The QuickAna Tool Scheduler

Nils Krumnack (Iowa State University)
Steve Farrell (LBL)
Xiaowen Lei (Arizona)

Introduction

- my basic idea is to have a utility (QuickAna) that provides similar functionality to TopRootCore and SUSYTools in run 1:
  - i.e. apply all the standard CP tools for the user
  - provide collections of fully corrected good objects to the user
  - provide an overall event weight to the user
  - evaluate all systematics in a standardized manner

- would like to improve on TopRootCore/SUSYTools:
  - should be a generic tool that can be used in any framework,
    - including athena and the derivation framework
  - should rely on the xAOD EDM, instead of implementing its own
  - should allow physics groups to add custom prescriptions easily
  - should implement harmonization TF prescriptions

- general hope is to consolidate on a single framework

General Reception

- presented at analysis release meeting
  - generally positive reception
- harmonization task force seemed interested
  - i.e. they'd like to have a piece of software that they can endorse as the "standard" prescription for event correction
- ASG is generally opposed to any such idea
  - they also never liked TopRootCore, so this is consistent
  - don't think they get on board with any tool with such goals
- talked to Top and SUSY about joining the effort:
  - Top interested and we will be working together
  - SUSY interested, but declined due to manpower concerns

- now looking into whether TopRootCore can meet our needs
  - refactoring it may be easier than starting from scratch
  - bonus: TRC is already known and has a user base
- at a later point, will once more ask SUSY group to join

Basic Interface for Users

- design assumes the new/casual user sees just a single tool
  - provides a "facade" that hides all the implementation details
  - internally redirects to various modules/packages

- imagine users start by calling a correctEvent() method:
  - applies all CP tools
  - initializes tool scheduler to this event

- the user could then call a variety of functions e.g.:
  - isGoodEvent(), eventWeight(), etc.
  - jets(), muons(), etc.

- for systematics users reconfigure QuickAna with applySystematics()
  - also can query which systematics affect which step
  - similar interface to the CP tools

Tool Configuration

- users only configure QuickAna, not individual CP tools
  - reduces chance for mistakes or mismatches between users
  - makes configuration more stable against version changes
- configuration options should roughly correspond to what you would put in a physics presentation/supporting note

- configuration will be fairly basic, i.e. users can ask for "loose" electrons, "tight" muons, "anti-kt 0.4" jets, etc. by name
  - could also have something like "Top tight" electrons if the Top group has group specific selection/correction prescriptions
- possibly other object type specific options?
  - e.g. JES decomposition choice for jets?

- advanced configuration would require replacing internal modules
  - seems preferable to "overly" flexible configuration
  - most users shouldn't need that

Auto-Documentation

- would like to make it easy for users to document what they did
  - ideally all configuration options should be physics relevant and documented
- could just copy configuration file into your .ppt
  - likely to be a wee bit ugly
  - also: users may ask for e.g. the "default" jet definition, and the documentation should include what that means
- instead could provide a makeSlide() method that provides a pre-formatted slide
- also could provide a printLog() method that prints the complete information to the log-file

- caveat: this will break down, if
  - the user doesn't specify the release he uses
  - or does some customizations that are not properly marked

Hypothetical Usage

- // do a normal dual-use tool initialization
- ana::EventTool *tool = new ana::EventTool ("myana");
- tool->setProperty ("jets", "anti-kt 0.4");
- tool->setProperty ("electrons", "tight");
- tool->initialize ();

- // do per-event processing
- for (auto sys = tool->sysBegin(), …) {
- tool->applySystematics (*sys);
- tool->correntEvent (event);
- …
- hist[*sys]->Fill (someVariable, tool->eventWeight());
- }

Persistification

- QuickAna itself is not meant to do any persistification
  - however, it puts all data into the xAOD EDM
  - a separate tool could write the relevant data out
  - will shallow-copy all collections for each systematic

- due to nature of shallow copy, collections would contain both accepted and rejected objects
- to distinguish, add special marker-fields:
  - ana_accept: passes good object definition
  - ana_select: passes user selection
- probably more, if we add more processing steps
- persistification can in principle thin collections before writing…

- implication: user needs to be able to break corrections into steps
  - i.e. need multiple versions of each correctEvent(), etc.
    or pass the level of correction needed into methods

User Object Selection

- some tools need to know user object selection, e.g.:
  - jet selection for better JES estimation
  - jet flavor for more precise systematics
  - generally for overlap removal
  - incorporating the object SF into event weight
- mostly optional, i.e. new users can skip this
  - mainly improves the systematics

