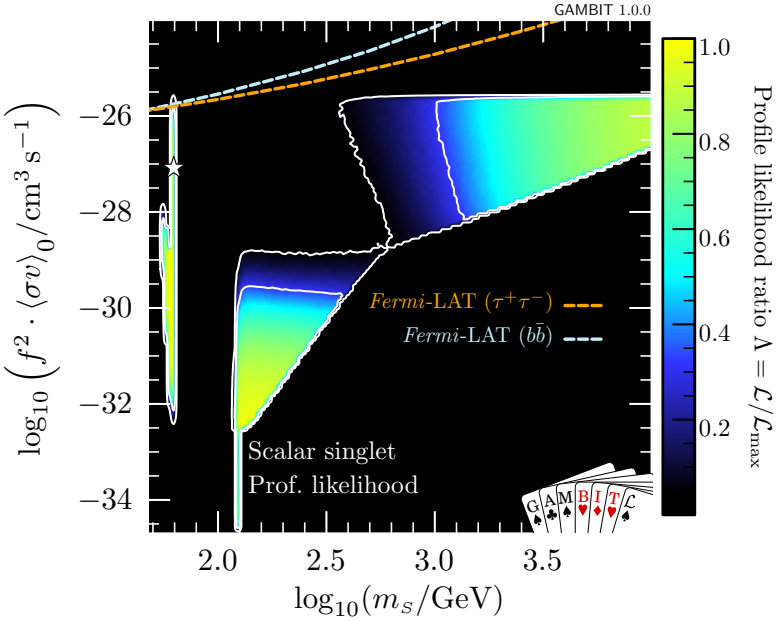
- GAMBIT results, samples, run files, best fits, benchmarks, etc are *all* available to download from Zenodo: www.zenodo.org/communities/gambit-official/
- GAMBIT code is public: gambit.hepforge.org

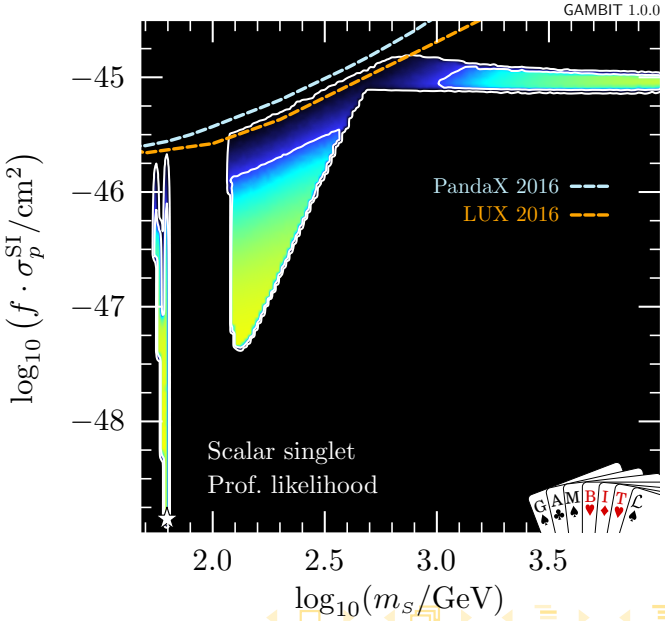
Backup slides

Some more scalar singlet DM (m_S, λ_{hS} + 13) (arXiv:1705.07931)



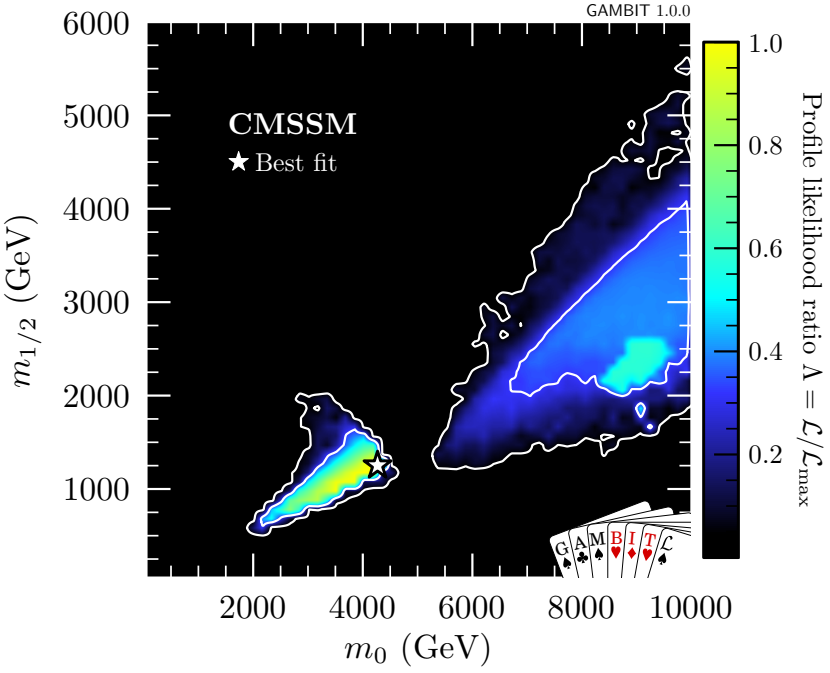


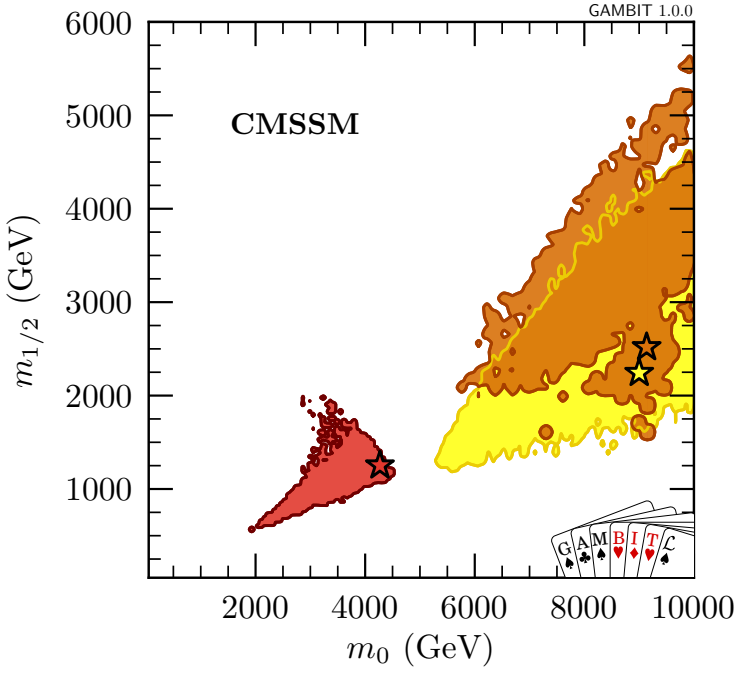




Some more CMSSM

(arXiv:1705.07935)

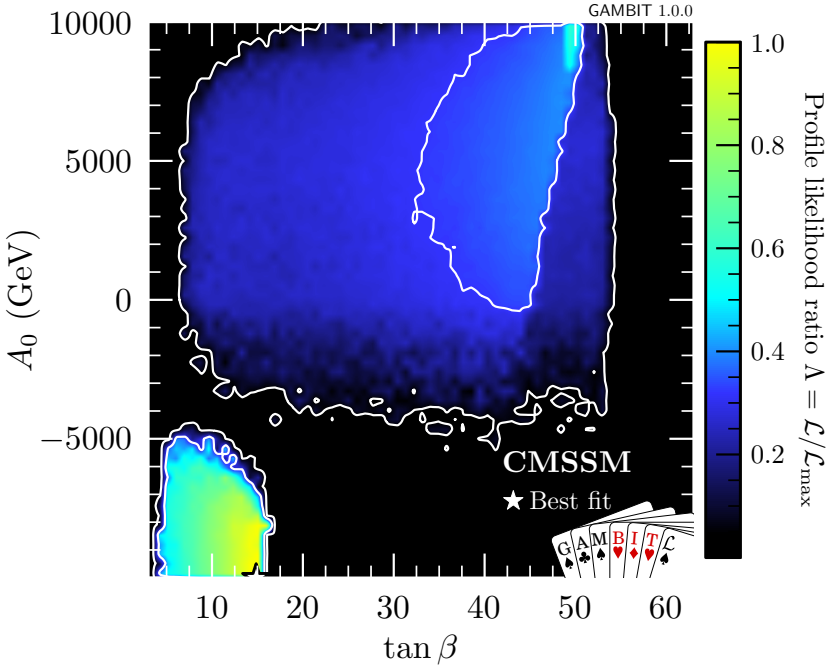


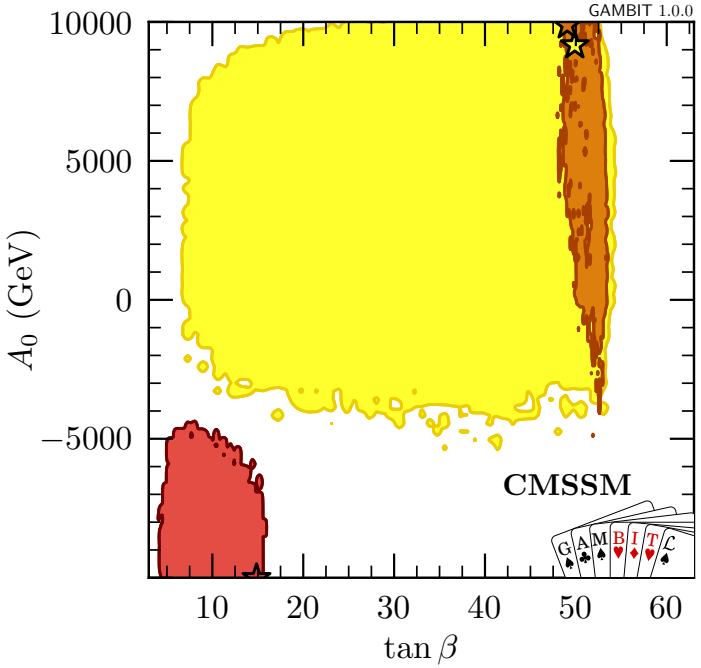


$\tilde{t}_1$ co-annihilation A/H funnel $\tilde{\chi}_1^{\pm}$ co-annihilation

Some more CMSSM

(arXiv:1705.07935)

