Core

done-ish undone

Core/Model Responsibilities

- guts of GAMBIT
- observables and likelihoods modular "rollcall"
- system allowing them to be hooked up easily and arbitrarily "rollcall" + "dependency resolver"
- System allowing arbitrary external codes "backends"
- system for reading/interpreting initialization file "ini file parser"
- Framework for defining models and their relationships "Model Bit"
- System of passing values (parameters, obs, likes) between scanner and obs/like calculators "Likelihood Container"

Boring Responsibilities

- error/exception managing/logging
- other logging
- documentation - Doxygen
- compilation/configuration system

  autotools? cmake? qmake?

  user does not need to install extras

  - rather choose the more <u>extensible</u> option.

- Naming conventions for capabilities?

  ex xsection is too generic. many things are ~~x~~sections.

- Reference or value?

  byVal necessary?

- Timing, Probability determination for the order Likelihoods are calculated.

Model Bit

- Heirarchical structure
- Built from module machinery

Components of a model

- Name
- Parameters object (wrapped in functor)
- Lineage list: isdesendentof(<model>)
- "Interpret as parent" (functor)
    - with dependencies

model_base - base class for inheritance chain.

Parameters mapped to capabilities...
⇒ default behavior?

replace models::<model_1>::isdescendentof(<model_2>)

with a global function?

isdescendentof(<model_1>, <model_2>)

Unfinished:

- backends
- model dependencies
- translation between parametrizations?
- conditional dependencies?
- multiple parents
- orthogonal model sectors
- Priors

Scanner Bit

not-started

To do (or done):

- MasterLike container object
  - holds references for necessary objects
- Templated (maybe) wrapper functor for scanner to use w/ specializations
- Input ini-file setup
- dynamic loading – maybe automatic once dlopen issues are sorted out.
- mask for output?

MasterLike contains

- Model params
- functor list
- inputMap and outputList "Graphs"

HECollider

Status

- AldoDet / Fast Sim
  - Summer student validating
  - development as well
- Pythia ⇒ Delphes ⇒ Analysis Chain
  - set up, but disagrees with "standalone"
  - debug this week
- Standard Event format
  - Analysis can use "generator" or "detector reconstructed" events
- Backending stalled
  - waiting on dlopen for classes....

Tasks

- 7 TeV correct

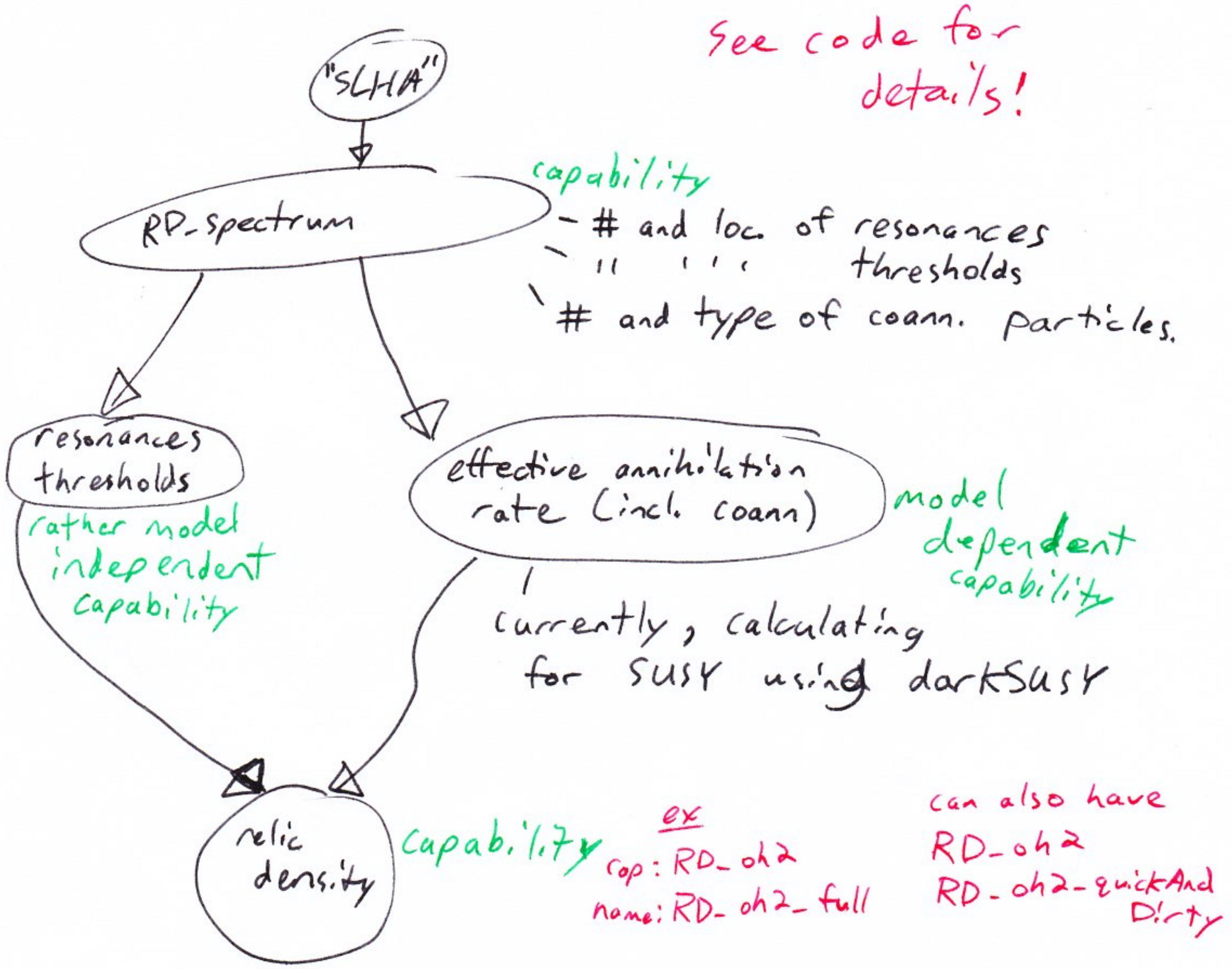
  8 TeV wrong?