- would also allow basic kinematic selection, e.g "pt>100e3"
  - using same formula mechanism as for the derivation framework
- advanced users would break up processing:
  - run pre-corrections
  - apply user object selection
  - run post-corrections
  - do event analysis

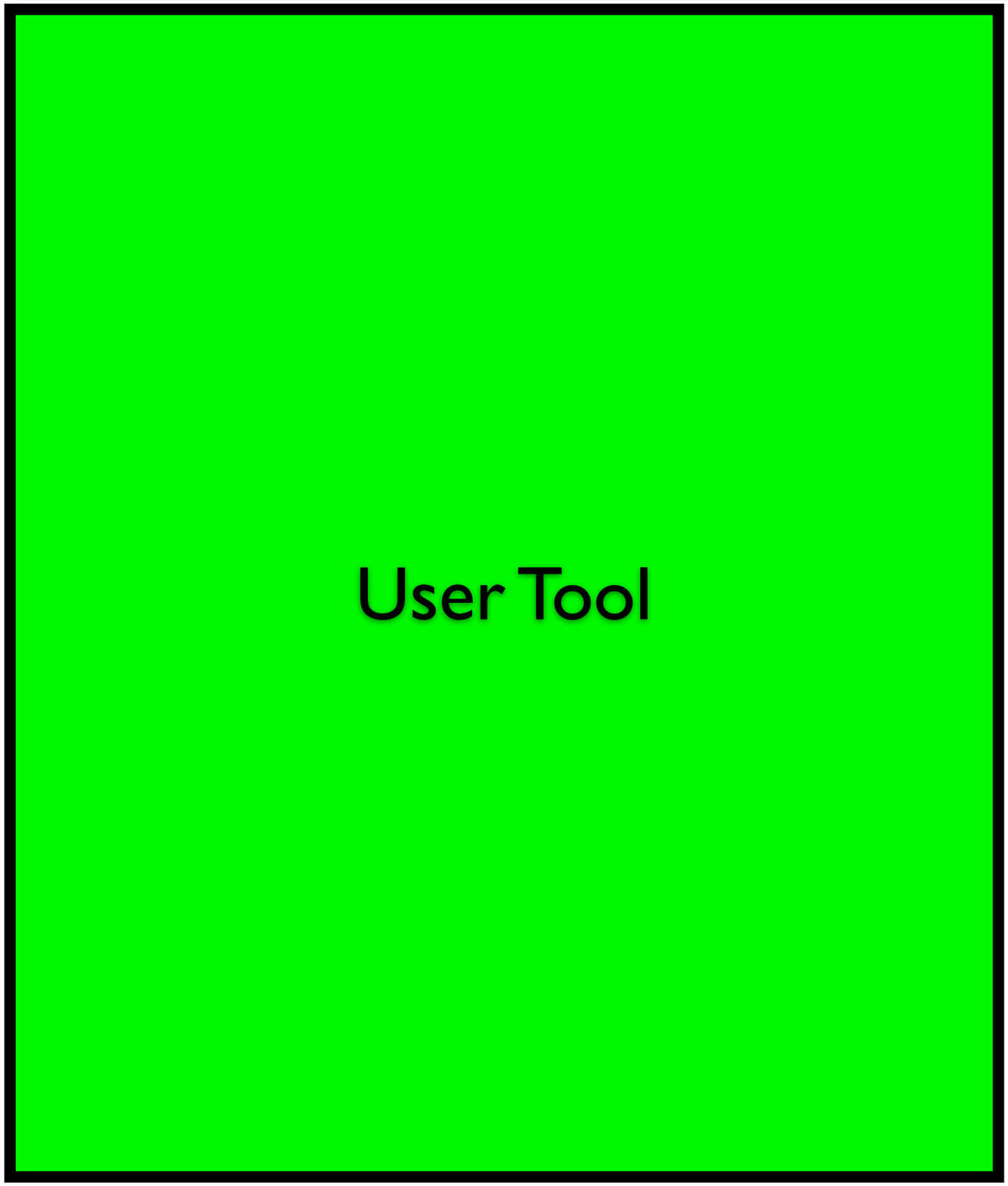
Systematics

- in the simplest form follow general interface for CP tools:
  - i.e. have applySystematics(), affectingSystematics() and recommendedSystematics()
  - also, need method to go from recommendedSystematics() to std::vector<SystematicSet>

- when working in multiple processing steps need to break this up:
  - i.e. ignore all systematics not affecting this job
- need at least affectingSystematics() separate per processing step
  - and a method to prune an std::vector<SystematicSet> for each processing step