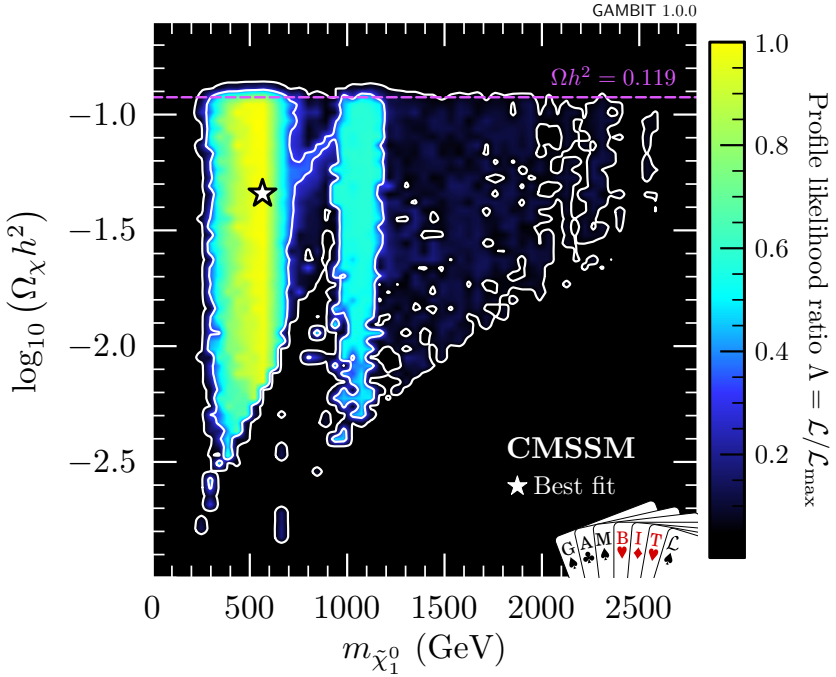


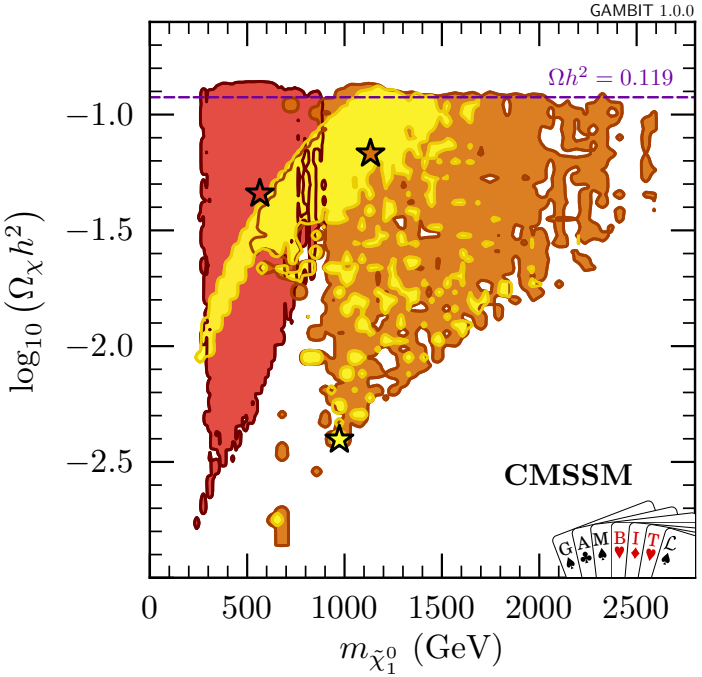


$\tilde{t}_1$ co-annihilation A/H funnel $\tilde{\chi}_1^{\pm}$ co-annihilation

Some more CMSSM

(arXiv:1705.07935)

