Designing Domain-Specific Languages for HEP Data Analysis

[1]Junkun Jiao, [2]Tong Liu
[1]Jilin University
[2]IHEP
on behalf of **Dr. Sai** working group

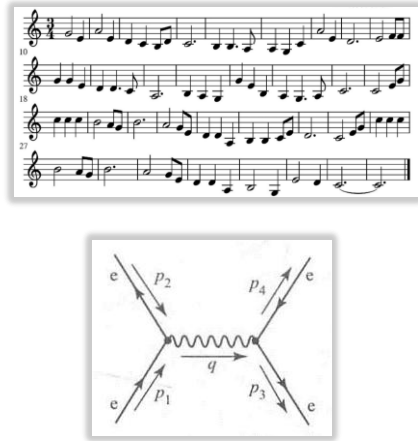
Machine Learning Workshop
IHEP, Beijing, April 14, 2026

Outline

- **Introduction:** What is a DSL?
- **Motivation:** Why DSL?
- ***HepScript*:** A DSL for BESIII Analysis
  - Design principles
  - Implementation details
  - Generation by LLMs
  - Evaluation
- **Discussion:** Future Plan
- **Example:** A Complete Analysis Pipeline
- **(*)Practice:** Design Your Own DSL

Introduction: What is a DSL

- "A **D**omain-**S**pecific **L**anguage (DSL) is a **computer programming language** of **limited expressiveness** focused on a **particular domain**" by Martin Fowler[1].

Based on a general purpose language (GPL) and more like a library of its host language.	A new and independent language with customed design.	Codes and texts are sometimes too complex.
SQLAlchemy Java with jOOQ	MySQL Regex The LaTeX Project CMake	
Internal DSL	**External DSL**	**Non-textual DSL**

Three Types of DSLs

[1] Martin Fowler. Domain-specific languages. China Machine Press, Beijing, 2013.

Introduction: DSLs' core value

- What can DSLs do?

```python
def apply_discount(customer):
    if customer.status == "premium" and customer.order_total > 100:
        customer.discount = 0.10
```

Versus

```
DSL

rule "Premium discount":
    when customer is premium and order total > $100
    then discount = 10%
```

Function definitions, condition syntax, and variable assignments --- things that users don't understand or need to care about.

The user (maybe a marketing person) can read, write, and modify this rule directly. No programming concepts required.

The DSL hides technical details, matches the user's mental model, and reduces errors.

This also works for agentic AI!

Motivation: Automating HEP workflows

- The data analysis workflow in high-energy physics (HEP) is time-consumption, involving substantial complexity. Increasing data volume is also outpacing traditional analysis manpower.
- Record & reproduce analysis.

Physical Motivation

Investigate and survey papers, reports, books, etc., to find a meaningful and feasible research topic.

01

MC Generation

Generate exclusive MC samples for signal and dominant background channels.

02

Event Selection

Separate the signal events from background events, based on characteristic variable distributions. And optimize the selection criteria to increase signal to noise ratio.

03

Fitting Procedure

Perform fit to the selected dataset, estimate parameters such as mass window range, signal yield, angular distribution parameter, etc.

04

Calculate Quantities to be Measured

Based on the physical knowledge, use the estimated parameters from fit to calculate quantities to be measured, such as the cross section.

05

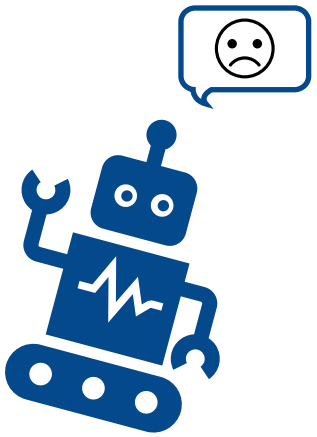
Systematic Uncertainty

Estimate the sources of systematic uncertainties during the whole data analysis workflow.

06

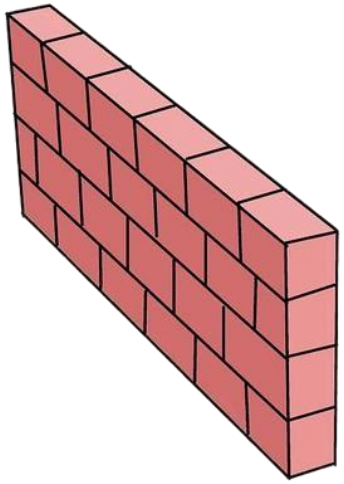
Motivation: Problems during the automation

- Three barriers toward autonomous HEP data analysis



Agentic AI

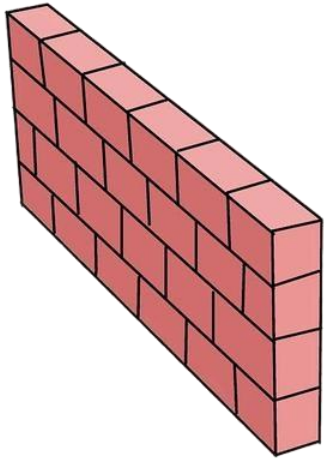
(1) Domain Knowledge



How to embed the extensive (often tacit) rules and procedures of HEP analysis into a standard format for agents?

What to do

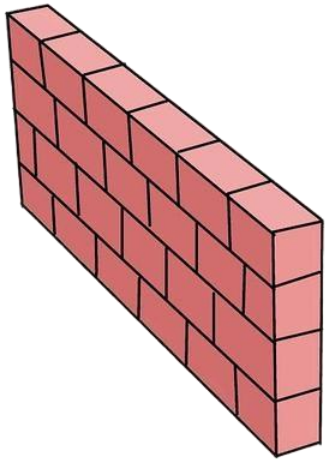
(2) Software Framework



How to bridge the semantic gap between a high-level analysis goal and the low-level, framework-specific codes required to execute it?

How to do

(3) LLM Generating Workflow

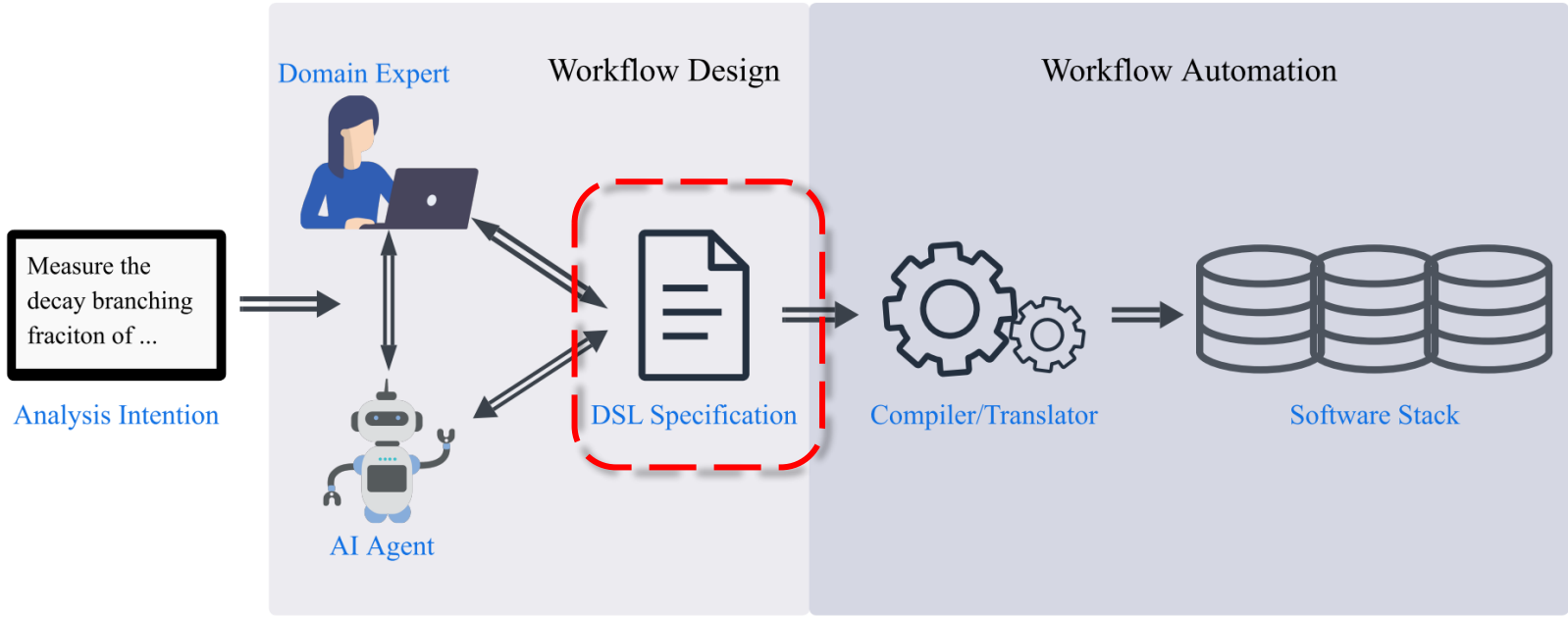


How to guide LLMs to generate logically coherent, multi-step executable workflows.

Control Workflow

Motivation: A workflow DSL as the solution

- Using a DSL to embed the knowledge and formally represent the workflow



A DSL also provides better error feedback.

This approach simultaneously solves all three barriers:
(1) It formalizes the HEP knowledge into its syntax,
(2) integrates with the software framework via a compiler/translator,
(3) and provides the necessary, verifiable structure for LLM generation.

Shift the problem from how to conduct an analysis to what to specify.

DSLs in HEP Community

- DSLs have already been used in HEP experiments.

Internal DSLs	External DSLs
F.A.S.T framework	AwkwardQL
NAIL (Python based)	CutLang&LHADA (ADL)
FuncADL (Python based)	
Bamboo framework	

An example of ADL[4] (event selection)

```
###### EVENT SELECTION
algo __preselection__
cmd "ALL " # to count all events
cmd "nJET >= 6 " # events with 6 or more jets
cmd "MET < 100 " # fully hadronic events should have small MET
#cmd "FillHistos "
#histo "Basics "

algo __teknik1__
__preselection__
cmd "mTopH1 - mTopH2 / 4.2 ^ 2 + mWH1 - 80.4 / 2.1 ^ 2 + mWH2 - 80.4 / 2.1 ^ 2 ~= 0 "
cmd "FillHistos "
histo "mWHh1 , Hadronic W reco (GeV), 50, 50, 150, mWH1 "
histo "mWHh2 , Hadronic W reco (GeV), 50, 50, 150, mWH2 "
```

These DSLs are **limited** to specific functionalities, such as database query or event processing, **unable** (also were not designed) to describe the complete analysis pipeline. **Insufficient** for our requirements of automating HEP workflows.