  ⇒ jets & piletup?

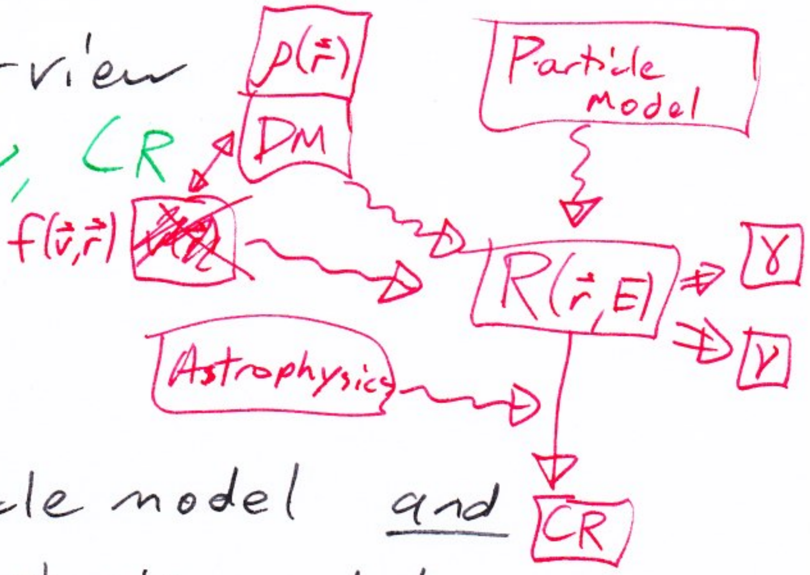
- Higgs?
- Flavor Physics inventory?