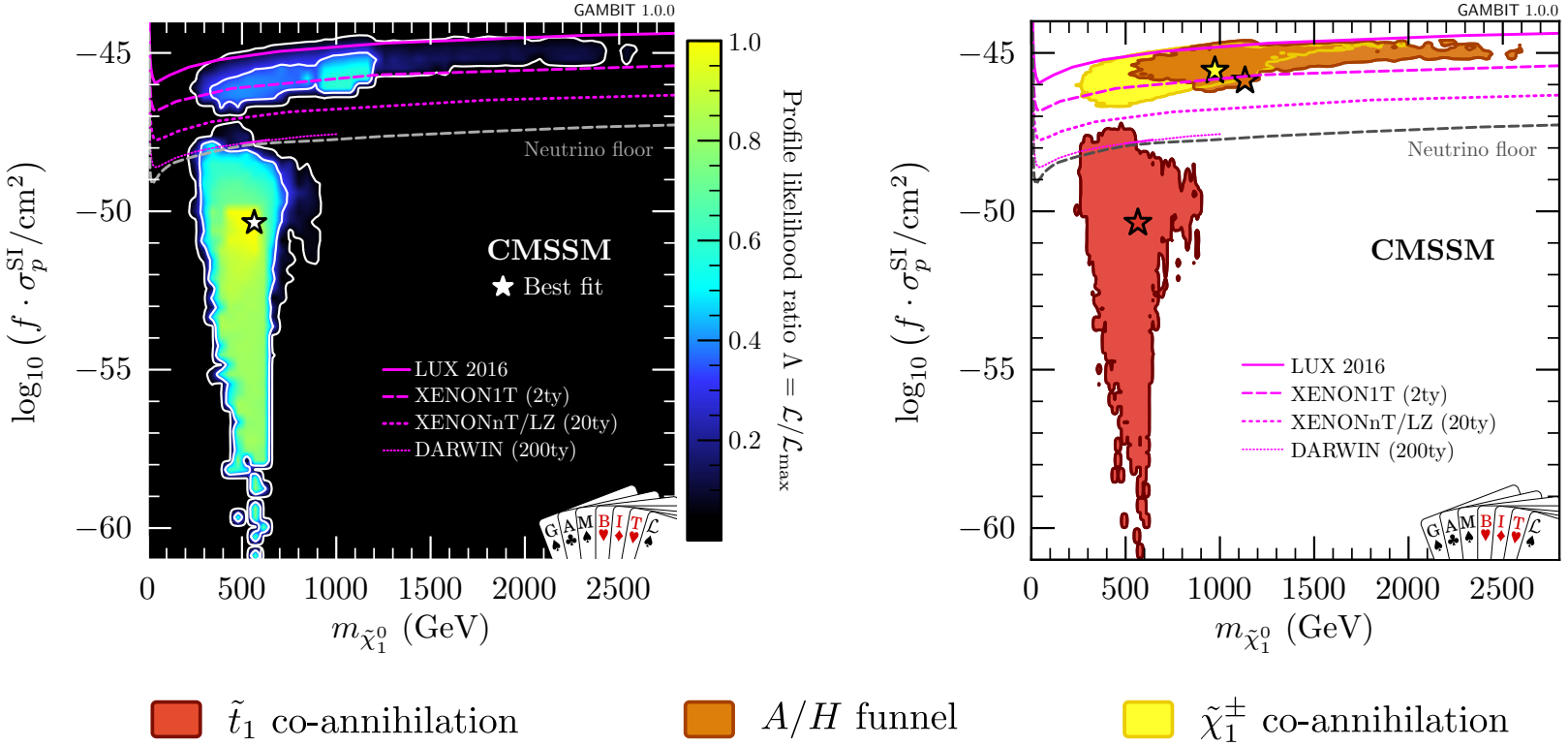




$\tilde{t}_1$ co-annihilation A/H funnel $\tilde{\chi}_1^\pm$ co-annihilation

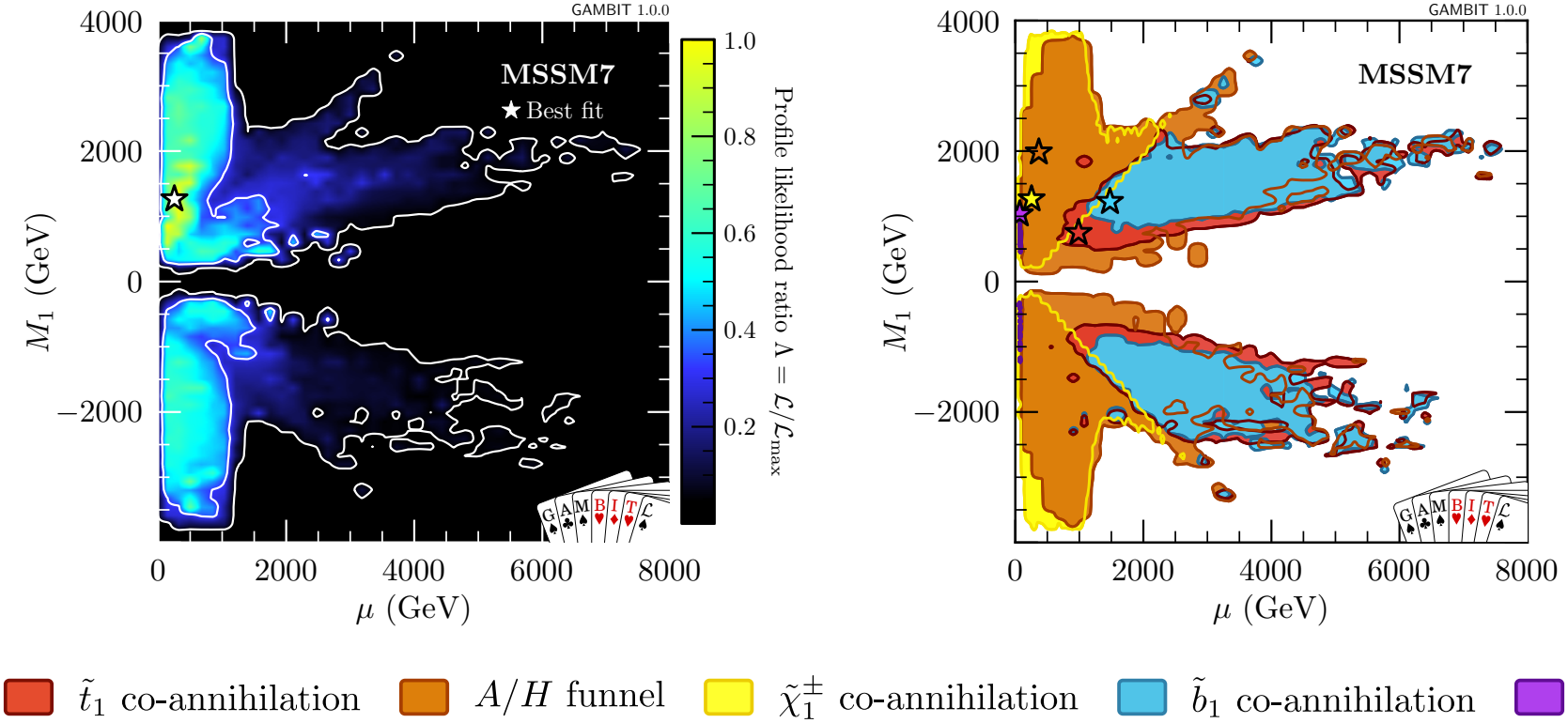
Some more CMSSM

(arXiv:1705.07935)



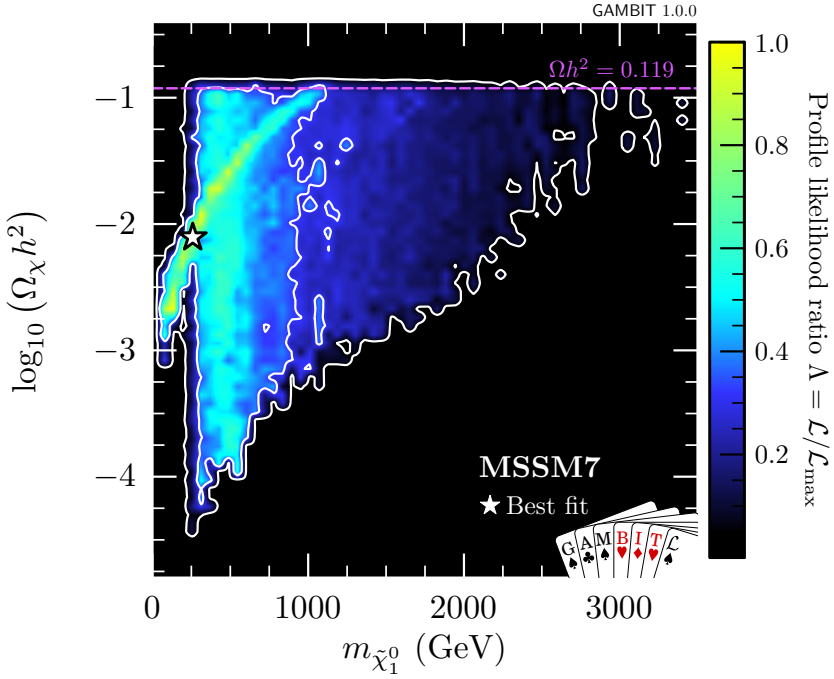
Some more MSSM-7

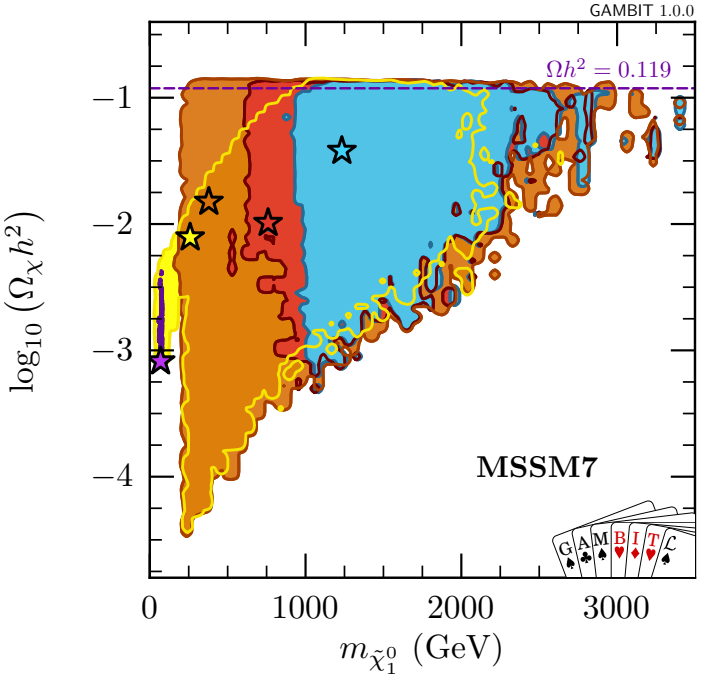
(arXiv:1705.07917)



Some more MSSM-7

(arXiv:1705.07917)

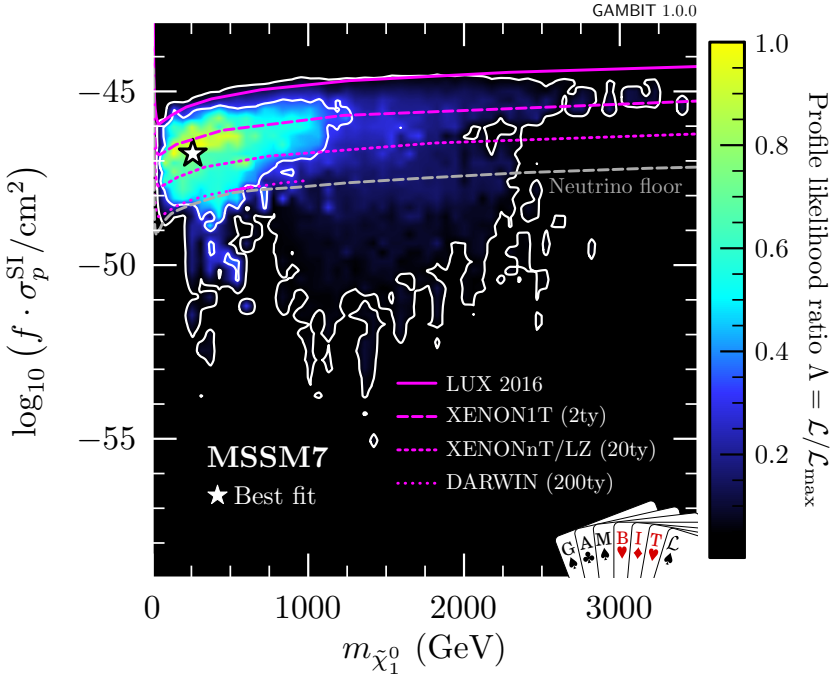


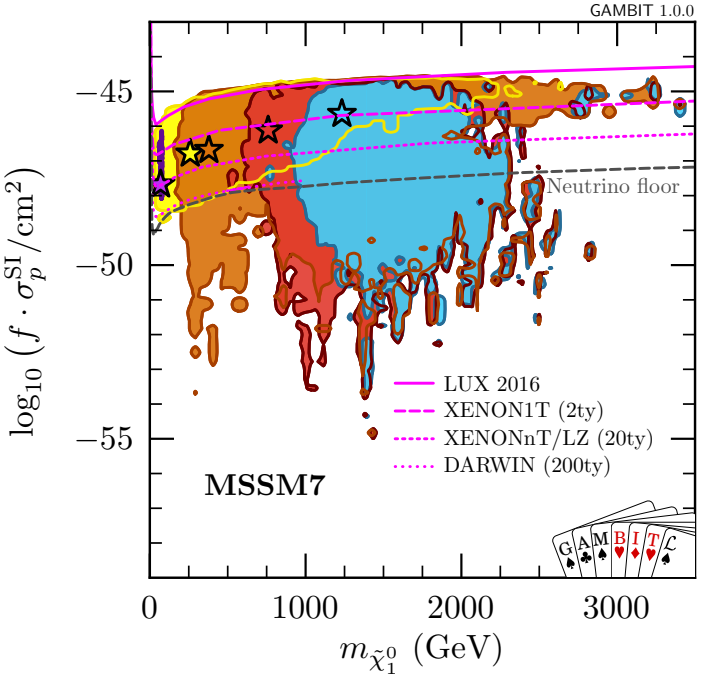


$\tilde{t}_1$ co-annihilation | A/H funnel | $\tilde{\chi}_1^\pm$ co-annihilation | $\tilde{b}_1$ co-annihilation | h/Z funnel