An exception: *b2luigi* [5]

It's a workflow orchestration tool (Python library).

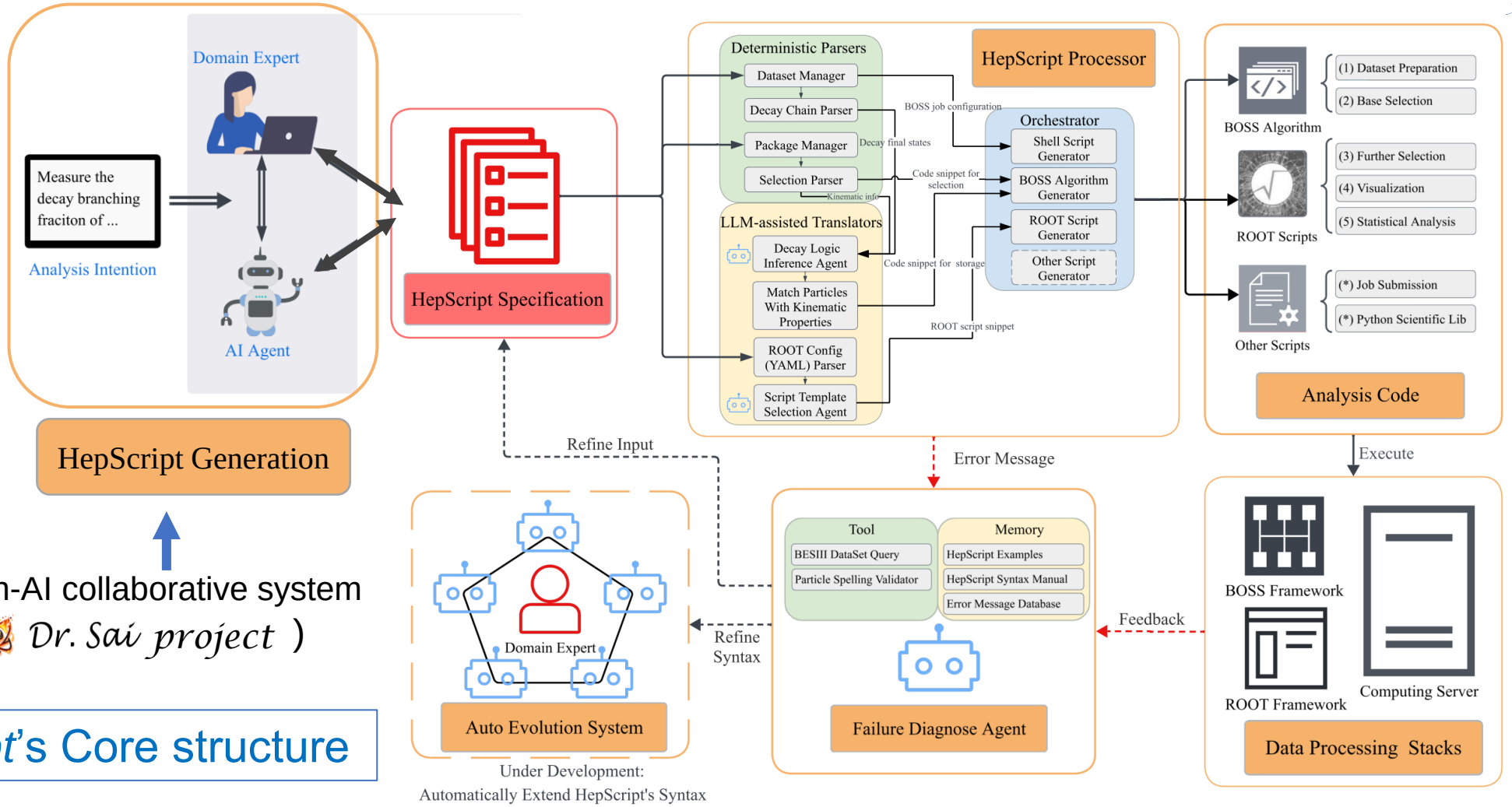
Not explicitly designed for AI agents.

[4] Harrison B. and Sekmen, Sezen and Unel, Gokhan. ArXiv: 2203.09886.

[5] A. Heidelbach, M. Eliachevitch, N. Braun, *et al.*, belle2/b2luigi: v1.2.2 (2025).

HepScript: A dual-use DSL for BESIII Analysis

- We propose to develop a new DSL for BESIII analysis, named ***HepScript,*** which is designed for both human experts and AI agents.



HepScript's Core structure

HepScript: Design principles

- Code generation architecture

  BESIII data analysis workflow involves two distinct software frameworks:

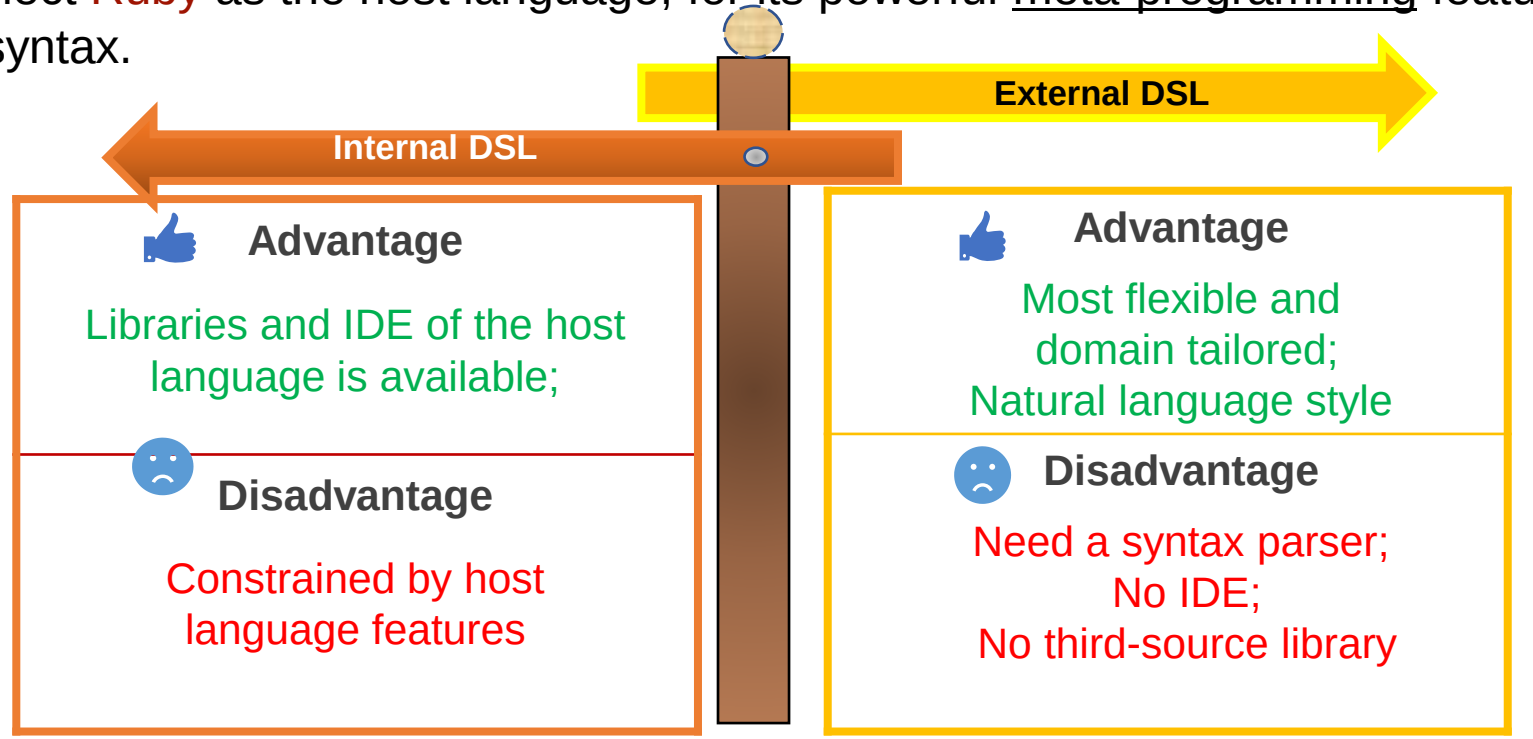
BOSS: Simulation, reconstruction, and basic data selection;

ROOT: Further data selection, statistical inference (fit), and visualization (plot).

Impractical to create a single compiled language spanning both frameworks!

