Status of CAFs for ND-GAr

ND-GAr Weekly Meeting
08.11.2022

Francisco Martínez López
f.martinezlopez@qmul.ac.uk

University of Southampton
Queen Mary University of London
Particle Physics STFC Rutherford Appleton Laboratory

What are CAFs?

- **C**ommon **A**nalysis **F**iles are the high level representation of art (FD/ND-GAr) or edep-sim (other ND) outputs useful for analysis.
- CAFAna is a framework to:
  - Read those files and fill histograms in an efficient/ergonomic way.
  - Oscillation and cross section analysis tools build on top of that.
- They can also be read by other fitters like MaCh3.
- CAFs were used for the FD TDR, PRISM analysis and also by other experiments.

ND_CAFMaker

- ND_CAFMaker takes as input GENIE and reconstructed objects from the DUNE ND and combines them into CAFs.
- Truth information is filled into the CAF from the GENIE event record.
- Reco information is added by the different “reco branch filler” modules, which makes switching to different chains of reco objects very flexible (ND-LAr, ND-LAr+TMS, ND-LAr+ND-GAr, SAND, …).
- FHICL-based configuration system ensures that configurations are easily versioned and modified.

ND_CAFMaker

- The usage is simple, the only executable, makeCAF, is controlled by an input fhicl file.
  - Can specify things like the input GENIE file, the input reco files or the output file.
  - You can also override some of the fhicl inputs.

```
Usage: ./makeCAF [options] <driver.fcl> [options]

General options:
  -h [ --help ]           print this help message

FCL overrides (for quick tests; edit your .fcl for regular usage):
  -g [ --ghep ] arg       input GENIE .ghep file
  -o [ --out ] arg        output CAF file
  --startevt arg          event number to start at
  -n [ --numevts ] arg    total number of events to process (-1 means 'all')
```

ND-GAr MC Production

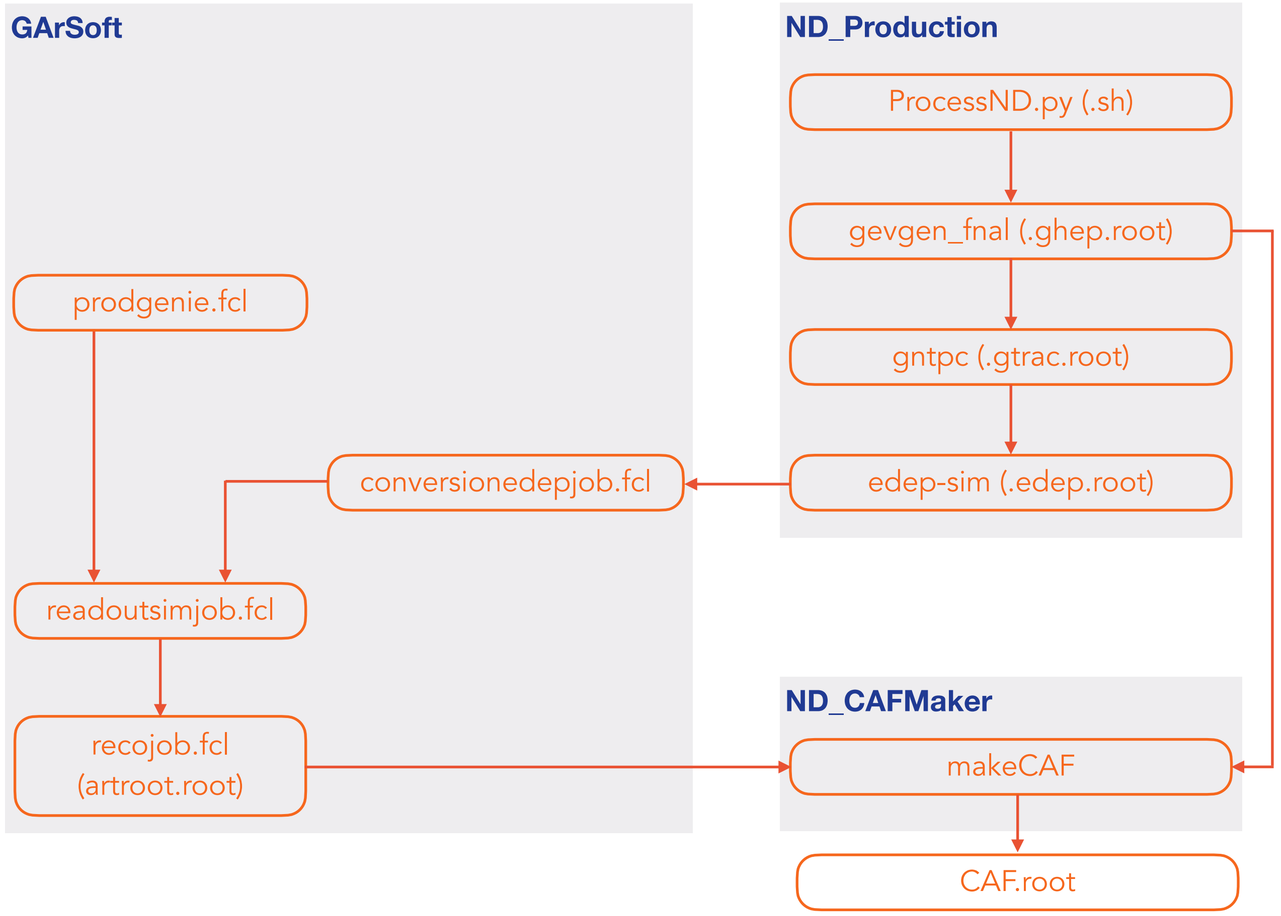
Thanks to Alex Booth!