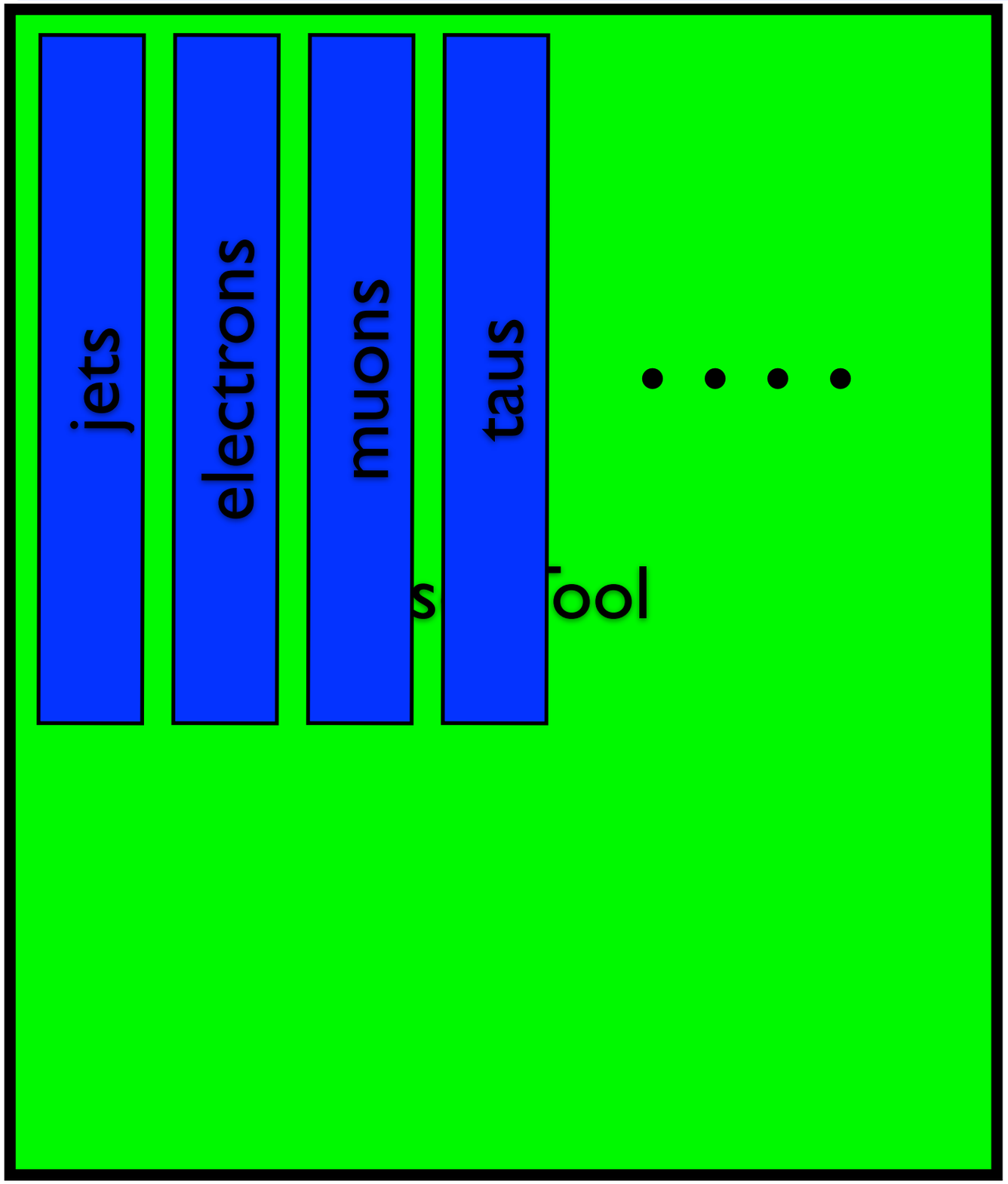
- users may also want to split systematics by object type
  - e.g. don't vary electrons for jet systematics
  - can save space in early processing stages
  - however: overlap removal, etc. do tie everything together