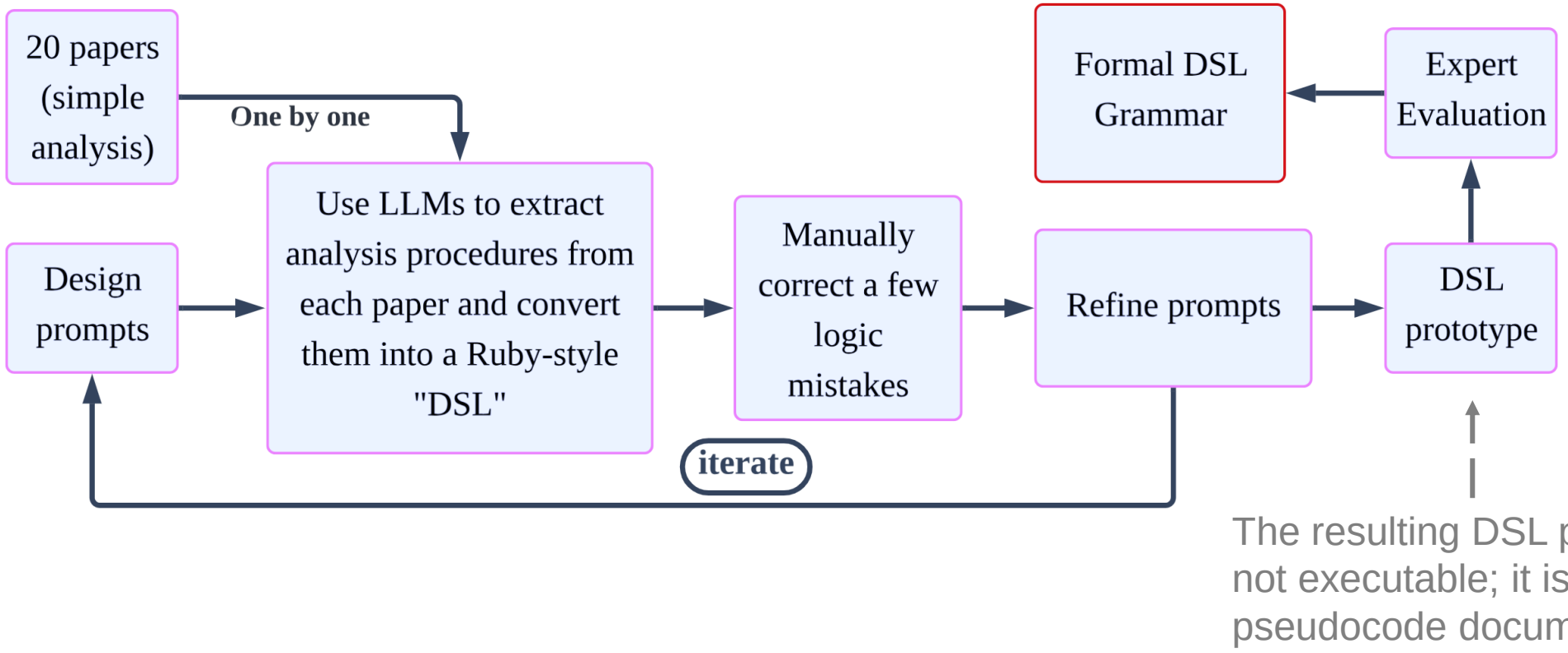
- Internal DSL schema

  We select Ruby as the host language, for its powerful meta-programming features, flexible and highly readable syntax.



HepScript: Design principles

- LLM-assisted design: (1) Investigate sematic patterns from BESIII publications.
  (2) Design language constructs from these semantic patterns.



Through this process, we establish a distillation of the domain model for BESIII data analysis and instantiated it into HepScript representation.

HepScript: Design principles

- An example of $\psi(3686) \to \Lambda\bar{\Sigma}^0\omega$

```
### Base Selection (BOSS) ###
alg_name = "LSomega"
my_Algorithm = Algorithm.new(alg_name)
my_Algorithm.set_header(["#{alg_name}Alg/#{alg_name}.h"])  # set the header file of the algorithm.
            .set_constant({"ECMS"=> [:double, 3.686]})
event_selection=Selection.new
event_selection.select_track {  # some requirements for charged track selection
    cos_theta   0.93
    Vz   100.0
    Vr   10.0
    nChrp   ">=3"
    nChrn   ">=3"
    nNet   "==0"
}
.select_photon {  # some requirements for photon selection
  tdc_emc_start   0
  tdc_emc_end   14
  angle_to_track   10.0
  energyThreshold_b 0.025
  energyThreshold_e  0.050
  nGam   ">=3"
}
.pid(has_lepton = false) {  # requirements for PID (particle identification) selection
  prob_cut   0.001
  identify :proton, "from_kaon_and_pion"
  nprp   ">=1"
  nprm   ">=1"
}
.remove([:prp <= :chrgp, :prm <= :chrgn])
.select_isolated_photon {
  angle_to_prm_track   20.0
  nGam   ">=3"
}
.kalman_kinematic_fit([:gamma, :gamma]) { # reconstruct specific intermediate states
    invariant_mass_of(:gamma,:gamma).constrain_to_nominal_mass_of(:pi0)
    chi2_cut 25
    npi0  ">=1"
}
.assign({:chrgp => :pip, :chrgn => :pim})   # assume the positive charged tracks as pip withou PID. Similarly, negative charged tracks as pim.
.secondary_vertex_fit([:prp, :pim]){   # perform a secondary vertex fit for the prp and pim particles
    build_virtual_particle(:Lambda).by_minimizing_mass_difference   # reconstruct the Lambda baryon
    remove_used_particle_from_candidate_list
}
```

```
.secondary_vertex_fit([:prm, :pip]){
    build_virtual_particle(:Lambda_bar).by_minimizing_mass_difference
    remove_used_particle_from_candidate_list
}
.kinematic_fit( # perform a kinematic fit with multiple hypotheses
  [:Lambda, :Lambda_bar, :gamma, :pi0, :pip, :pim]) {
  nominal # nominal kinematic fit, use momentum of particles from this fit.
  vertex_fit([4, 5])   # perform a vertex fit to ensure that the pip and pim are from omega are from common vertex.
  constrain_four_momentum   # constrain sum of the participated particles' four-momentum to the CMS system.
}
.kinematic_fit([:Lambda, :Lambda_bar, :gamma, :gamma, :pi0, :pip, :pim]) { # perform an additional kinematic fit with the one more photon
  constrain_four_momentum
  vertex_fit([5, 6])
}
my_Algorithm.with_decay_card(decay_card_for_signal).apply(event_selection)
root_files = my_Algorithm.execute_on([psip_data, psip_incMC, exMC_signal]) # execute the algorithm on datasets to obtain root files
```

Here shows the event selection part of the complete workflow in the DSL specification, including: charged track selection; photon selection; PID; vertex fit; secondary vertex fit; kinematic fit with mass constraint available.

- ☑ High readability
- ☑ Modular structure
- ☑ Enable direct translation to BOSS algorithm
- ☐ More features under developing

☑ *HepScript:* Implementation details

- Implementation: Hybrid Code Generation Strategy
  The DSL processor translates HepScript specifications into BOSS/ROOT code via three integrated approaches.

Strategy	When to Use	How It Works	Example / Visual Cue
Templated Generation	Code with static structure and few dynamic variables	Placeholder {{key}} substitution from DSL spec	📄 `event_selection.select_track { cos_theta 0.93 Vz 100.0 Vr 10.0 nChrp ">=3" nChrn ">=3" nNet "==0" }`
Translator-Based Generation	Flexible or semantically complex syntax	Custom Ruby classes parse DSL → target code	⚙ **DSL:** .kinematic_fit([:Lambda, :Lambda_bar, :gamma, :gamma, :pi0, :pip, :pim]) { constrain_four_momentum vertex_fit([5, 6]) }
LLM-Assisted Generation	Ambiguous, analysis-dependent logic	Prompt LLM with physics constraints and examples	🧠 **Cascade Decay:** γγ pairing (π^0 vs η) distinguished by mass minimization 📊 **ROOT Script:** YAML + exemplars → LLMs → plotting/fitting script

Key Feature: Syntax-directed translation, DSL syntax maps directly to individual phases in the analysis workflow.

HepScript: Generation by LLMs

- LLM-Generation: **RAG** vs **Fine-tuning** (comparable performance[6])

  We prefer RAG, because fine-tuning is expensive, difficult to optimize, and cannot easily adapt to a rapidly evolving DSL.

- LLM-Generation: In-context learning

  Since the RAG remains under development (refer to Zijie's talk), we use a **comprehensive *HepScript* example** and a **YARD grammar reference** to guide LLMs, to generate *HepScript* specifications.

Component	Content	Purpose
Comprehensive Workflow Example	Complete end-to-end HepScript specification for a canonical BESIII analysis	Demonstrates intended usage of all core constructs in a realistic workflow
Complete Grammar Reference	Auto-generated YARD documentation for public HepScript APIs	Serves as authoritative, agent-readable syntax reference for LLM consultation

```ruby
module EventCut
# Calculates invariant mass of particle combination
  # @param args [Array<Symbol>] Particle names for mass calculation
  # @return [EventCut] self for method chaining
  # @raise [ArgumentError] if invalid particle types provided
  # @example Calculate pip pim invariant mass
  #   invariant_mass_of(:pip, :pim)
  def invariant_mass_of(*args)
  end
# Applies range cut (require a variable to stay between the lower limit and upper limit)
  # @param min [Float] Lower limit value
  # @param max [Float] Upper limit value
  # @return [EventCut] self for method chaining
  # @note frequently used for mass window cuts
  # @example Select J/psi mass window, combining ep and em tracks
  #   invariant_mass_of(:ep,:em).between(3.0, 3.2)  # GeV
  def between(min, max)
  end
end
```

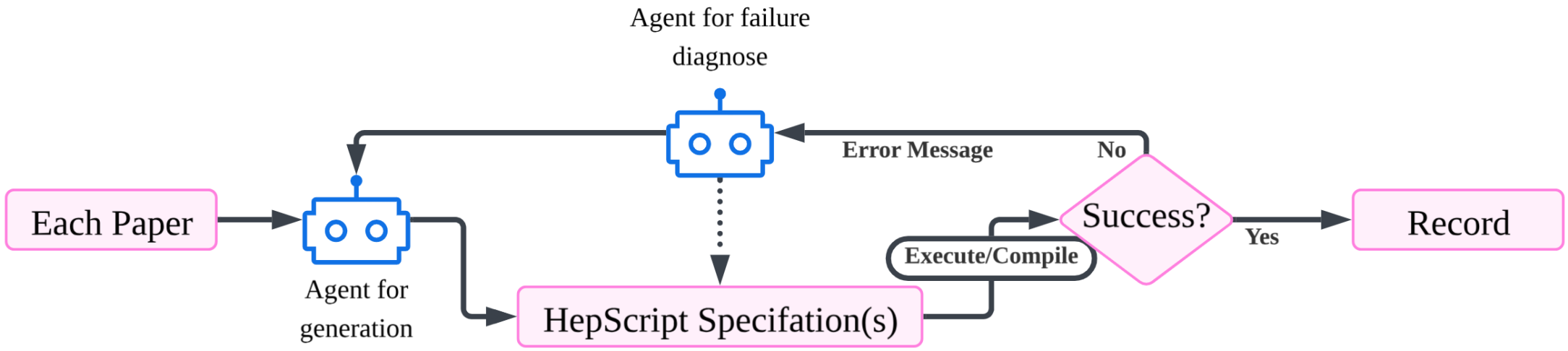
Example of the YARD format syntax reference

[6] N. Bassamzadeh and C. Methani, arXiv:2407.02742 (2024).