Some more MSSM-7

(arXiv:1705.07917)

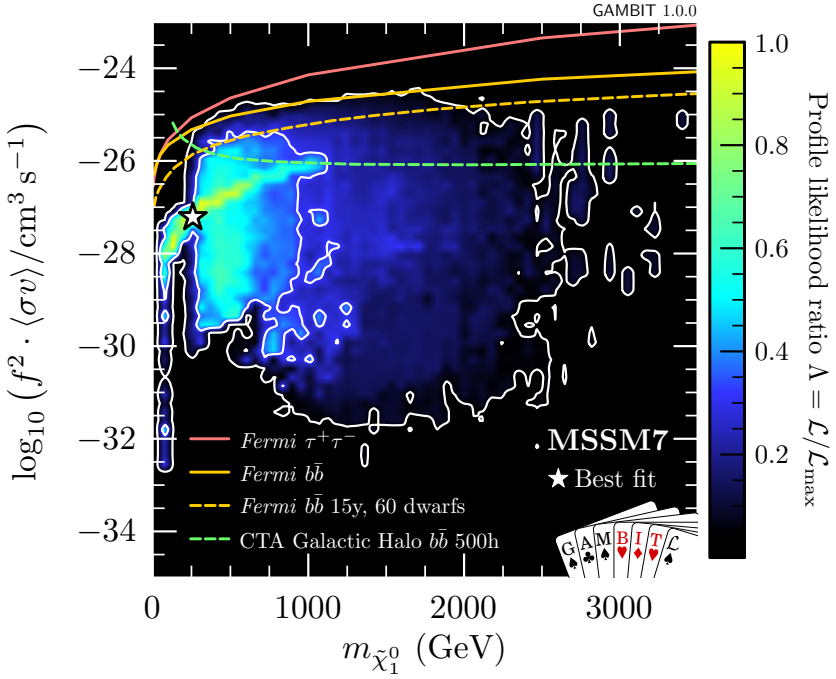


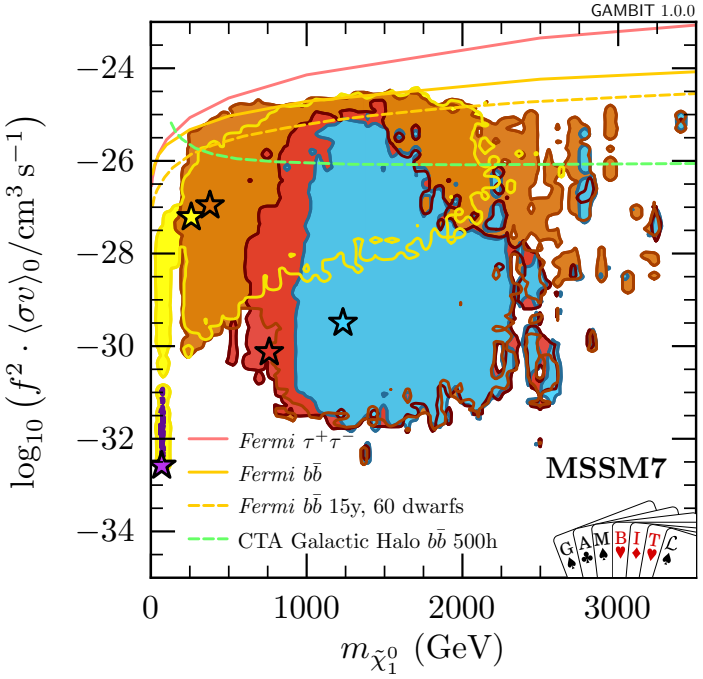


$\tilde{t}_1$ co-annihilation | A/H funnel | $\tilde{\chi}_1^\pm$ co-annihilation | $\tilde{b}_1$ co-annihilation | h/Z funnel

Some more MSSM-7

(arXiv:1705.07917)

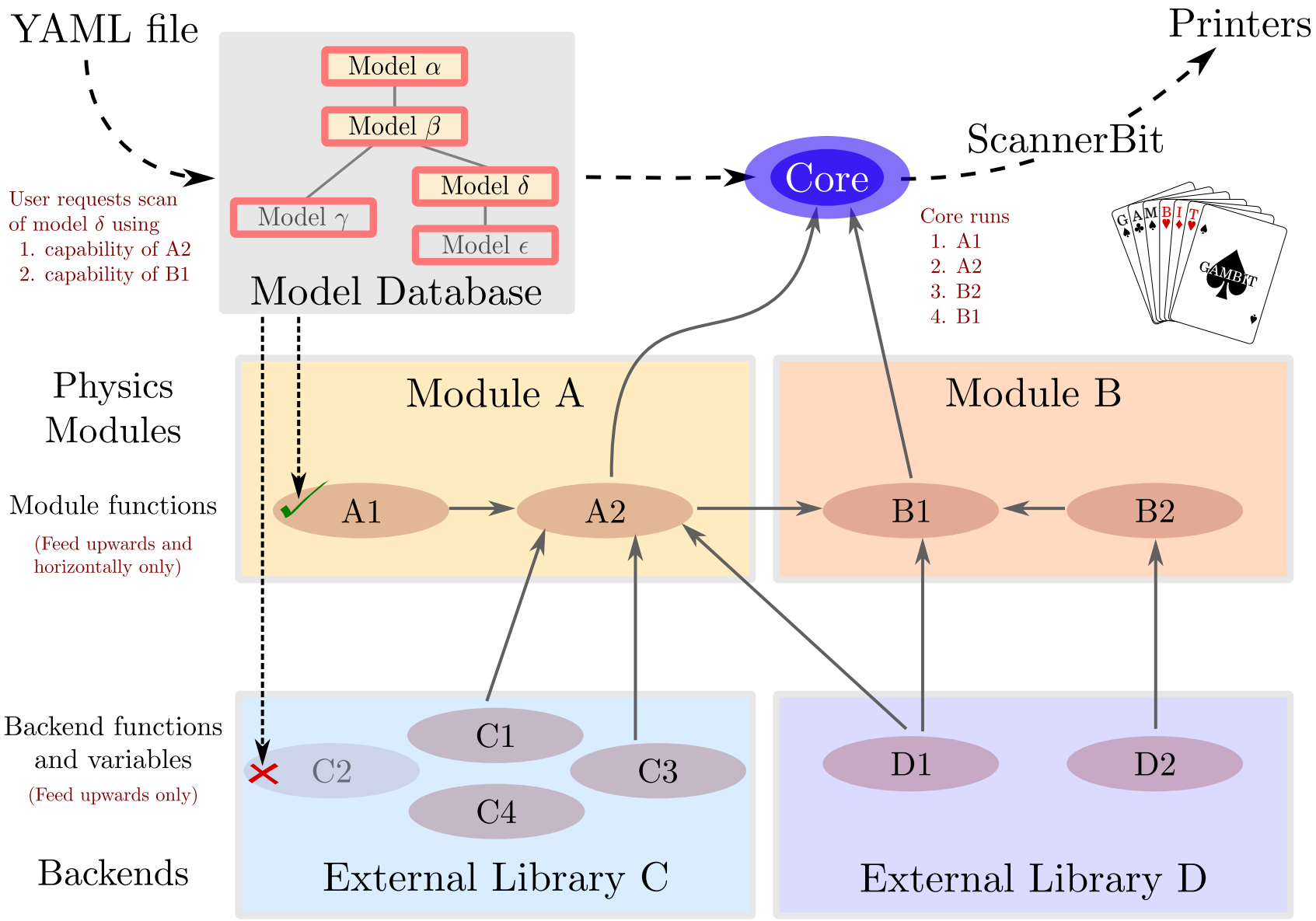




$\tilde{t}_1$ co-annihilation A/H funnel $\tilde{\chi}_1^\pm$ co-annihilation $\tilde{b}_1$ co-annihilation h/Z funnel

Functional overview of GAMBIT

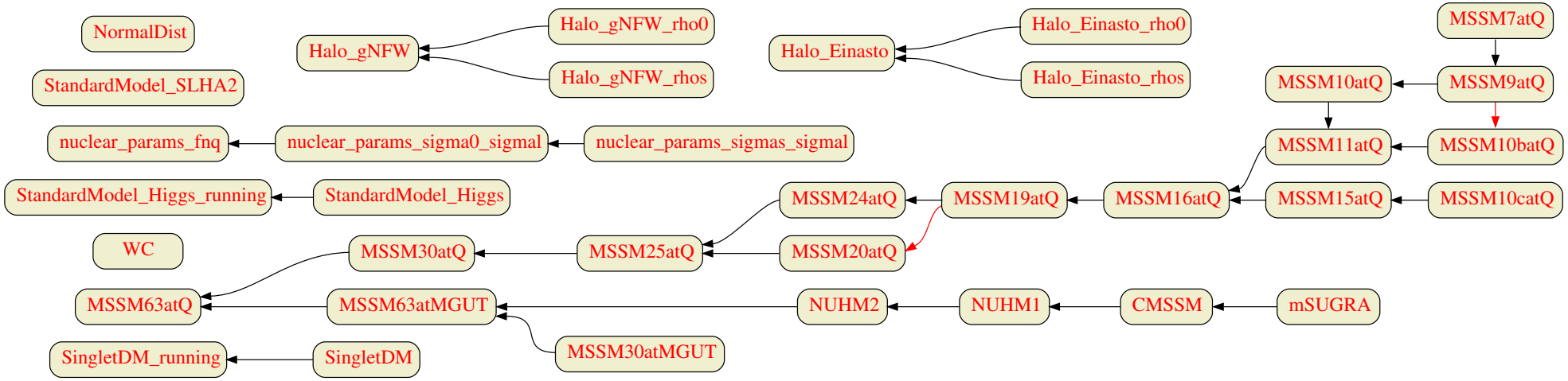
(arXiv:1705.07908)