Implementation Details



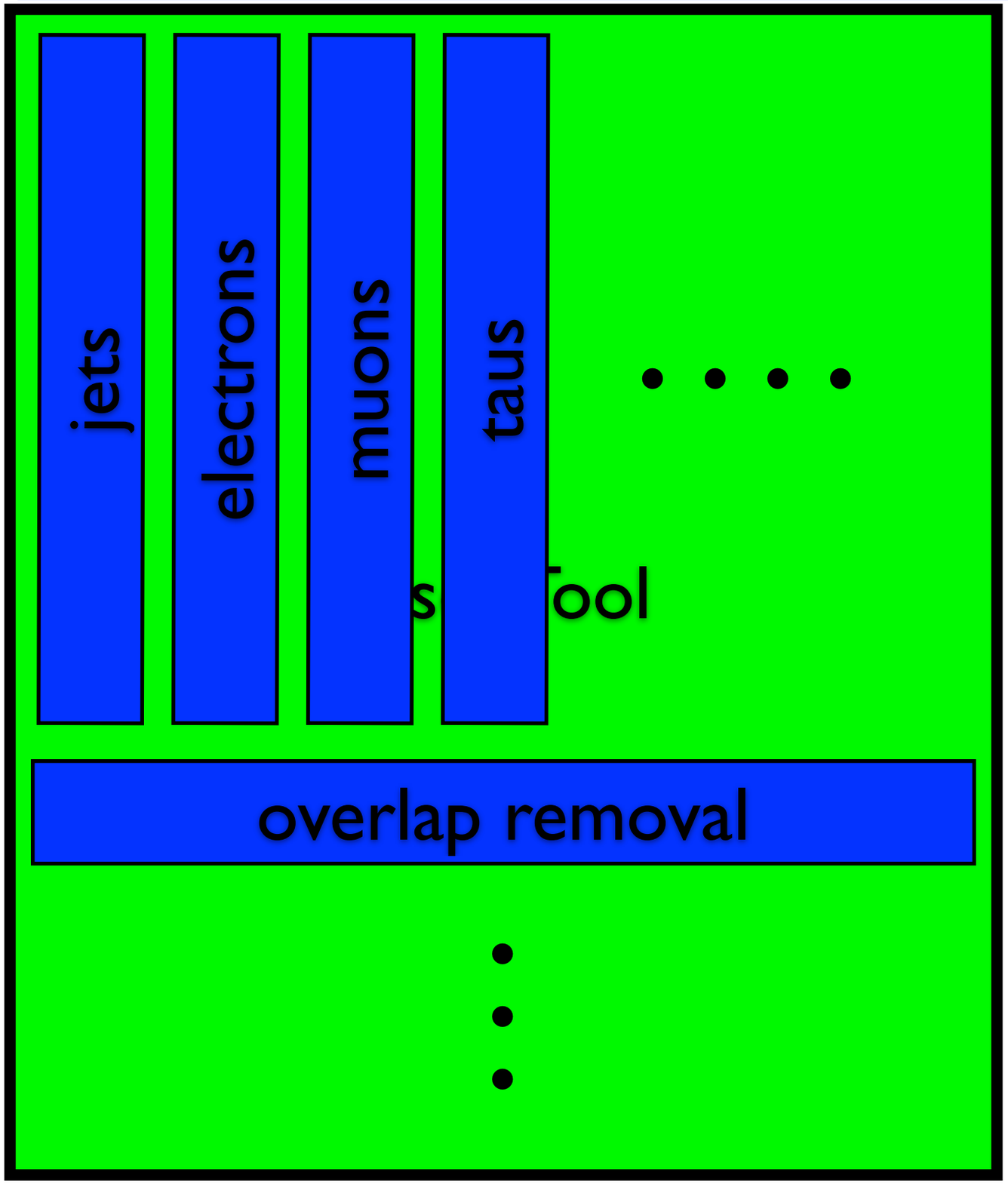
- users create a single tool
  - and that's all they ever see