HepScript: Evaluation of its dual-use capability

- Evaluation Method: LLM-Generation within a simple agentic loop

We evaluate HepScript along two dimensions: as a **human-facing abstraction** (fidelity and code reduction) and as an **AI-facing interface** (LLM generation success from BESIII papers)



Experiment Setup	Description
Test Corpus	50 BESIII papers (45 remains, filtered those difficult to express from arXiv publications by date)
Ground Truth	Human-written HepScript specifications per paper, verified by second expert
LLMs Involved	DeepSeek-V3.2, DeepSeek-R1, GPT-4o, GLM-4.7, Qwen3-max
Success Metrics	(1) Syntax Correctness; (2) Logical Correctness; (3) Compilation/Execution Success;

HepScript: Evaluation results

- Evaluation Result: LLM-Generation within a simple agentic loop

We split this evaluation is into three parts: **(1) BOSS code translation** from *HepScript* specifications, **(2) HepScript generation** from BES3 papers, and **(3) ROOT script generation**.

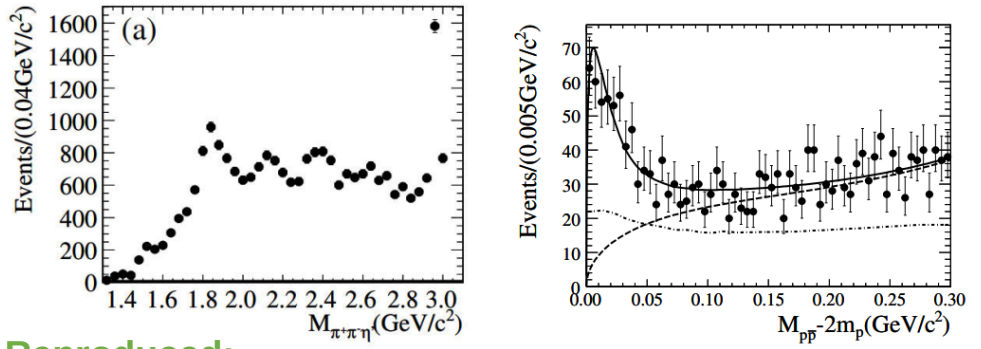
LLM	Initial Success	After One Retry
DeepSeek-V3.2	96.5%	100%
DeepSeek-R1	87.7%	100%
GPT-4o	96.5%	100%
GLM-4.7	96.5%	100%
Qwen3-max	93.8%	100%

(1) BOSS Code Translation: Human-Written Specifications (63 Packages)

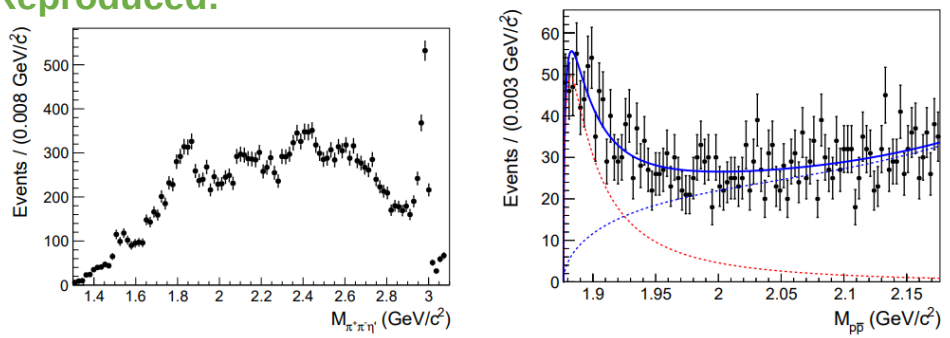
LLM	Initial	one Retry	Three Retries
DeepSeek-R1	47.3%	87.8%	**94.6%**
GLM-4.7	43.2%	90.5%	**95.9%**

(2) HepScript Generation: LLM-Generated Specs (72 Packages)

Original: $J/\psi \to \gamma\pi^+\pi^-\eta'$ $\psi' \to \pi^+\pi^- J/\psi(\to \gamma p\bar{p})$



Reproduced:



(3) ROOT Script Generation: LLM-Reproduced Figures

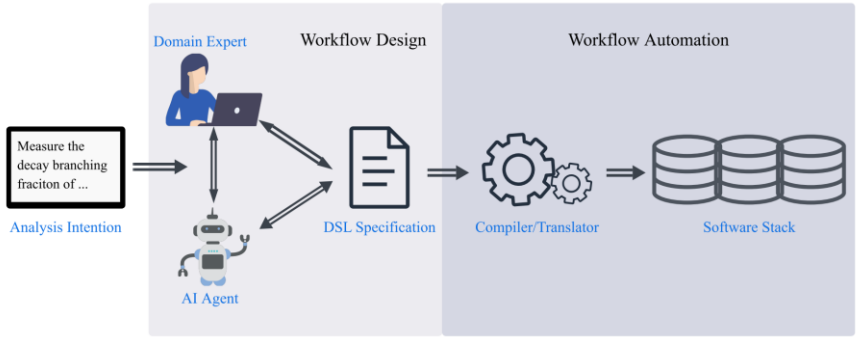
HepScript reduces manual coding effort by **93%** and, combined with an agentic loop, enables LLMs to generate correct workflows with **95%** success after three retries.

Discussion & Conclusion

- A research agenda for DSL-grounded HEP workflow automation

Challenge	Proposed Direction
Expressiveness vs. Generability	Iterative refinement for practical scenarios; add systematic uncertainties; **self-evolutionary mechanism** for grammar extension
Scalable Retrieval	Search for *HepScript* examples with structure-aware retrieval based on decay process (not standard semantic similarity)
Knowledge Preservation	Database of formalized workflow specifications as long-term memory for multi-agent systems; As this database grows, it opens possibilities for **machine learning on workflows** themselves.
Practical Usage	Encapsulate *HepScript* as a **tool or "skill"** for larger multi-agent systems (Dr. Sai).

- Conclusion: A formally specified workflow DSL acts as a shared layer between the human expertise, AI agent automation, and production softwares.



- ✓ A powerful strategy for automating intricate HEP workflows.
- ✓ The principle is generalizable across HEP experiments, or even other scientific domains.

How to use HepScript:

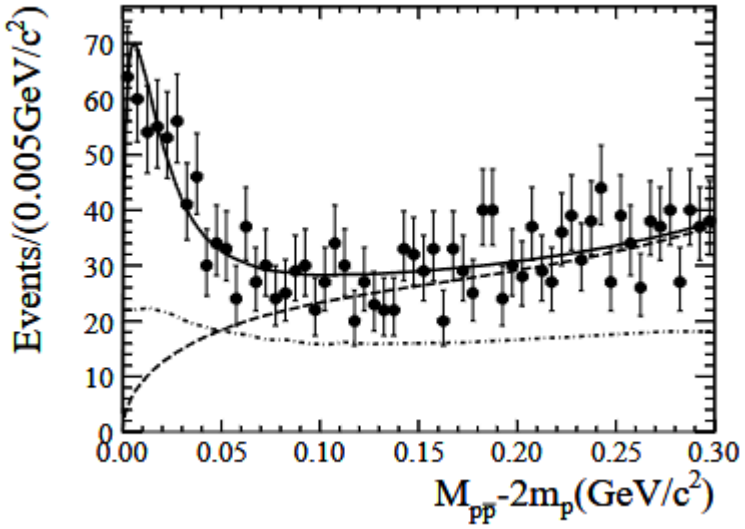
A Case Study by Tong Liu

A complete analysis pipieline

1001.5328v1

Observation of a $p\bar{p}$ mass threshold enhancement in $\psi' \to \pi^+\pi^- J/\psi(J/\psi \to \gamma p\bar{p})$ decay

The decay channel $\psi' \to \pi^+\pi^- J/\psi(J/\psi \to \gamma p\bar{p})$ is studied using a sample of 1.06×10^8 ψ' events collected by the BESIII experiment at BEPCII. A strong enhancement at threshold is observed in the $p\bar{p}$ invariant mass spectrum. The enhancement can be fit with an S-wave Breit-Wigner resonance function with a resulting peak mass of $M = 1861^{+6}_{-13}$ (stat)$^{+7}_{-26}$ (syst) MeV/c^2 and a narrow width that is $\Gamma < 38$ MeV/c^2 at the 90% confidence level. These results are consistent with published BESII results. These mass and width values do not match with those of any known meson resonance.



