GlueX Reconstruction Proposal

Paul Mattione, Carnegie Mellon University

Top-Level Objects/Factories

- Multiple ways to provide users with physics information:
  - **DAnalysisIndependentResults**: The lowest level at which reconstruction results are available: DTrackTimeBased, DNeutralShowerCandidates, DBeamPhotons
  - **DSingleRFSingleVertexResults**: Assume all tracks & showers came from a common RF beam bunch and vertex
  - **DMultiRFSingleVertexResults**: Allow matching of tracks & showers to different RF beam bunches, but assume a single vertex per RF bunch
  - **DMultiRFMultiVertexResults**: Allow matching of tracks & showers to different RF beam bunches, and allow multiple vertices per RF bunch

Details

- Comments/Uses:
  - **DAnalysisIndependentResults**: No assumptions about physics event, user can reconstruct anything they like from this information.
  - **DSingleRFSingleVertexResults**: Single RF/vertex assumptions mean high-level data (DVertex, DChargedTrack, DNeutralTrack) are in simplest form. Useful for quick, 1st-order analyses.
  - **DMultiRFSingleVertexResults**: More sophisticated than single-RF results, but one DVertex per beam bunch yields more complicated DNeutralTracks (shower/vertex matching).
  - **DMultiRFMultiVertexResults**: Again more sophisticated, but again more complicated DNeutralTrack results. Also, vertex grouping algorithm may not match reaction topology.

Implementation

* We can choose to provide users with some or all of these methods
  * These methods provide users with increasingly accurate/sophisticated results, but with the cost of increased assumptions/complications.

* Proposal:
  * During cooking, create and save the **DAnalysisIndependentResults** and **DSingleRFSingleVertexResults** objects
    * Provides all information needed for user to build high-level objects however they want.
    * Provides quick-and-dirty, default high-level results for users to perform 1^{st}-order analyses.
  * For analyses, provide factories for the **DMultiRFSingleVertexResults** and **DMultiRFMultiVertexResults** objects
    * Users can quickly build these objects with the given assumptions from the current data at analysis time.
  * Users can also implement their own factories to reconstruct objects (e.g. a DVertex factory that performs a kinematic fit to the proton & π^- tracks)

Major Changes to Current Code

* Create a DRFBunch factory to match tracks to RF bunches
* Have DChargedTrackHypothesis & DNeutralTrackHypothesis inherit from DKinematicData
* Have factory tags for DVertex and DRFBunch:
  * The default factory assumes only one of each
  * The factories with the "MULTIRF"/"MULTIVERTEX" tags attempt to split up the tracks into multiple vertices/bunches as needed
* **Would other factories (e.g. DChargedTrack) then need "MULTI" tags in order to use the correct "MULTIRF"/etc. objects???**
* Create the top-level object factories: they will reconstruct the objects with/without the "MULTI" tags

High-Level Object/Factory Details

DRFBunch

* Match tracks & photons to RF beam bunches: one created for each group
* Members:
  * Time-matched DTrackTimeBased objects
  * Time-matched DBeamPhoton objects
  * RF Vertex (Single fit vertex for all tracks matching this RF bunch)
    * If a track has > 1 DTrackTimeBased, use the DTrackTimeBased with the largest tracking FOM
    * Use current vertex finder algorithm
  * RF Time (at POCA to RF vertex)
* Factories: Default (only one DRFBunch), "MULTIRF" (can be > 1, detailed next slide)

DRFBunch – "MULTIRF"

* For each DTrackTimeBased:
  * Project the SC hit-time & RF times to the beamline (POCA)
    * If no SC hit, use DC time
  * Select the RF beam bunch that is closest to the projected SC time
* Select the RF beam bunches that match the tagged photons
* Group tracks to RF bunches:
  * Of the photon-matched RF bunches, select the one with the most matched tracks (matched track: at least one matching DTrackTimeBased)
  * Group these matched tracks together with that RF beam bunch
  * Repeat grouping until no remaining tracks or no remaining photon-matched RF bunches
  * If tracks still remain, select the RF beam bunch with the most remaining matched tracks and group the tracks to it (repeat until no tracks remaining)
    * This occurs if the true photon wasn't tagged

DChargedTrackHypothesis

* Inherits from DKinematicData
* Construct one for each possible PID (+: Proton, π^+, K^+; -: π^-, K^-)
  * This is independent of which mass hypotheses were used during track reconstruction
* Other members:
  * Particle ID
  * χ^2's and NDF from dE/dx (DC, TOF, BCAL?), timing
  * dMatchedTimeDetector (SYS_TOF, etc.)
  * Associated objects: DTrackTimeBased, DRFBunch

Building DChargedTrackHypothesis

- For each DTrackTimeBased:
  - Position-match the TOF/BCAL/FCAL hits
- Create a DChargedTrackHypothesis for each possible PID of each track
  - If no DTrackTimeBased for specific mass, copy tracking results from existing fit (for kaons, choose pion fits)
  - Calculate dE/dx χ^2's (DC, TOF, BCAL?)
  - Project the matching TOF/BCAL/FCAL time to the corresponding DRFBunchTrack vertex, calculate timing χ^2
  - Calculate the overall PID FOM by weighting the χ^2 from the various sources

Charged Particle TOF: Misc.