Implementation Details



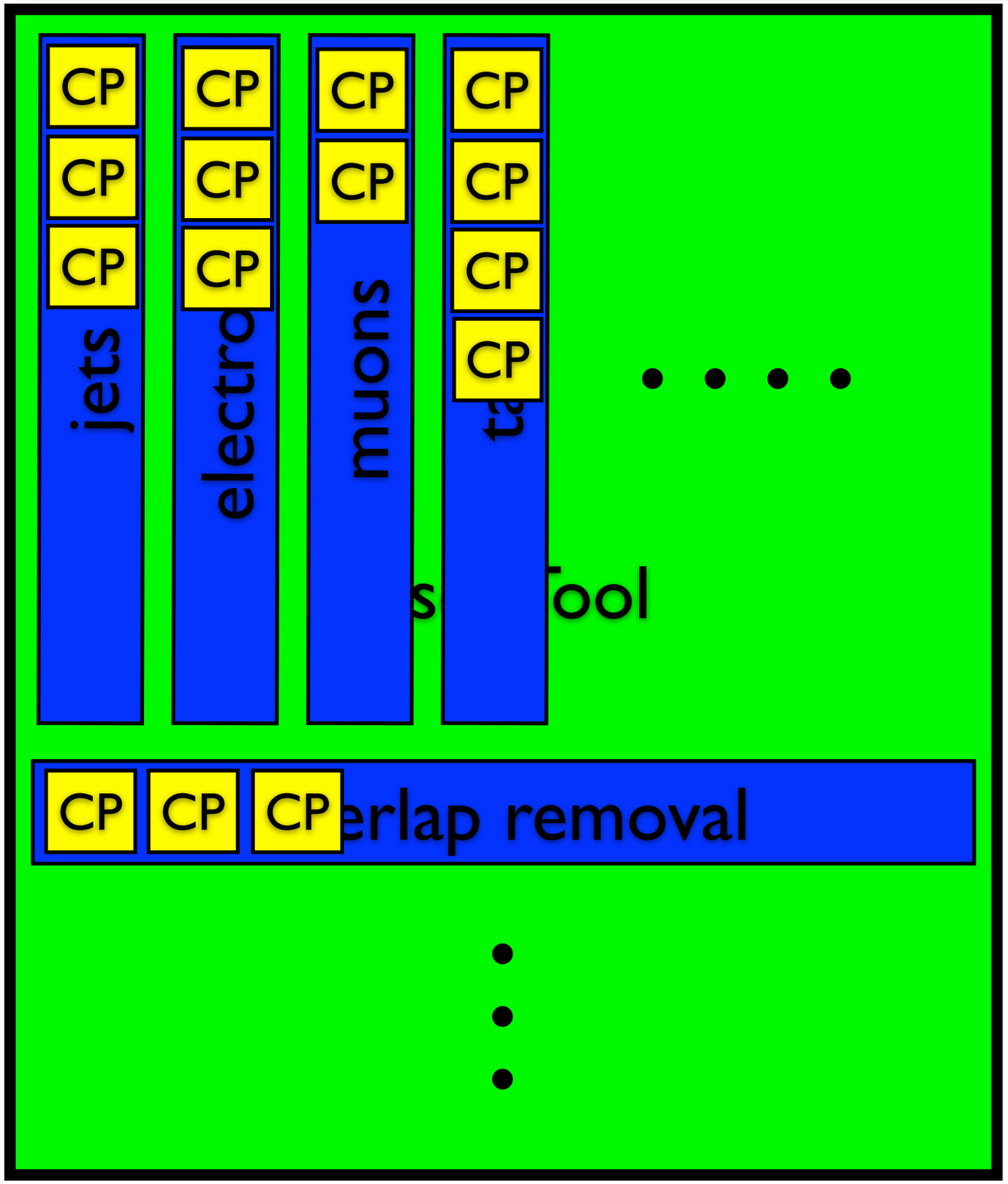
- users create a single tool
  - and that's all they ever see
- internally it creates one tool for each object type
  - knows which collection to use and how to correct it

Implementation Details



- users create a single tool
  - and that's all they ever see
- internally it creates one tool for each object type
  - knows which collection to use and how to correct it
- also creates some global tools internally
  - i.e. tools with no specific collections associated

Implementation Details



- users create a single tool
  - and that's all they ever see
- internally it creates one tool for each object type
  - knows which collection to use and how to correct it
- also creates some global tools internally
  - i.e. tools with no specific collections associated
- the internal tools are responsible for creating and using the CP tools

Internal Tool Design

- user tool creates internal tools based on its configuration
  - after that central tool no longer cares about configuration
  - each configuration option should belong to one internal tool
- in the ideal case the name specifies the object collection to use, plus the exact configuration of CP tools to use
  - will also need special names for fake estimates, etc.

- the internal tools should be simple, minimal and focused:
  - should be understandable by the "inexperienced" user
  - should only contain "physics" code, no data handling, etc.
  - tool should be solely responsible for a single object type
    - i.e. looking at one file could tell me everything about jets, etc.
  - could use these independently from QuickAna as well

- framework takes care of data handling, systematics management…
  - probably break it up into multiple internal modules

Summary & Outlook

- trying to provide a common tool scheduler for run 2
  - aims to provide corrected objects for a "normal" analysis
  - tries to sacrifice flexibility in favor of simplicity
  - should be framework independent
  - components usable without the framework

- still in the early phases of the project
  - have a good starting point with TopRootCore
  - still need to study it in detail to see what changes it needs
- hope to get other groups to join in
  - still hopeful for SUSY group
  - harmonization TF seemed interested
  - potentially absorb some smaller frameworks
- overall timescale unclear, but likely tight