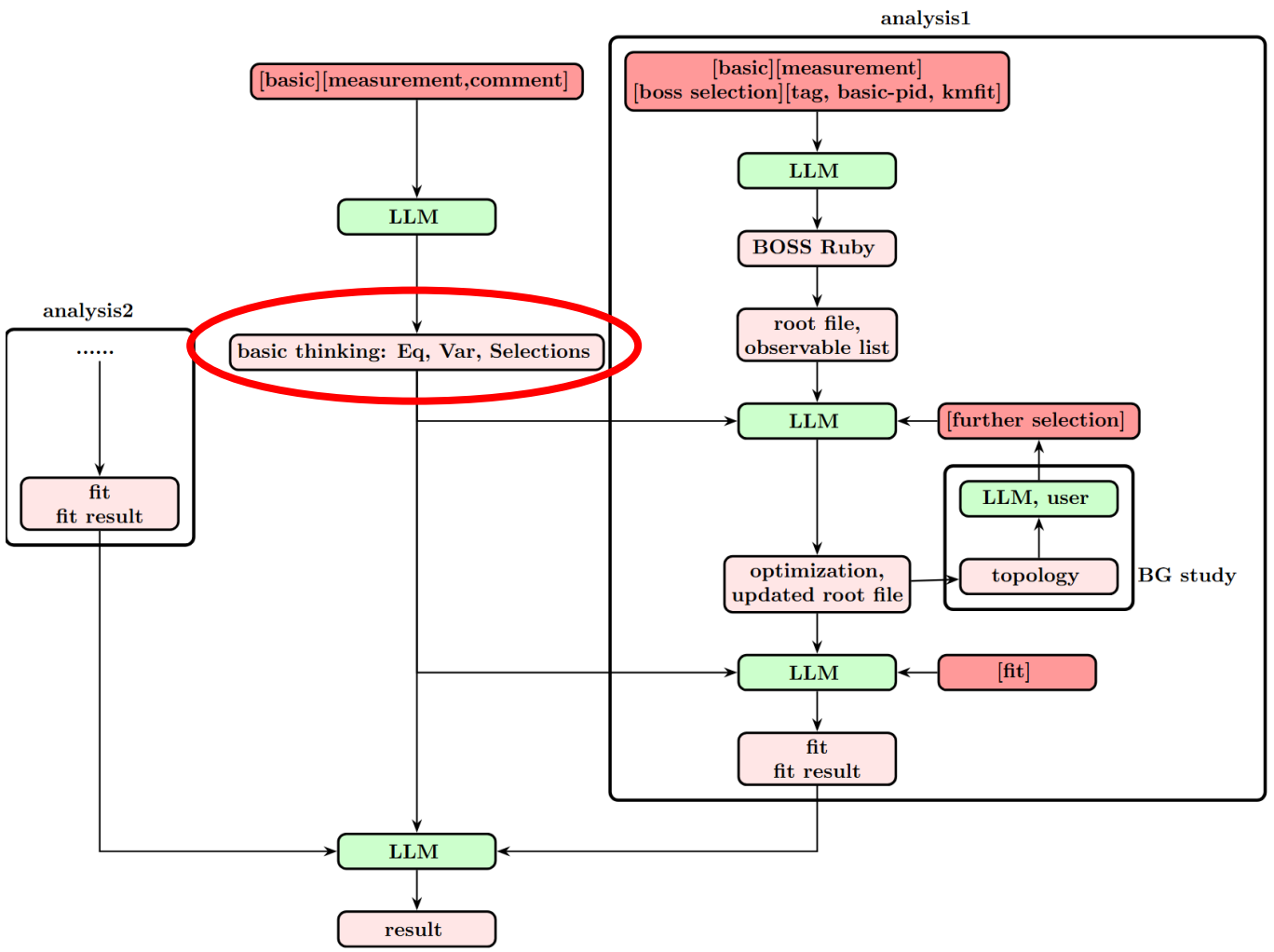
Original paper

LLM


```yaml
metadata:
  name: psip_to_pip_pim_jpsi_jpsi_to_gamma_p_anti-p
  BAM: 000
  dataType: 3686-09
  comment: "Study of psi(2S) -> pip pim jpsi, jpsi -> gamma p anti-p using 1.06e8 psi(2S) events. The analysis aims to observe and characterize a near-threshold enhancement in the p anti-p invariant mass spectrum. Signal yields are determined by fitting the data. The main background from psi(2S) -> pip pim jpsi, jpsi -> pi0 p anti-p is studied using exclusive MC and data sidebands."
  selections:
    - analysis_1: psi(2S) -> pip pim jpsi, jpsi -> gamma p anti-p

analysis_1:
  dir:
    dir: .
    dir_data: None
    dir_sigMC: None
    dir_incMC: None
    dir_draw: None
  sample:
    data: None
    incMC: None
    sigMC:
      - num_events: 50000
        process: psi(2S) -> pip pim jpsi, jpsi -> gamma p anti-p
    otherMC:
      - num_events: 50000
        process: psi(2S) -> pip pim jpsi, jpsi -> pi0 p anti-p
  base_selection:
    comment: "Select events with at least one photon and four charged tracks (|cosTheta|<0.93, net charge=0). PID assigns highest CL hypothesis (pi, K, p). Require one identified proton and one anti-proton; no PID for remaining two tracks. Photon: E>25 MeV (barrel), >50 MeV (endcap), isolated >30 deg from anti-p, >10 deg from other charged tracks. Apply EMC timing cuts. Pre-select using recoil mass: |M_recoil(pip pim) - m_jpsi| < 0.006 GeV/c2. Apply cuts: |U_miss| < 0.05 GeV, Pt_gamma^2 < 0.0005 (GeV/c)^2, and |M(pip pim p anti-p) - m_psi(2S)| > 0.03 GeV/c2 to suppress backgrounds."
    tagged_data: "nCharge==4"
    tag:
      common: pip pim p anti-p gamma
    basic_pid:
      - pi: "momentum less than 0.7 GeV/c"
      - p: "momentum larger than 0.3 GeV/c"
    kmfit:
      - var: 4C
        setting: setChisqCut(100,0.005)
  further_selection:
    chisq:
      - {var: chisq4C, optimize: 1D, strategy: FoM, default: (0,100), Nbin_region: "100, (0,200)"}
    cosTheta:
      - {var: polarAngle_charged, optimize: False, strategy: FoM, default: (-0.93,0.93)}
    theta:
      - {var: openAngle_gamma_anti-p, optimize: False, strategy: FoM, default: (0.5236, 3.1416)}
    Mass:
      - {var: "p4_p + p4_anti-p", optimize: 2D, strategy: Punzi, default: (1.83, 1.89), Nbin_region: "100, (1.8, 2.0)"}
      - {var: "p4_ecms - p4_pip - p4_pim", optimize: False, strategy: FoM, default: (3.087, 3.107), Nbin_region: "100, (3.05, 3.15)"}
      - {var: "p4_ecms - p4_pip - p4_pim - p4_p - p4_anti-p", optimize: False, strategy: FoM, default: (0.0, 0.05), Nbin_region: "100, (-0.1, 0.1)"}
      - {var: "p4_p + p4_anti-p", optimize: False, strategy: FoM, default: (1.8, 2.0), Nbin_region: "100, (1.8, 2.0)"}
  fit:
    Mass:
      - var: "p4_p+p4_anti-p"
        fit_method: 1D
        BG: "fixed_bkg_shape"
        signal: "acceptance_weighted_BW"
        FixPar:
          - "a1 and a2 in background function f_bkg(delta) are fixed at pi0 p anti-p phase-space MC values"
          - "Background shape parameters from exclusive MC and data sidebands"
        par_region: "Peak mass M within (1.84, 1.88) GeV/c^2, width Γ < 0.1 GeV/c^2"
        Nbin_region: "unbin, (1.8, 2.0)"

systematic_uncertainties:
```


YAML configuration file

A complete analysis pipieline



分析目标：观测ψ' → π⁺π⁻J/ψ（J/ψ → γpp̄）衰变中的pp̄质量阈值增强现象，研究pp̄不变质量谱在阈值附近的行为。

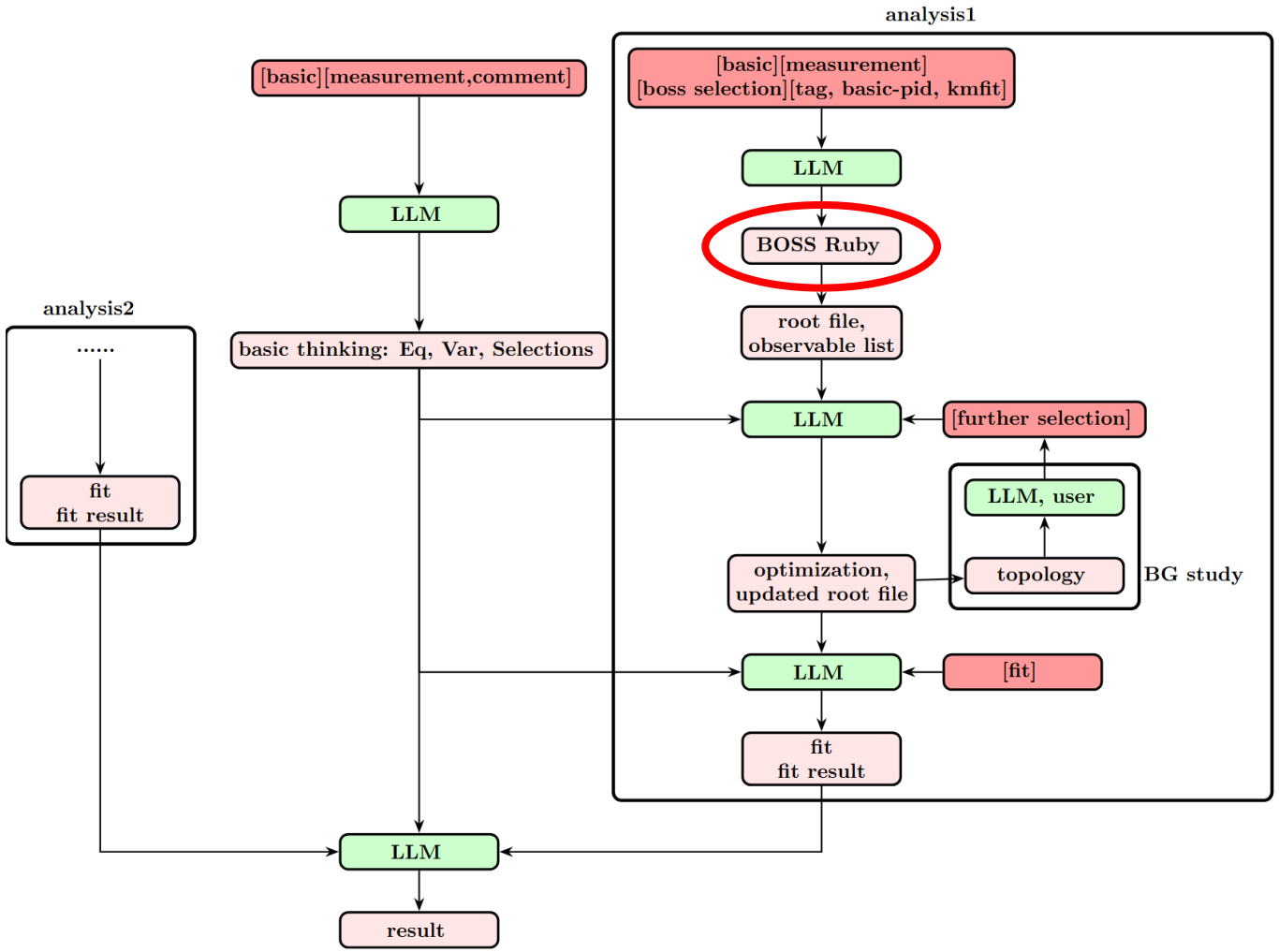
涉及的计算公式
- **pp̄不变质量**：
  \(M_{pp̄} = \sqrt{(E_p + E_{p̄})^2 - (\vec{p}_p + \vec{p}_{p̄})^2} \)
  - \(E_p, E_{p̄} \)：质子（p⁺）和反质子（p̄⁻）的能量
  - \(\vec{p}_p, \vec{p}_{p̄} \)：p⁺和p̄⁻的三维动量
  - **来源**：通过重建末态粒子p⁺（2212）、p̄⁻（-2212）的四动量计算。