Hierarchical Model Database (arXiv:1705.07908)

- Models are defined by their parameters and relations to each other
- Models can inherit from (be subspaces of) **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

(arXiv:1705.07908)

- 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

(arXiv:1705.07908)

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

(arXiv:1705.07908)

pat@xpspedition: ~/gambit 163x45

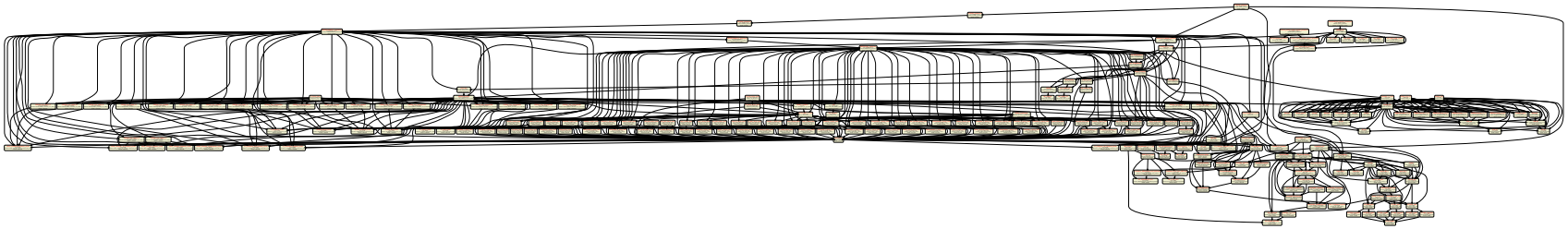
All relative paths are given with reference to /home/pat/gambit.

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)

Dependency Resolution

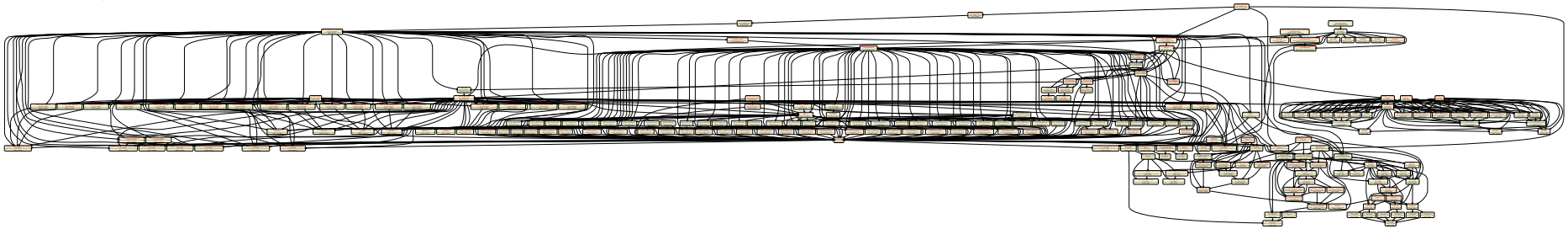
(arXiv:1705.07908)



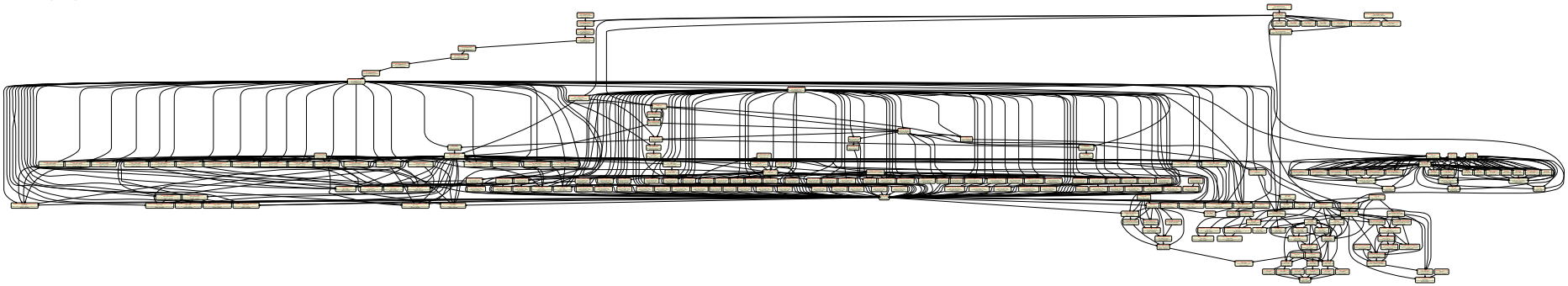
- Module functions and backend functions get arranged into a **dependency tree**
- Starting with requested observables and likelihoods, GAMBIT fills each dependency and backend requirement
- Obeys **rules** at each step: allowed models, allowed backends, constraints from input file, etc
- $\rightarrow$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

Dependency Resolution

CMSSM:

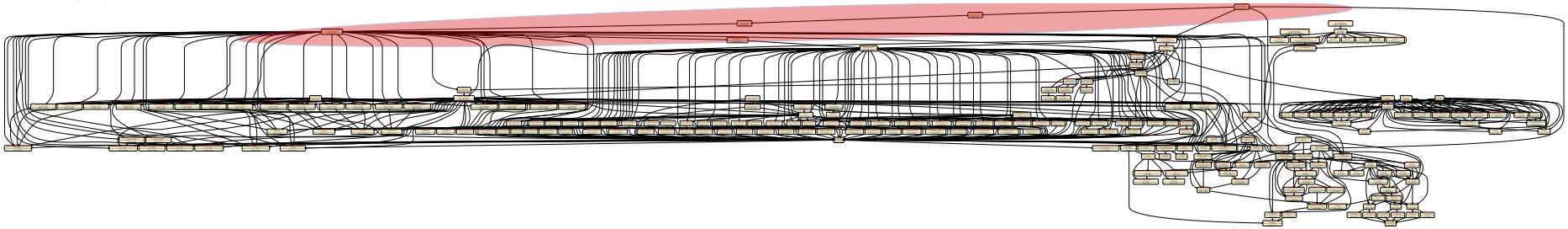


MSSM7:

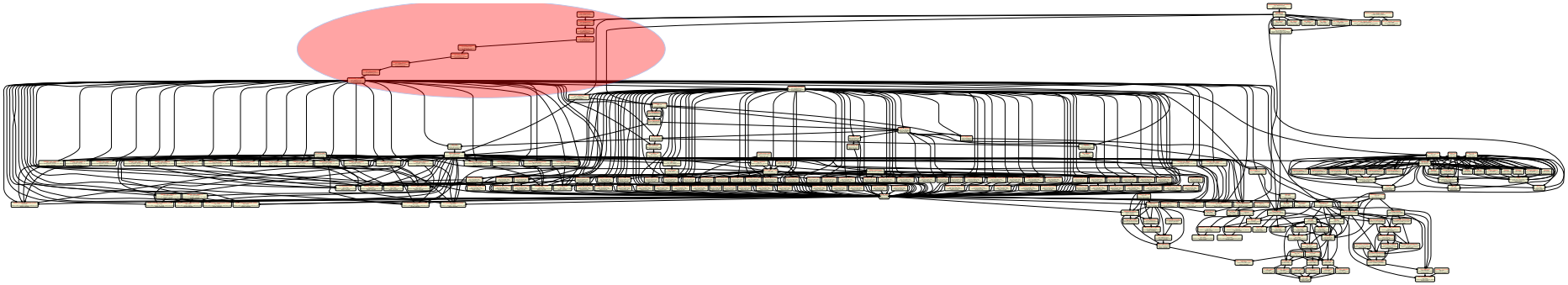


Dependency Resolution

CMSSM:



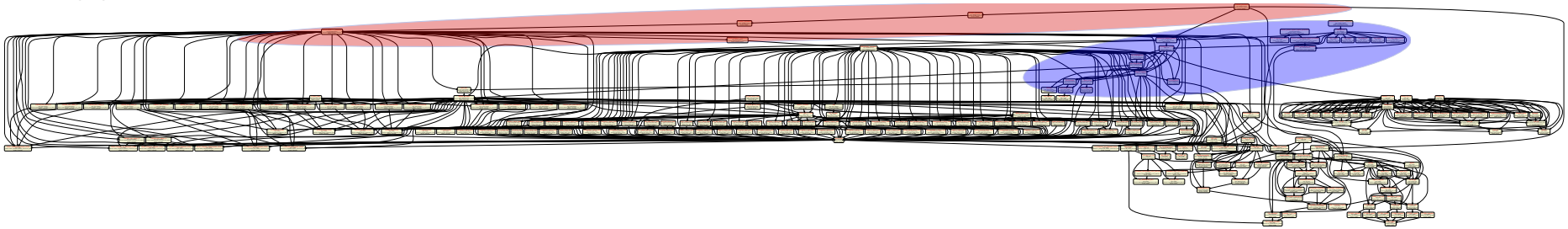
MSSM7:



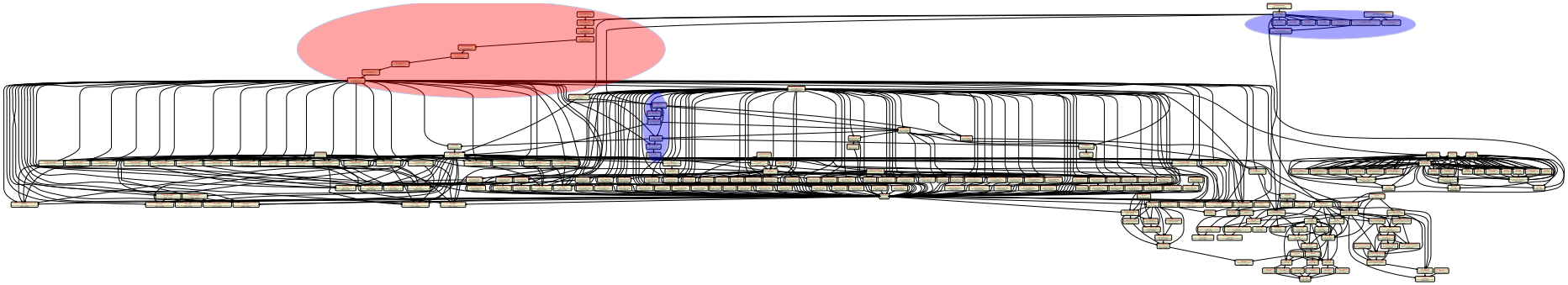
Red: Model parameter translations

Dependency Resolution

CMSSM:



MSSM7:

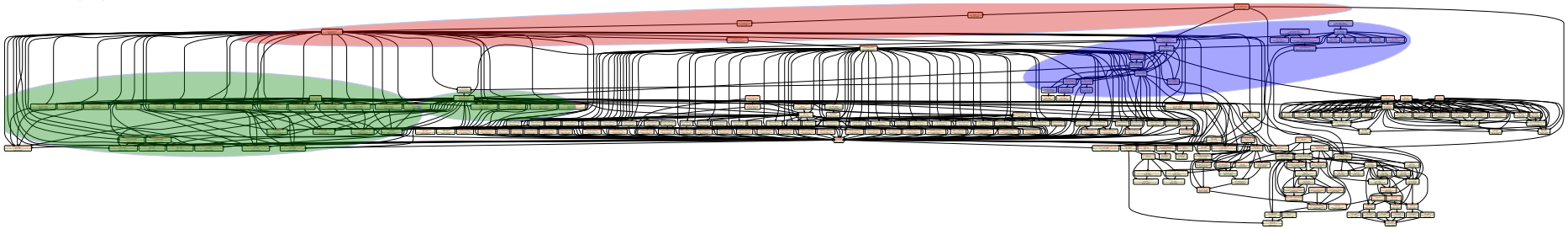


Red: Model parameter translations

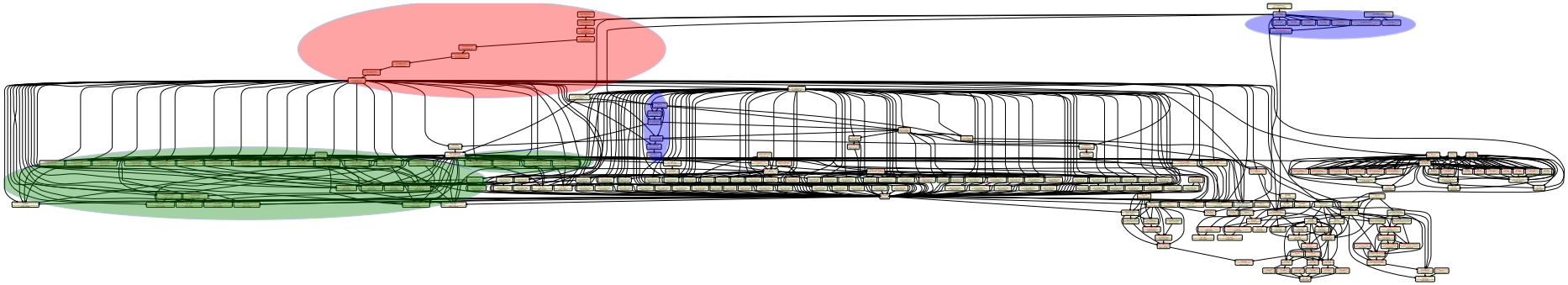
Blue: Precision calculations

Dependency Resolution

CMSSM:



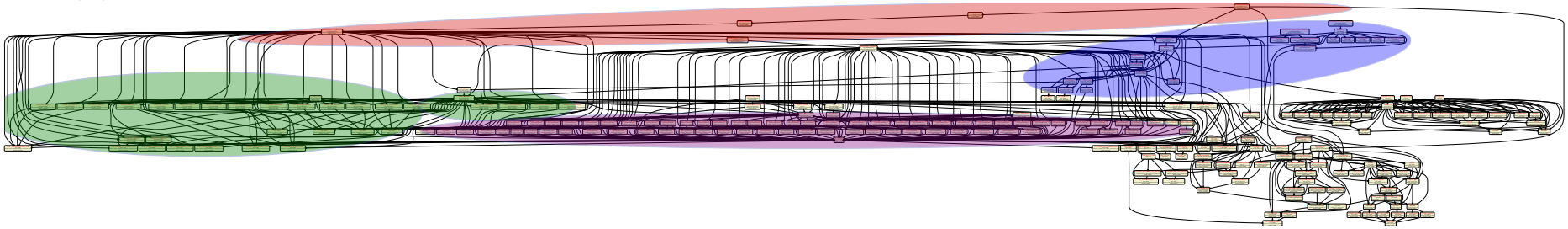
MSSM7:



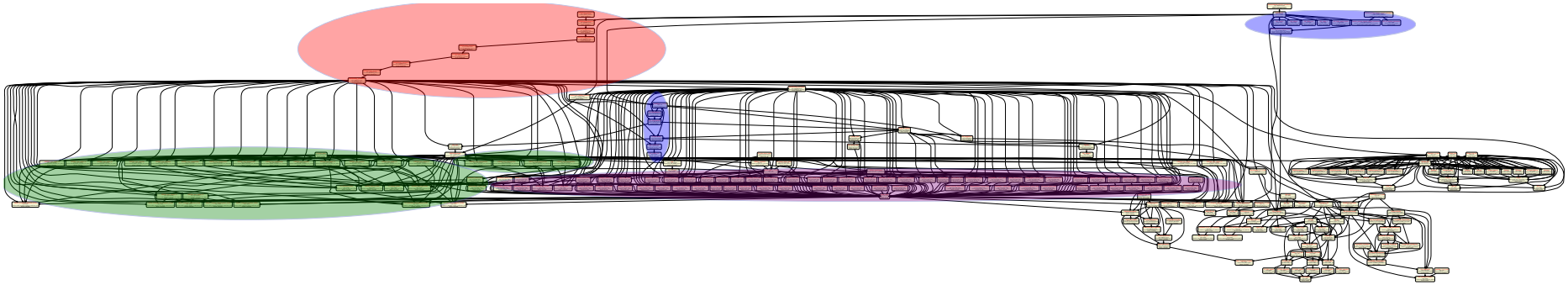
Red: Model parameter translations
Blue: Precision calculations
Green: LEP rates+likelihoods

Dependency Resolution

CMSSM:



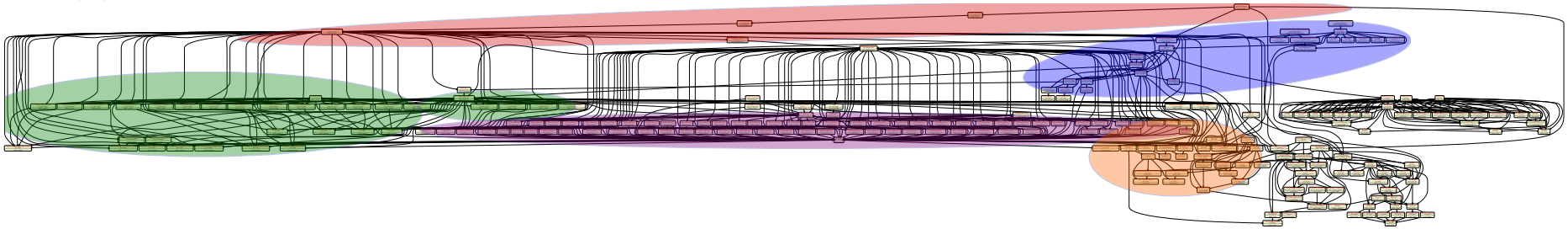
MSSM7:



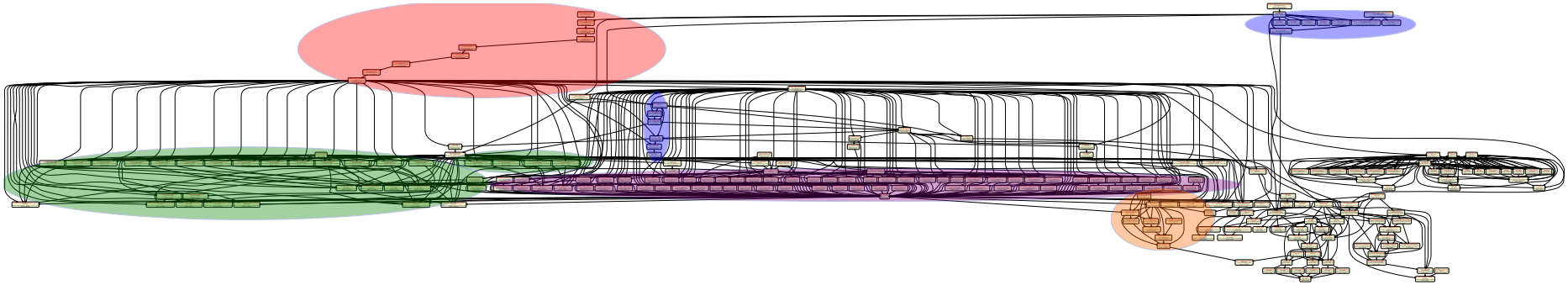
Red: Model parameter translations
Blue: Precision calculations
Green: LEP rates+likelihoods
Purple: Decays

Dependency Resolution

CMSSM:



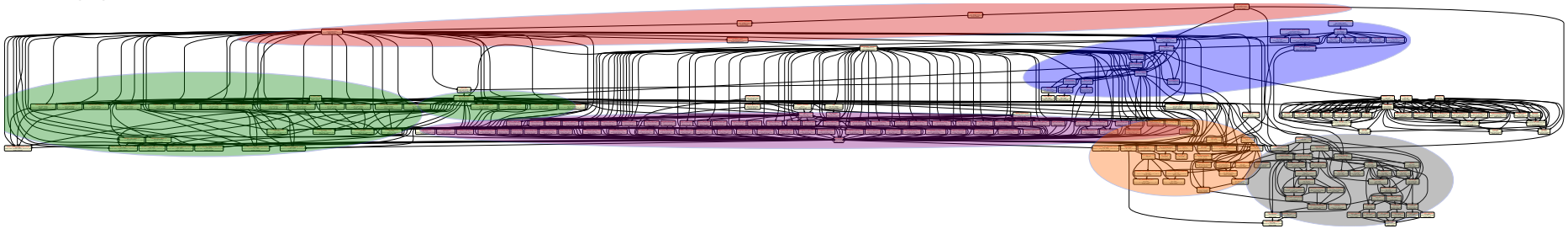
MSSM7:



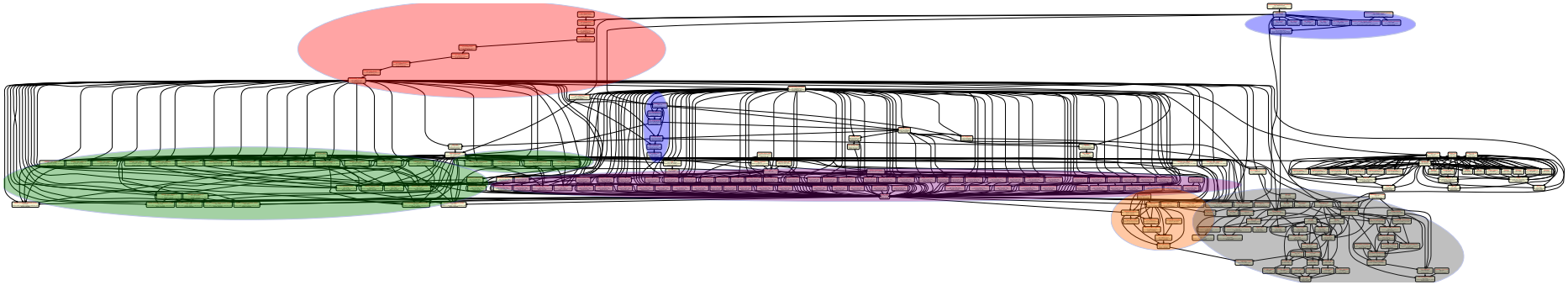
Red: Model parameter translations
Blue: Precision calculations
Green: LEP rates+likelihoods
Purple: Decays
Orange: LHC observables and likelihoods

Dependency Resolution

CMSSM:



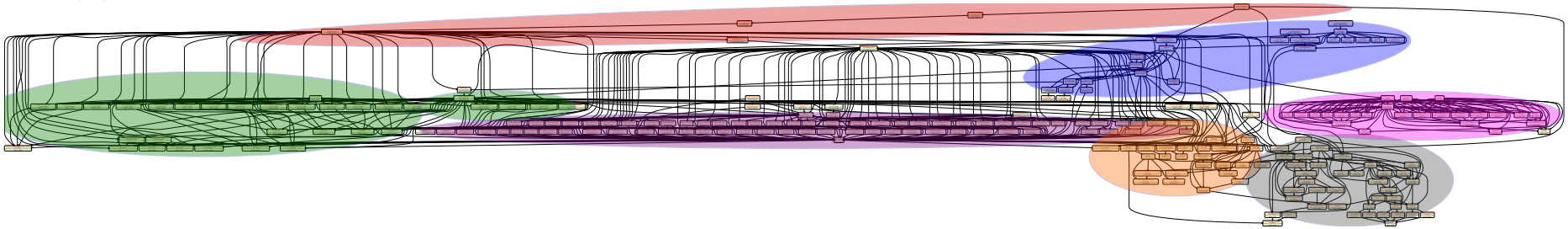
MSSM7:



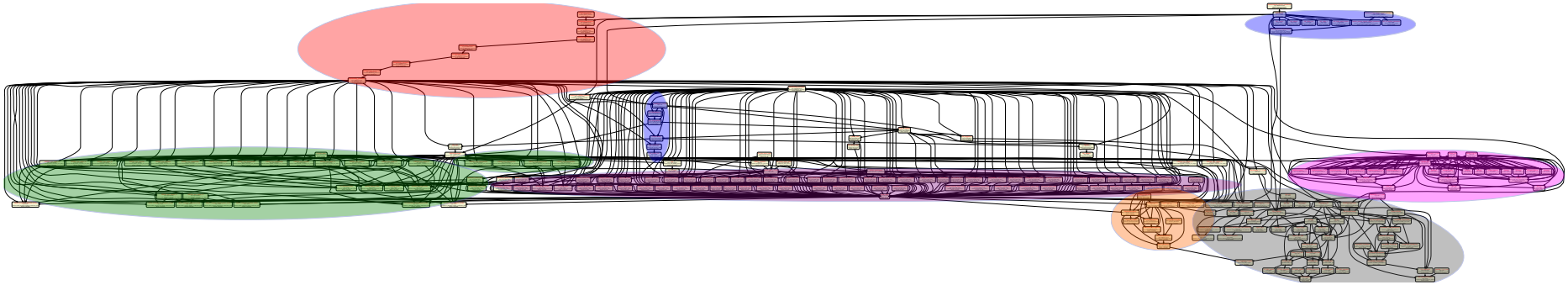
Red: Model parameter translations
Blue: Precision calculations
Green: LEP rates+likelihoods
Purple: Decays
Orange: LHC observables and likelihoods
Grey: DM direct, indirect and relic density

Dependency Resolution

CMSSM:



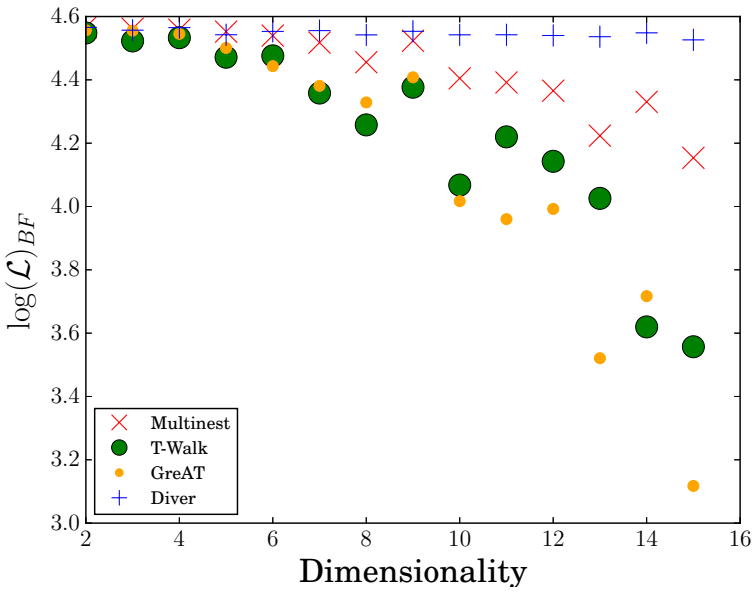
MSSM7:

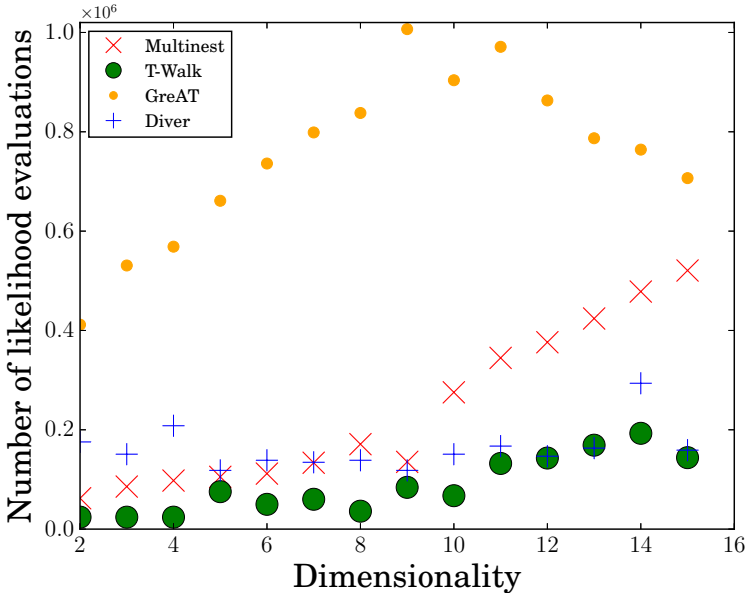


Red: Model parameter translations
Blue: Precision calculations
Green: LEP rates+likelihoods
Purple: Decays
Orange: LHC observables and likelihoods
Grey: DM direct, indirect and relic density
Pink: Flavour physics

Scanner performance comparison (arXiv:1705.07959)

Extensive scanner tests on scalar singlet model with different numbers of nuisance paramters





Diver scales far better with dimensionality than MultiNest or other scanners

Expansion: adding new observables and likelihoods

Adding a new module function is easy:

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

```
#define MODULE FlavBit                                  // A tasty GAMBIT module.
START_MODULE

  #define CAPABILITY Rmu                                // Observable: BR(K->mu nu)/BR(pi->mu nu)
  START_CAPABILITY
    #define FUNCTION SI_Rmu                             // Name of a function that can compute Rmu
    START_FUNCTION(double)                              // Function computes a double precision result
    BACKEND_REQ(Kmunu_pimunu, (my_tag), double, (const parameters*)) // Needs function from a backend
    BACKEND_OPTION( (SuperIso, 3.6), (my_tag) )                      // Backend must be SuperIso 3.6
    DEPENDENCY(SuperIso_modelinfo, parameters)          // Needs another function to calculate SuperIso info
    ALLOW_MODELS(MSSM63atQ, MSSM63atMGUT)               // Works with weak/GUT-scale MSSM and descendents
    #undef FUNCTION
  #undef CAPABILITY
```

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

Expansion: adding new models

1. Add the model to the **model hierarchy**:
   - Choose a model name, and declare any **parent model**
   - Declare the model's parameters
   - Declare any **translation function** to the parent model

```
#define MODEL NUHM1
#define PARENT NUHM2
  START_MODEL
  DEFINEPARS(M0,M12,mH,A0,TanBeta,SignMu)
  INTERPRET_AS_PARENT_FUNCTION(NUHM1_to_NUHM2)
#undef PARENT
#undef MODEL
```

2. Write the translation function as a standard C++ function:

```
void MODEL_NAMESPACE::NUHM1_to_NUHM2 (const ModelParameters &myP, ModelParameters &targetP)
{
   // Set M0, M12, A0, TanBeta and SignMu in the NUHM2 to the same values as in the NUHM1
   targetP.setValues(myP,false);
   // Set the values of mHu and mHd in the NUHM2 to the value of mH in the NUHM1
   targetP.setValue("mHu", myP["mH"]);
   targetP.setValue("mHd", myP["mH"]);
}
```

3. If needed, declare that existing module functions work with the new model, or add new functions that do.

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)

```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)

```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
```

Other nice technical features

- **Scanners**: Nested sampling, differential evolution, MCMC, t-walk. . .
- Mixed-mode **MPI + openMP** parallelisation, mostly automated → 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, . . .)
- available as docker plugin or vagrant virtual machine
- more more more. . .

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

ColliderBit details

LEP likelihoods

- complete model-independent recast of direct sparticle searches

Higgs likelihoods:

- for now: HiggSignals + HiggsBounds + constraints from invisible fits (Bernon, Dumont, Kraml et al)
- future: full simulation and ATLAS+CMS combination, more correlations, CP info, no SM-like coupling assumptions

Fast LHC likelihoods

- no simplified models, just faster direct simulation

ColliderBit details

LHC likelihoods:

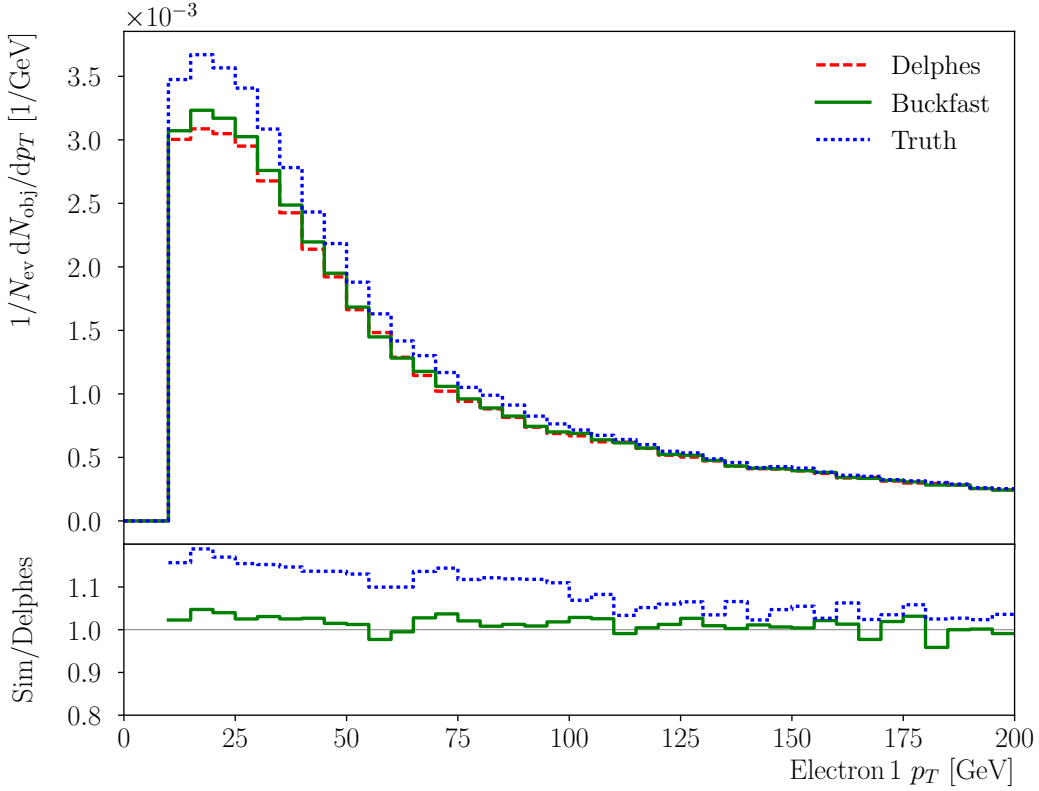
- MC generation: Pythia8 parallelised with OpenMP + other speed tweaks

ColliderBit details

LHC likelihoods:

- MC generation: Pythia8 parallelised with OpenMP + other speed tweaks
- Detector simulation: fast simulation based on 4-vector smearing
  → matches DELPHES results very closely (but much faster!)

Leading electron p_{T} distribution (CMSSM example):
red: detector-level simulation with DELPHES
green: 4-vector smearing with GAMBIT
blue: truth-level distribution



ColliderBit details

LHC likelihoods:

- MC generation: Pythia8 parallelised with OpenMP + other speed tweaks
- Detector simulation: fast simulation based on 4-vector smearing
  $\rightarrow$ matches DELPHES results very closely (but much faster!)
- Cross-sections: LO + LL from MC generator by default (fast NLO in works for SUSY)

ColliderBit details

LHC likelihoods:

- MC generation: Pythia8 parallelised with OpenMP + other speed tweaks
- Detector simulation: fast simulation based on 4-vector smearing
  $\rightarrow$ matches DELPHES results very closely (but much faster!)
- Cross-sections: LO + LL from MC generator by default (fast NLO in works for SUSY)
- Analysis framework: custom event-level, independent of experiment or simulation

ColliderBit details

LHC likelihoods:

- MC generation: Pythia8 parallelised with OpenMP + other speed tweaks
- Detector simulation: fast simulation based on 4-vector smearing
  $\rightarrow$ matches DELPHES results very closely (but much faster!)
- Cross-sections: LO + LL from MC generator by default (fast NLO in works for SUSY)
- Analysis framework: custom event-level, independent of experiment or simulation
- Likelihood: inline systematic error marginalisation (via `nulike`)

ColliderBit details

LHC likelihoods:

- MC generation: Pythia8 parallelised with OpenMP + other speed tweaks
- Detector simulation: fast simulation based on 4-vector smearing
  $\rightarrow$ matches DELPHES results very closely (but much faster!)
- Cross-sections: LO + LL from MC generator by default (fast NLO in works for SUSY)
- Analysis framework: custom event-level, independent of experiment or simulation
- Likelihood: inline systematic error marginalisation (via `nulike`)
- Initially shipping with:
  - ATLAS SUSY searches (0ℓ, $0/1/2\ell$ $\tilde{t}$, b jets + MET, $2/3\ell$ EW)
  - CMS multi-ℓ SUSY
  - CMS DM (t pair + MET, mono-b, monojet)
  - ATLAS + CMS Run II 0ℓ
  - refresh coming with more Run II analyses shortly