- Ongoing work on integrating the ND-GAr toolchain with the sim-reco tools.
- The main ND_Production relies on a Python snippet which generates the job submission script. For a given geometry, this script:
  - Runs GENIE via gevgen_fnal rather than via art.
  - Then runs edep-sim (GEANT4) on the GENIE output.
- There exists an art module which takes the GENIE and edep-sim outputs to create an art file that can then be used in GArSoft.
- In parallel, progress is being made to move the ND production to POMS.

ND-GAr MC Production

Thanks to Alex Booth!

- Discussion about the future of the ND production is ongoing:
  - It could be that the main ND switches to an art-based framework.
- Current integration effort focuses on a gevgen/edep-sim type of chain, but this may well change.



Proposed gevgen/edep-sim ND-GAr production chain

ND-GAr CAFMaker

- Modified makeCAF.C and Params.h adding ND-GAr config variables.
  - Now it will automatically chain the ND-GAr reco products in the CAF if the reco file is provided.
- Created NDGArRecoBranchFiller able to fill some quantities in sr.nd.ndgar using GArSoft anatree.root (see feature/ndgar_cafmaker).
  - Use anatrees as these have a simpler structure.
- For the moment, the only ND-GAr specific fields in duneanaobj/StandardRecord come from the “pseudo reco”.
  - We prepared also a branch in duneanaobj (again feature/ndgar_cafmaker) to start replacing these with the proper reco information.

ND-GAr CAFMaker

```
int nFSP;
std::vector<int> pdg;
std::vector<float> ptrue;
std::vector<float> trkLen;
std::vector<float> trkLenPerp;
std::vector<float> partEvReco;
int gastpc_pi_pl_mult;
int gastpc_pi_min_mult;
```

Variables declared in duneanaobj/StandardRecord/SRGAr.h

```
void NDGArRecoBranchFiller::_FillRecoBranches(std::size_t ii,
                                              caf::StandardRecord &sr,
                                              const cafmaker::Params &par) const
{

NDGArRecoTree->GetEntry(ii);

std::cout << "Event number: " << fEvent << std::endl;
std::cout << "Number of particles: " << fTrackStartX->size() << std::endl;

//todo: currently filling pseudo reco variables in SRGAr.h
//once GAr reco is integrated they should be replaced with proper reco

//using forward variables only!

int pi_pl_mult = 0;
int pi_min_mult = 0;

sr.nd.nFSP = fTrackStartX->size();

for (size_t i=0; i< fTrackStartX->size(); ++i){
    sr.nd.gar.pdg.push_back(fTrackPIDCheatedF->at(i));
    sr.nd.gar.ptrue.push_back(fTrackPF->at(i));
    sr.nd.gar.trkLen.push_back(fTrackLenF->at(i));
    if (fTrackPIDCheatedF->at(i) == 211){
    ++pi_pl_mult;
    } else if (fTrackPIDCheatedF->at(i) == -211){
    ++pi_min_mult;
    }
}
sr.nd.gar.gastpc_pi_pl_mult = pi_pl_mult;
sr.nd.gar.gastpc_pi_min_mult = pi_min_mult;
}
```

Main loop in NDGArRecoBranchFiller.