Dark Bit

- Relic Density calculation

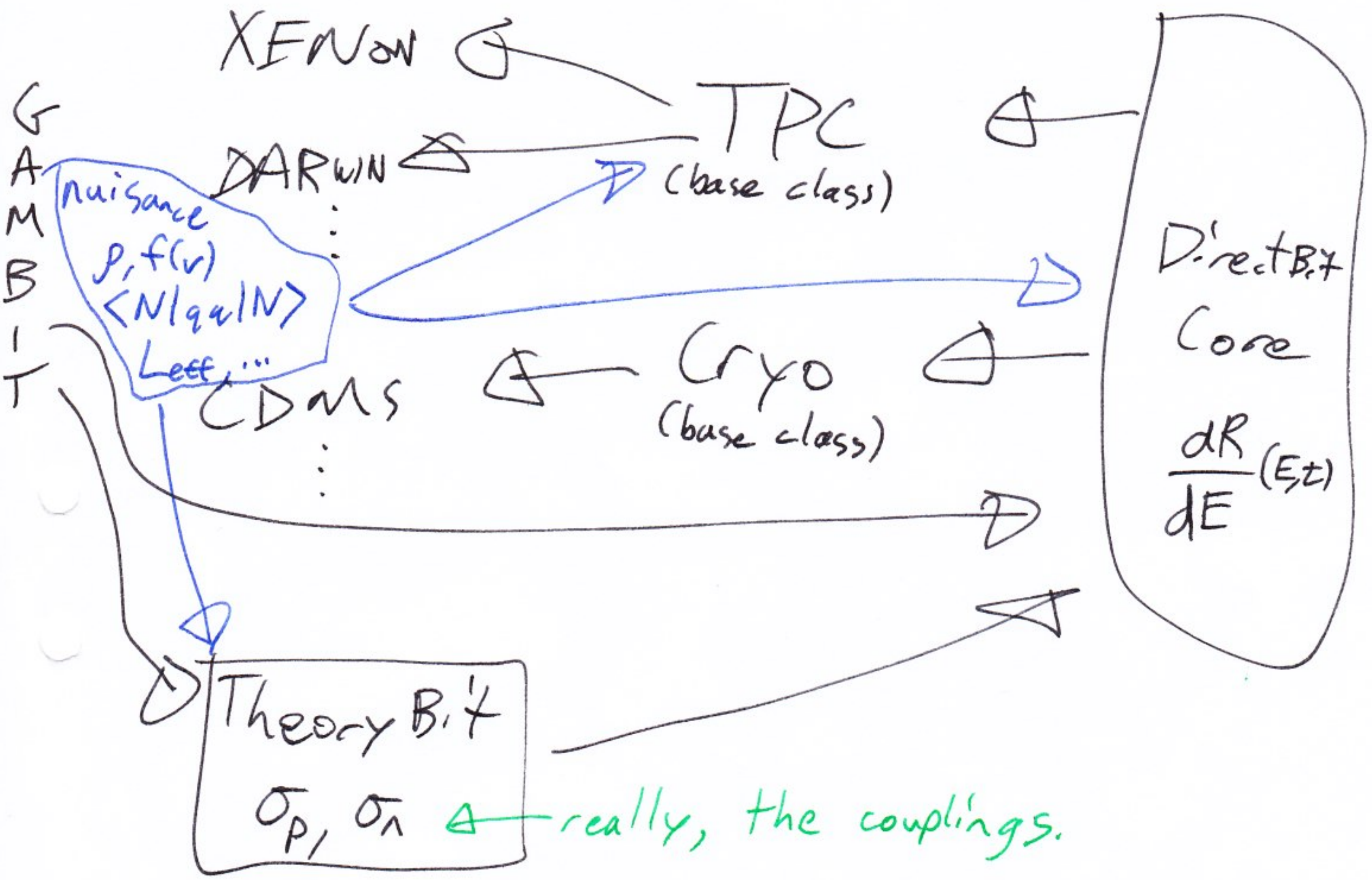

See code for details!
"SLHA"
RD_spectrum
capability
- # and loc. of resonances
- " " " thresholds
- # and type of coann. particles.
resonances thresholds
rather model independent capability
effective annihilation rate (incl. coann)
model dependent capability
currently, calculating for SUSY using darkSUSY
relic density
capability
ex
cap: RD_oh2
name: RD_oh2_full
can also have
RD_oh2
RD_oh2_quickAndDirty

- Dark Bit overview
  - Indirect γ, ν, CR
  - Direct
  - Relic Density



- Indirect - particle model <u>and</u> Astrophysics model
  - γ, ν may deal only with ~~the~~ the injection rate function.

Direct Bit



Interface layer needed?

⇒ rollcall header?

Still work in progress...

Code Discussions

- Group Backend requirements into "Blocks"
- Conditional Backend requirement on models.

Communication btw code parts

~~Utils~~

Module specific:
- put it in ModuleBit

Useful for many modules:
- put it in Utils.

ex: types go in ~~gambitCommonTypes.hpp~~ CommonReturnTypes.hpp

Ini File - discussed already

- support both yaml and table-style formats.
- Yaml easy for devel...?
- ini file parser delay?
  - Multiple ini files for multiple modules?
    Configuration files

Misc

- Avoid SLHA file I/O... Can we have a struct SLHA and make minor mods to external codes?
  (still need to code a parser anyways)
  (still need SLHA struct as well)
- Lower case w/ underscore for file names.

- Each module can have its own ExampleModule.cpp

- Put an overall TODO file in the source code? Useful for development.
- Shift from GET_DEP to safe pohters

Exception Handling

- maybe some observable crashes... if it's not too important, still continue the point ⇒ need to catch different errors?
  - still messes up scanner ⇒ can't trust overall likelihood.
- definitely want custom string messages for errors.
- good + bad points in separate files.

Config file stuff

- track config files for each module/backend function "HAS_CONFIG_FILE"

Modules as standalones

- Neat little tool to hook up backends?
- IN_CORE flag?
- Mini-core for Module standalone release