事例挑选与样本作用
1. **信号MC样本**：
   - 模拟ψ' → π⁺π⁻J/ψ, J/ψ → γpp̄过程
   - **作用**：
     - 优化事例挑选条件
     - 估计本底贡献和重建效率
     - 模拟pp̄不变质量谱的形状
2. **数据样本**：
   - 真实实验数据（如BESIII的ψ'数据）
   - **作用**：
     - 提取观测的pp̄不变质量谱
     - 与信号MC对比，识别阈值附近的增强结构

事例挑选过程分析
- **目标过程**：ψ' → π⁺π⁻J/ψ（J/ψ → γpp̄）
- **末态粒子重建要求**：
  - **带电径迹**：4条（π⁺, π⁻, p⁺, p̄⁻）
  - **中性径迹**：≥1个（γ）
  - **重建策略**：全重建（所有末态粒子需被探测）
- **粒子共享性**：
  - **不共享**：
    - pp̄对（p⁺和p̄⁻）：研究对象，需独立重建
  - **共享**：
    - π⁺π⁻对：来自ψ'的直接衰变，仅一种来源
    - γ：来自J/ψ衰变，仅一种来源

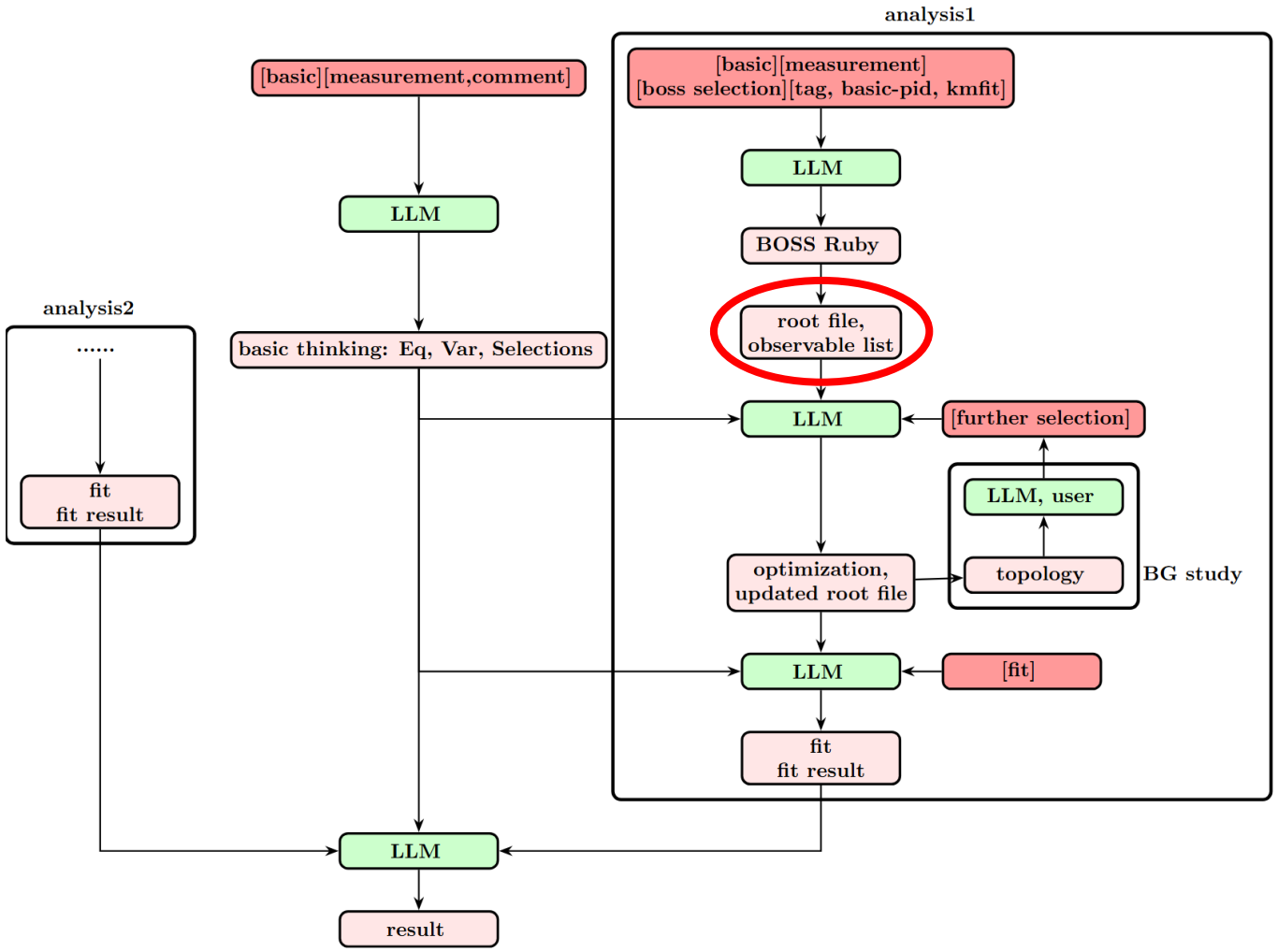
总结
- **关键粒子**：π⁺（211）、π⁻（-211）、γ（22）、p⁺（2212）、p̄⁻（-2212）
- **输出目的**：用于后续优化事例挑选、本底扣除及阈值增强现象的定量分析。

A complete analysis pipieline



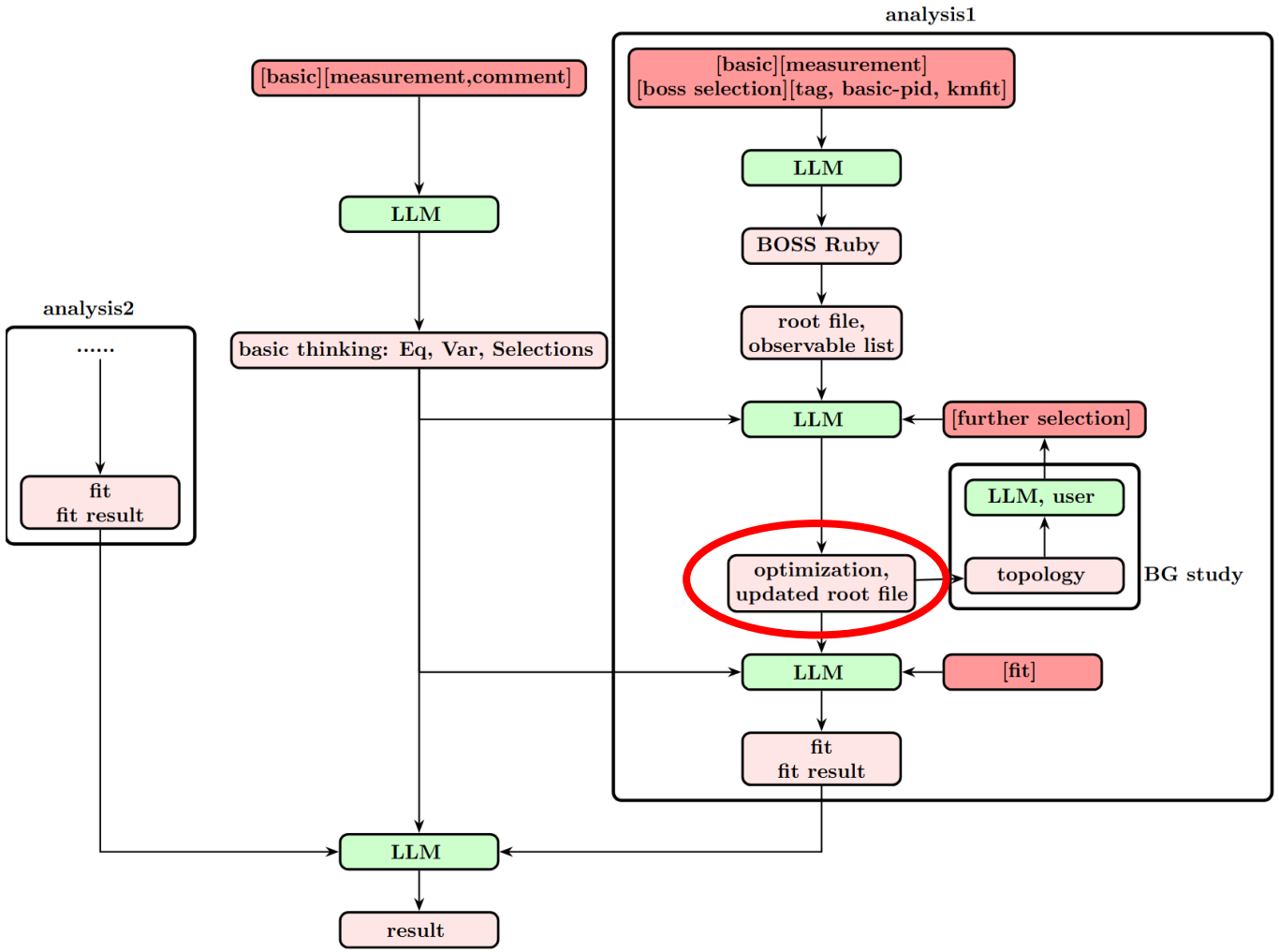

```ruby
# 1. 数据集加载
require_relative '../DSL4BES/Algorithm.rb'
datasets = DatasetManager.load_real_data('config/BES3_dataset.md')
inc_datasets = DatasetManager.load_inclusive_mc('config/BES3_incMC_sample.md')
data = DatasetManager.real_data.find("709_3686")  # psi(2S)数据
incMC = DatasetManager.inclusive_mc.find("709_3686")

# 2. 衰变卡定义
decay_card = <<~DECAYCARD
    Decay psi(2S)
    1.0000  pi+  pi-  J/psi       PHSP;
    Enddecay
    Decay J/psi
    1.0000  gamma  p+  anti-p-       PHSP;
    Enddecay
    End
DECAYCARD

# 3. 独占MC生成
exMC = DatasetManager.create_exclusive_mc do |config|
  config.sample_name = "psi2s_pip_pim_jpsi_gamma_p_pbar"
  config.related_dataset = data
  config.events = 100000
  config.decay_card = decay_card
  config.cross_section = :default
end

# 4. 算法初始化
alg = Algorithm.new('Psi2sPipPimJpsiGammaPPbar')
alg.set_header(["Psi2sPipPimJpsiGammaPPbarAlg/Psi2sPipPimJpsiGammaPPbar.h"])
    .set_constant({"ECMS" => [:double, 3.686]})
    .set_namespace(["CLHEP::Hep3Vector", "CLHEP::HepLorentzVector", "namespace std"])
    .set_alias({"std::vector<double>" => "Vdouble"})

# 5. 衰变卡解析
DecayCardResolver.new(decay_card).parse
generator = NtupleCodeGenerator.new(DecayCardResolver.reconstruction_list)

# 6. 事件选择
event_selection = Selection.new
event_selection.select_track {
                cos_theta   0.93
                Vz          10.0
                Vr          1.0
                nChrp       "==2"
                nChrn       "==2"
```