ND-GAr CAFMaker

```
int nFSP;
std::vector<int> pdg;
std::vector<float> ptrue;
std::vector<float> trkLen;
std::vector<float> trkLenPerp;
std::vector<float> partEvReco;
int gastpc_pi_pl_mult;
int gastpc_pi_min_mult;
```

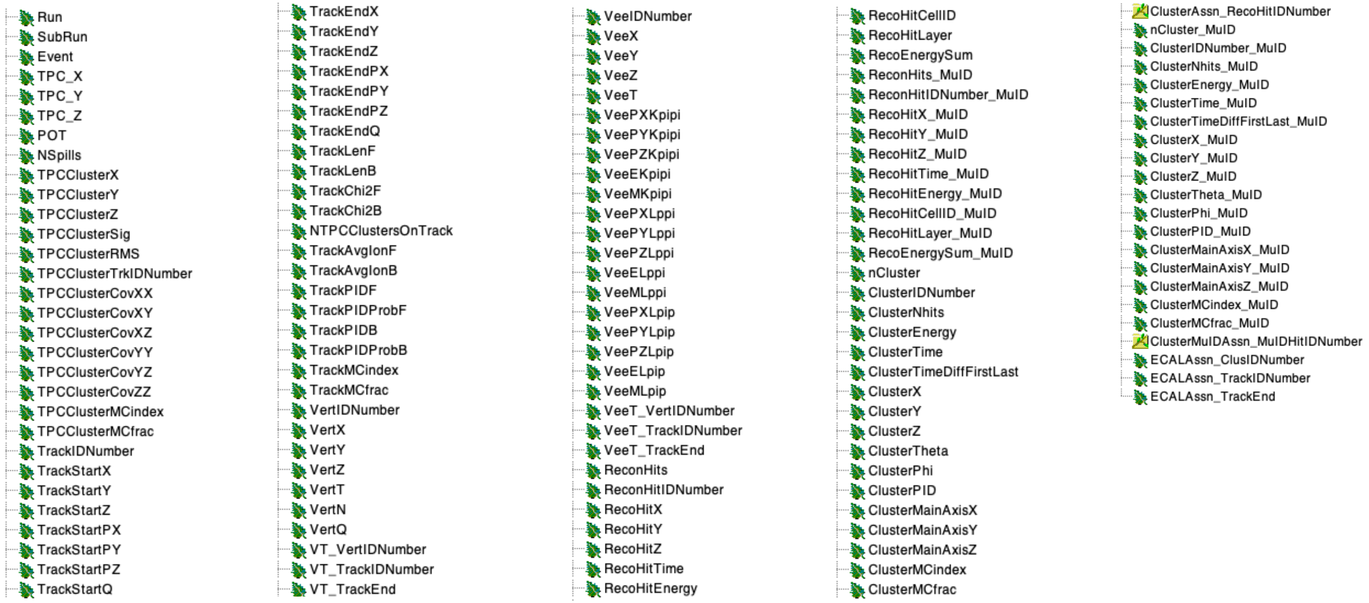
Variables declared in duneanaobj/StandardRecord/SRGAr.h

```
void NDGArRecoBranchFiller::_FillRecoBranches(std::size_t ii,
                                              caf::StandardRecord &sr,
                                              const cafmaker::Params &par) const
{

NDGArRecoTree->GetEntry(ii);

std::cout << "Event number: " << fEvent << std::endl;
std::cout << "Number of particles: " << fTrackStartX->size() << std::endl;

//todo: currently filling pseudo reco variables in SRGAr.h
//once GAr reco is integrated they should be replaced with proper reco

//using forward variables only!

int pi_pl_mult = 0;
int pi_min_mult = 0;

sr.nd.nFSP = fTrackStartX->size();

for (size_t i=0; i< fTrackStartX->size(); ++i){
    sr.nd.gar.pdg.push_back(fTrackPIDCheatedF->at(i));
    sr.nd.gar.ptrue.push_back(fTrackPF->at(i));
    sr.nd.gar.trkLen.push_back(fTrackLenF->at(i));
    if (fTrackPIDCheatedF->at(i) == 211){
    ++pi_pl_mult;
    } else if (fTrackPIDCheatedF->at(i) == -211){
    ++pi_min_mult;
    }
}
sr.nd.gar.gastpc_pi_pl_mult = pi_pl_mult;
sr.nd.gar.gastpc_pi_min_mult = pi_min_mult;
}
```