* If no BCAL/FCAL/TOF hit, use DC time instead
* If track hits in multiple systems: use BCAL, then TOF, then FCAL hits
  * TOF: best time resolution, but BCAL: hits sooner, significant energy loss
* If > 1 BCAL/FCAL/TOF position-matched hit in the chosen system
  * If none are > 3 σ apart in time, hit-time = weighted average
  * If at least one is > 3 σ apart in time, use reconstructed DC time instead

DChargedTrack

- Members:
  - Vector of DChargedTrackHypothesis objects corresponding to the same track
  - Associated objects: DRFBunch
- Methods:
  - charge() : Returns track charge
  - Pion() : Returns either the π^+ or π^- DChargedTrackHypothesis
  - Kaon() : Returns either the K^+ or K^- DChargedTrackHypothesis
  - Proton() : Returns either the proton DChargedTrackHypothesis or NULL
  - BestFOM() : Returns the DChargedTrackHypothesis with the best FOM

DNeutralShowerCandidate

* Implementation unchanged
* Members: shower position, energy, time, and uncertainties
* Associated objects: DFCALShower/DBCALShower
* Create a DNeutralShowerCandidate for each DFCALShower and DBCALShower that is not unambiguously matched to a DChargedTrack
  * e.g. if a neutral shower matches the proton DTrackTimeBased fit for a track but not the pion fit, then it is still a DNeutralShowerCandidate
    * This is especially the case for low-momentum tracks where the track-fitting fails.
    * **IS THIS NECESSARY???**

DVertex

- Members:
  - Vertex position
  - Time (calculated using track times from PIDs with best FOM)
  - Vector of DChargedTrack objects
  - Associated objects: DRFBunch
- Creation (multiple factories):
  - DVertex_factory (Default):
    - Creates a single DVertex for each DRFBunch
    - Copies the vertex position from the DRFBunch
  - DVertex_factory_multiple (tag = “MULTIVERTEX”)
    - Try to reconstruct multiple vertices either via the current algorithm or by grouping via track DOCAs
  - Other
    - User-defined factories that could be used to create vertices for their own analysis (e.g. try to pair specific PIDs with a kinematic fitter, etc.)

DNeutralTrackHypothesis

- Inherits from DKinematicData
- For each DNeutralShowerCandidate, create a hypothesis for each ID (γ , n) & DVertex combination
  - Vertices: if no charged tracks, then use target center
- Other members:
  - Particle ID
  - χ^2's and NDF from timing
  - dMatchedTimeDetector (SYS_BCAL, SYS_FCAL)
  - Associated objects: DVertex, DRFBunch, DNeutralShowerCandidate, DBCALShower/DFCALShower
- Project BCAL/FCAL shower time to vertex, compare to vertex time: calculate χ^2

DNeutralTrack

* Members:
  * Vector of DNeutralTrackHypothesis objects corresponding to the same track
  * Associated objects: DNeutralShowerCandidate, DBCALShower/ DFCALShower
* Methods:
  * Photon() : Returns the photon DNeutralTrackHypothesis with the highest FOM (may be more than one DVertex)
  * Neutron() : Returns the neutron DNeutralTrackHypothesis with the highest FOM (may be more than one DVertex)
  * BestFOM() : Returns the DNeutralTrackHypothesis with the best FOM

DParticleSet

* Implementation virtually unchanged
* One per DVertex
* Members:
  * DVertex
  * Vectors of DNeutralTrack objects & DChargedTrack objects corresponding to this DVertex
  * Associated objects: DRFBunch

Top-Level Object/Factory Details

DAnalysisIndependentResults

* Members:
  * Vector of DNeutralShowerCandidate objects
  * Vector of DTrackTimeBased objects
  * Vector of DBeamPhoton objects
* These are the lowest-level reconstruction objects that a user may want to use as a starting point to perform his/her analysis.
* All objects derived from these include assumptions or algorithm choices for vertexing, matching to RF bunches, PID, etc. that a user may wish to change for his/her analysis.

DSingleRFSingleVertexResults

* Members:
  * Vector of DParticleSet objects
* The default factories for the DRFBunch and DVertex objects are used

DMultiRFSingleVertexResults

* Members:
  * Vector of DParticleSet objects
* The default DVertex factory is used, but the “MULTIRF” DRFBunch factory is used.

DMultiRFMultipleVertexResults

* Members:
  * Vector of DParticleSet objects
* The “MULTIRF” and “MULTIVERTEX” tags are used for the DRFBunch and DVertex factories, respectively.