A complete analysis pipieline

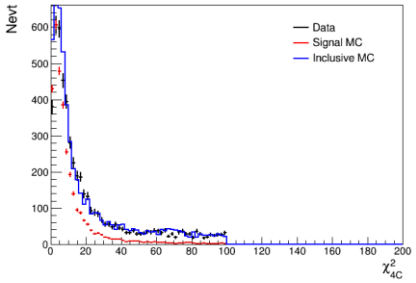


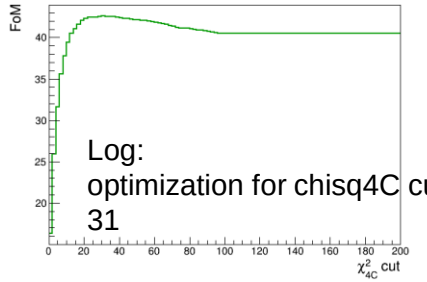

```
indexmc         number of MC particles in the event, following variables are arrays
pdgid           PDG ID of each MC particle
trkIdx          track index of each MC particle
motheridx       mother index of each MC particle
mctrue_P4_psip 4-momentum of true ψ(2S) from MC truth
mctrue_P4_pip  4-momentum of true π+ from MC truth
mctrue_P4_pim  4-momentum of true π- from MC truth
mctrue_P4_J_over_psi 4-momentum of true J/ψ from MC truth
mctrue_P4_gamma 4-momentum of true γ from MC truth
mctrue_P4_prp  4-momentum of true p+ from MC truth
mctrue_P4_prm  4-momentum of true anti-p- from MC truth
runno           run number
evtno           event number
nchrg           total number of charged tracks
nchrp           number of positive charged tracks
nchrn           number of negative charged tracks
ngam            number of photon candidates
npip            number of π+ candidates
npim            number of π- candidates
nprp            number of p+ candidates
nprm            number of anti-p- candidates
nkp             number of K+ candidates
nkm             number of K- candidates
nep             number of e+ candidates
nem             number of e- candidates
nmup            number of μ+ candidates
nmum            number of μ- candidates
P4_psip         4-momentum of reconstructed ψ(2S)
mpsip           invariant mass of reconstructed ψ(2S)
d3_psip         3-momentum magnitude of reconstructed ψ(2S)
cos_psip        cosine of polar angle of ψ(2S) in its rest frame
P4_pip          4-momentum of reconstructed π+
mpip            invariant mass of reconstructed π+
d3_pip          3-momentum magnitude of reconstructed π+
cos_pip         cosine of polar angle of π+ in ψ(2S) rest frame
P4_pim          4-momentum of reconstructed π-
mpim            invariant mass of reconstructed π-
d3_pim          3-momentum magnitude of reconstructed π-
cos_pim         cosine of polar angle of π- in ψ(2S) rest frame
P4_J_over_psi 4-momentum of reconstructed J/ψ
mJ_over_psi    invariant mass of reconstructed J/ψ
d3_J_over_psi 3-momentum magnitude of reconstructed J/ψ
cos_J_over_psi cosine of polar angle of J/ψ in ψ(2S) rest frame
P4_gamma        4-momentum of reconstructed γ
```

A complete analysis pipieline



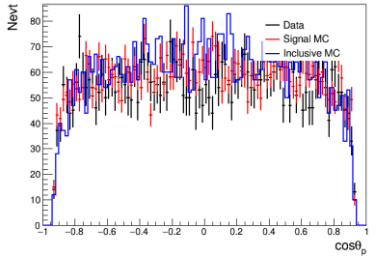
Optimization result:

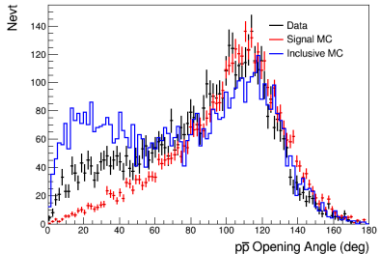


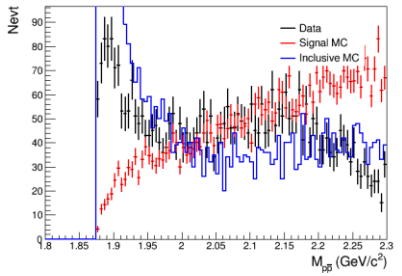


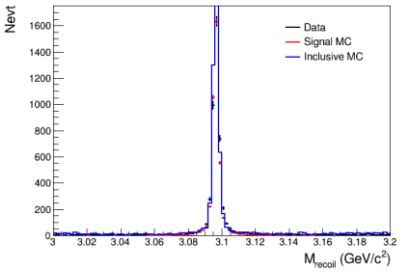
Log:
optimization for chisq4C cut: 31

And other unoptimized variables:

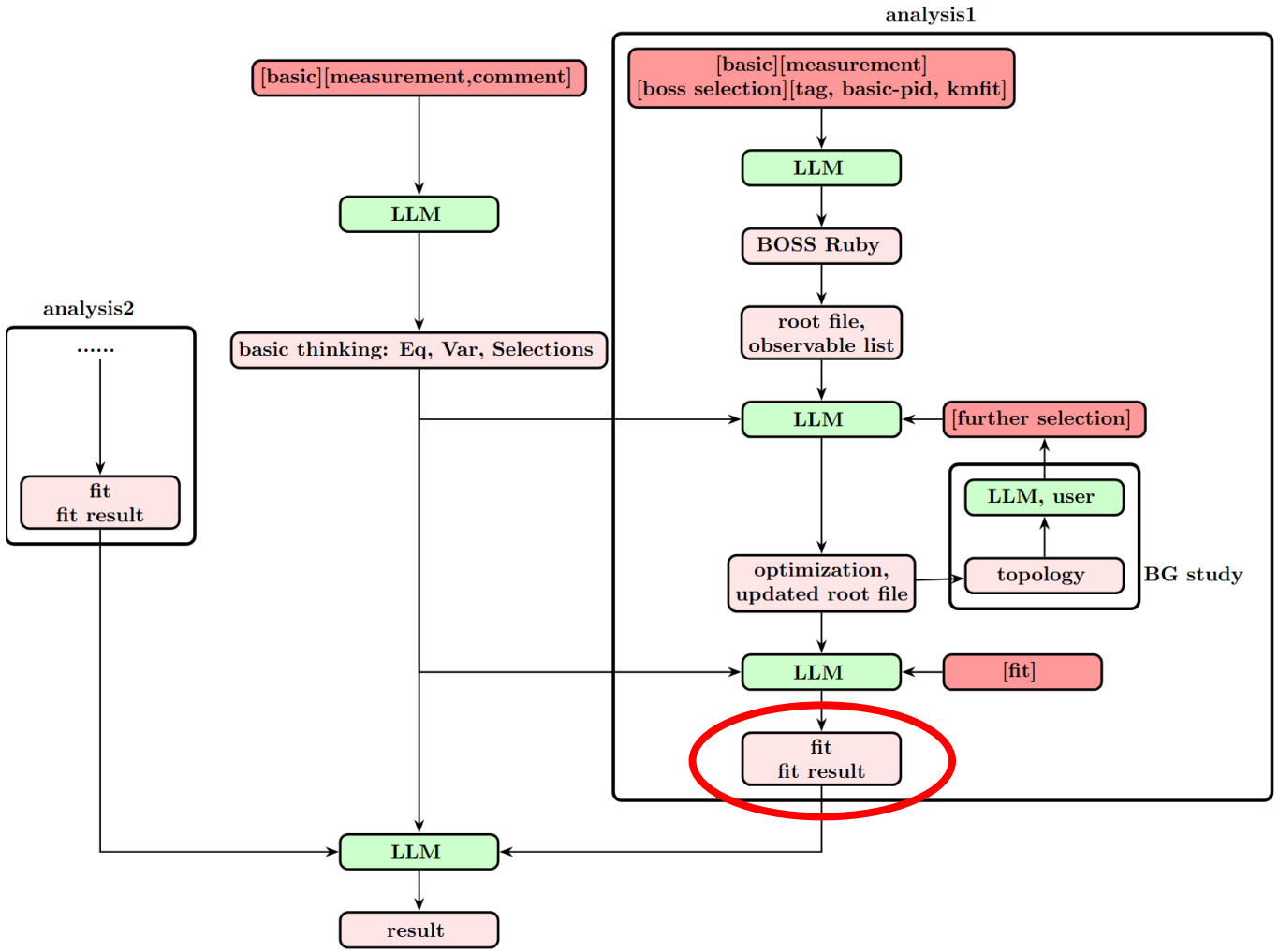


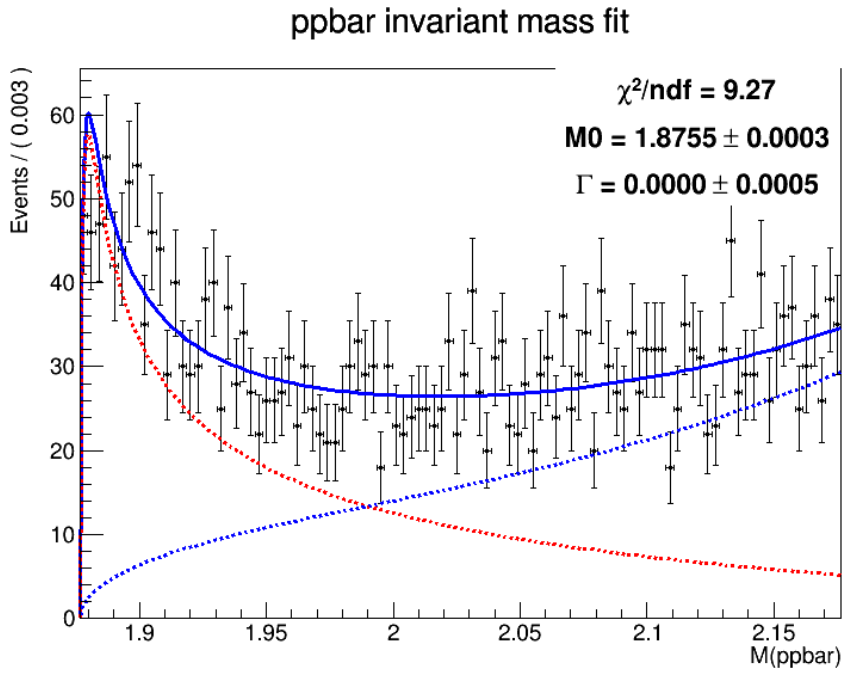




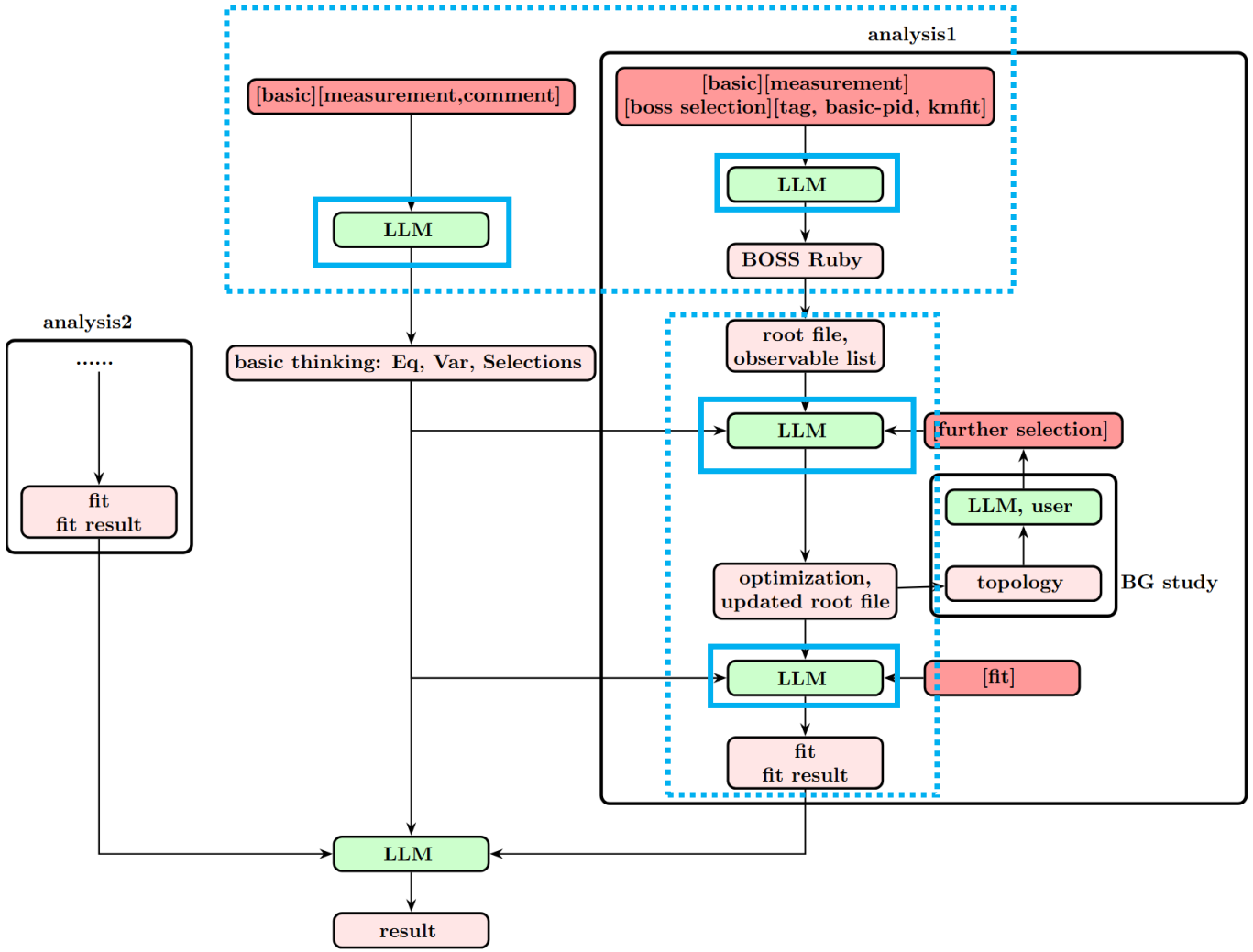


A complete analysis pipieline





A complete analysis pipieline



Big-skills:

Small-skills:

What are skills?

Agent Skills are a lightweight, open format for extending AI agent capabilities with specialized knowledge and workflows.

```
my-skill/
├── SKILL.md          # Required: instructions + metadata
├── scripts/          # Optional: executable code
├── references/       # Optional: documentation
└── assets/           # Optional: templates, resources
```

THANK YOU 

☑ (*) Design Your Own DSL

Example Scenario: The Smart Home Routine System

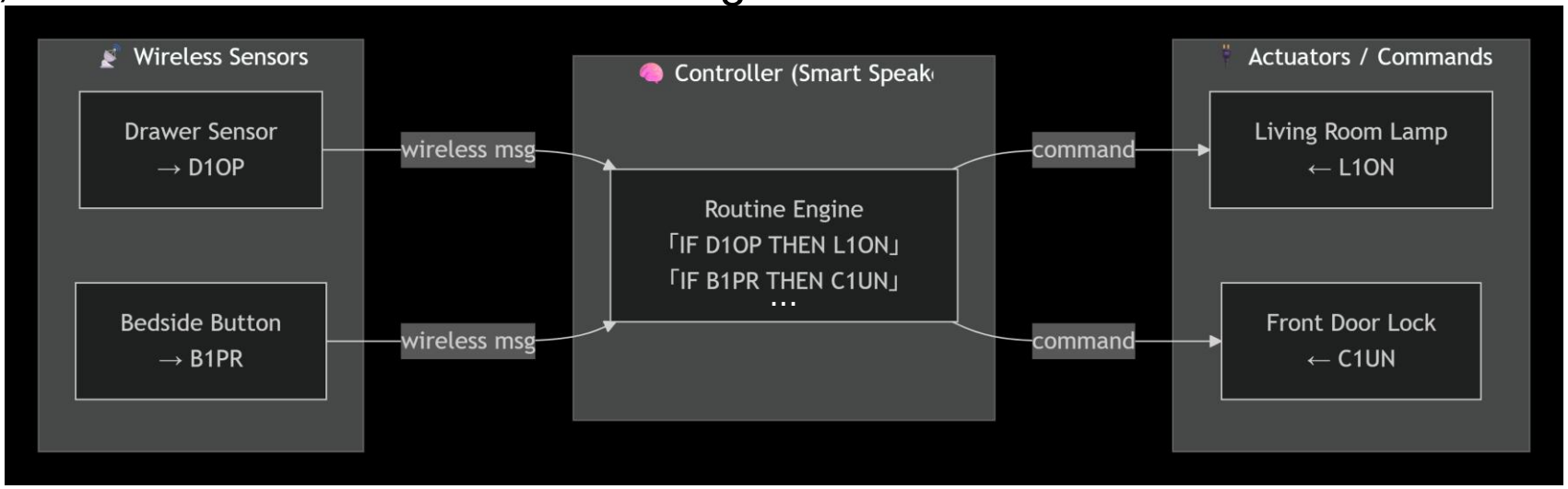
Imagine a company that installs smart home automation kits. Each kit includes small wireless sensors and controllers that respond to simple four-character messages.

For example:

- **D1OP** : A drawer sensor sends this when opened
- **L1ON** : A command that turns on living room lamp
- **B1PR** : Button press detected on bedside remote
- **C1UN** : Command to unlock the front door

The brains of the system is a controller program running on a low-cost device—perhaps a used smart speaker or an old tablet.

It listens for sensor messages, decides what to do based on the homeowner's personalized routine, and sends out command messages.



☑ (*) Design Your Own DSL

Now consider three different homeowners, all using the same basic kit but with wildly different needs:

- **Ms. Chen's morning routine:** When she presses her bedside button, the system should turn on the hallway light, wait 30 seconds, then unlock the front door. If the hallway light doesn't turn on (maybe the bulb is dead), the system should send her a notification instead.
- **Mr. Adebayo's evening routine:** When the sun sets (detected by a light sensor), the system should close the smart blinds, turn on the living room lamp, and set the thermostat to 21° C. But only if it's a weekday. On weekends, he wants the blinds to stay open and the lights to come on gradually.
- **Dr. Kowalski's security routine:** When she opens the drawer containing her valuables, the system should lock all exterior doors immediately. However, if she first opens a specific cabinet (where she keeps the "disarm" key fob) within 10 seconds before opening the drawer, the doors should *not* lock.

> All three homes use identical sensors, lamps, locks, and blinds. The only difference is the sequence, timing, and conditional logic that ties them together.

Please **design a small DSL** (maybe call it HomeScript) that lets homeowners describe their routines in plain, structured language. A compiler then translates that description into the low-level four-character commands the hardware understands.

(*) Reference HomeScript (by deepseek-r1)

Ms. Chen's morning routine

```
routine "Morning Unlock" {
    trigger: button "bedside" pressed

    action: light "hallway" on
    wait 30 seconds

    if light "hallway" is on {
        door "front" unlock
    } else {
        notify "Hallway light failure"
    }
}
```

Mr. Adebayo's evening routine

```
routine "Sunset Comfort" {
    trigger: light_sensor "outdoor" becomes dark

    if today is weekday {
        blinds "living" close
        light "living" on
    } else {
        light "living" dim over 10 minutes
    }
    thermostat set 21
}
```

What about the remaining security routine?