Main loop in NDGArRecoBranchFiller.

ND-GAr CAFMaker - Next steps

- What ND-GAr reco variables should go in the CAFs?
- Do we need all the info in the anatree or just a subset of these?
- Maybe we can look at what ND-LAr and TMS did.



Information in a *GArSoft* anatree.root (excluding MC info).

```
BUFFER_LOOKUP_VAR(caf::SRTrack, trk_start_x)
BUFFER_LOOKUP_VAR(caf::SRTrack, trk_start_y)
BUFFER_LOOKUP_VAR(caf::SRTrack, trk_start_z)
BUFFER_LOOKUP_VAR(caf::SRTrack, trk_end_x)
BUFFER_LOOKUP_VAR(caf::SRTrack, trk_end_y)
BUFFER_LOOKUP_VAR(caf::SRTrack, trk_end_z)
BUFFER_LOOKUP_VAR(caf::SRTrack, trk_start_dir_x)
BUFFER_LOOKUP_VAR(caf::SRTrack, trk_start_dir_y)
BUFFER_LOOKUP_VAR(caf::SRTrack, trk_start_dir_z)
BUFFER_LOOKUP_VAR(caf::SRTrack, trk_end_dir_x)
BUFFER_LOOKUP_VAR(caf::SRTrack, trk_end_dir_y)
BUFFER_LOOKUP_VAR(caf::SRTrack, trk_end_dir_z)
BUFFER_LOOKUP_VAR(caf::SRTrack, trk_visE)

BUFFER_LOOKUP_VAR(caf::SRShower, shw_start_x)
BUFFER_LOOKUP_VAR(caf::SRShower, shw_start_y)
BUFFER_LOOKUP_VAR(caf::SRShower, shw_start_z)
BUFFER_LOOKUP_VAR(caf::SRShower, shw_dir_x)
BUFFER_LOOKUP_VAR(caf::SRShower, shw_dir_y)
BUFFER_LOOKUP_VAR(caf::SRShower, shw_dir_z)
BUFFER_LOOKUP_VAR(caf::SRShower, shw_visE)
```

ND-LAr reco info going into the CAFs (from ND_CAFMaker)

ND-GAr CAFMaker - Next steps

- What ND-GAr reco variables should go in the CAFs?
- Do we need all the info in the anatree or just a subset of these?
- Maybe we can look at what ND-LAr and TMS did.

- Run
- SubRun
- Event
- TPC_X
- TPC_Y
- TPC_Z
- POT
- NSpills
- TPCClusterX
- TPCClusterY
- TPCClusterZ
- TPCClusterSig
- TPCClusterRMS
- TPCClusterTrkIDNumber
- TPCClusterCovXX
- TPCClusterCovXY
- TPCClusterCovXZ
- TPCClusterCovYY
- TPCClusterCovYZ
- TPCClusterCovZZ
- TPCClusterMCindex
- TPCClusterMCfrac
- TrackIDNumber
- TrackStartX
- TrackStartY
- TrackStartZ
- TrackStartPX
- TrackStartPY
- TrackStartPZ
- TrackStartQ

- TrackEndX
- TrackEndY
- TrackEndZ
- TrackEndPX
- TrackEndPY
- TrackEndPZ
- TrackEndQ
- TrackLenF
- TrackLenB
- TrackChi2F
- TrackChi2B
- NTPCClustersOnTrack
- TrackAvgIonF
- TrackAvgIonB
- TrackPIDF
- TrackPIDProbF
- TrackPIDB
- TrackPIDProbB
- TrackMCindex
- TrackMCfrac
- VertIDNumber
- VertX
- VertY
- VertZ
- VertT
- VertN
- VertQ
- VT_VertIDNumber
- VT_TrackIDNumber
- VT_TrackEnd

- VeeIDNumber
- VeeX
- VeeY
- VeeZ
- VeeT
- VeePXKpipi
- VeePYKpipi
- VeePZKpipi
- VeeEKpipi
- VeeMKpipi
- VeePXLppi
- VeePYLppi
- VeePZLppi
- VeeELppi
- VeeMLppi
- VeePXLpip
- VeePYLpip
- VeePZLpip
- VeeELpip
- VeeMLpip
- VeeT_VertIDNumber
- VeeT_TrackIDNumber
- VeeT_TrackEnd
- ReconHits
- ReconHitIDNumber
- RecoHitX
- RecoHitY
- RecoHitZ
- RecoHitTime
- RecoHitEnergy

- RecoHitCellID
- RecoHitLayer
- RecoEnergySum
- ReconHits_MuID
- ReconHitIDNumber_MuID
- RecoHitX_MuID
- RecoHitY_MuID
- RecoHitZ_MuID
- RecoHitTime_MuID
- RecoHitEnergy_MuID
- RecoHitCellID_MuID
- RecoHitLayer_MuID
- RecoEnergySum_MuID
- nCluster
- ClusterIDNumber
- ClusterNhits
- ClusterEnergy
- ClusterTime
- ClusterTimeDiffFirstLast
- ClusterX
- ClusterY
- ClusterZ
- ClusterTheta
- ClusterPhi
- ClusterPID
- ClusterMainAxisX
- ClusterMainAxisY
- ClusterMainAxisZ
- ClusterMCindex
- ClusterMCfrac

- ClusterAssn_RecoHitIDNumber
- nCluster_MuID
- ClusterIDNumber_MuID
- ClusterNhits_MuID
- ClusterEnergy_MuID
- ClusterTime_MuID
- ClusterTimeDiffFirstLast_MuID
- ClusterX_MuID
- ClusterY_MuID
- ClusterZ_MuID
- ClusterTheta_MuID
- ClusterPhi_MuID
- ClusterPID_MuID
- ClusterMainAxisX_MuID
- ClusterMainAxisY_MuID
- ClusterMainAxisZ_MuID
- ClusterMCindex_MuID
- ClusterMCfrac_MuID
- ClusterMuIDAssn_MuIDHitIDNumber
- ECALAssn_ClusIDNumber
- ECALAssn_TrackIDNumber
- ECALAssn_TrackEnd

Information in a *GArSoft* anatree.root (excluding MC info).

```
SetBranchAddress("nLines", &_nLines);
SetBranchAddress("nHitsInTrack", _nHitsInTrack);
SetBranchAddress("TrackLength", _TrackLength);
SetBranchAddress("TotalTrackEnergy", _TotalTrackEnergy);
SetBranchAddress("Occupancy", _Occupancy);

SetBranchAddress("DirectionX_Upstream", _DirectionX_Upstream);
SetBranchAddress("DirectionZ_Upstream", _DirectionZ_Upstream);

SetBranchAddress("DirectionX_Downstream", _DirectionX_Downstream);
SetBranchAddress("DirectionZ_Downstream", _DirectionZ_Downstream);

SetBranchAddress("TrackHitPos", _TrackHitPos);
```

TMS reco info going into the CAFs (from ND_CAFMaker)

Conclusions

- First steps towards the production of ND-GAr CAFs within the integrated ND_CAFMaker framework.
  - Required ND-GAr specific variables added to makeCAF configuration.
  - Skeleton of NDGArRecoBranchFiller available in our feature branch, so far only filling some simple quantities.
- Progress on integrating the ND-GAr MC production with ND_Production will provide us with files to test the CAFMaker.
  - Short term goal: make CAFs with reco from gevgen/edep-sim chain.
- Next steps: need to decide what ND-GAr reco variables to put in the CAFs.