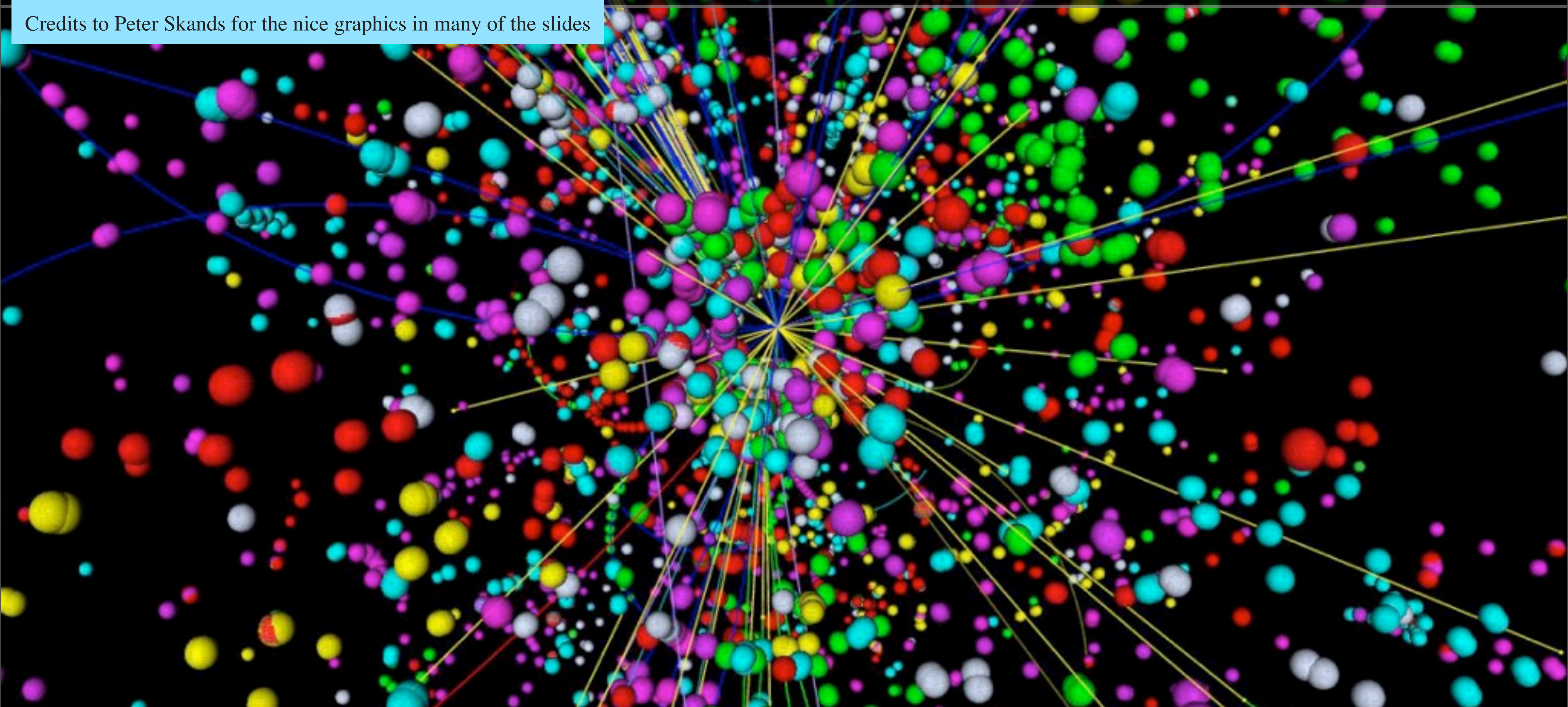
Documenting VINCIA through activity diagrams

CERN, LHCPhenoNet annual meeting, Dec 2013

Juan José López Villarejo (CEA-Saclay)

Credits to Peter Skands for the nice graphics in many of the slides

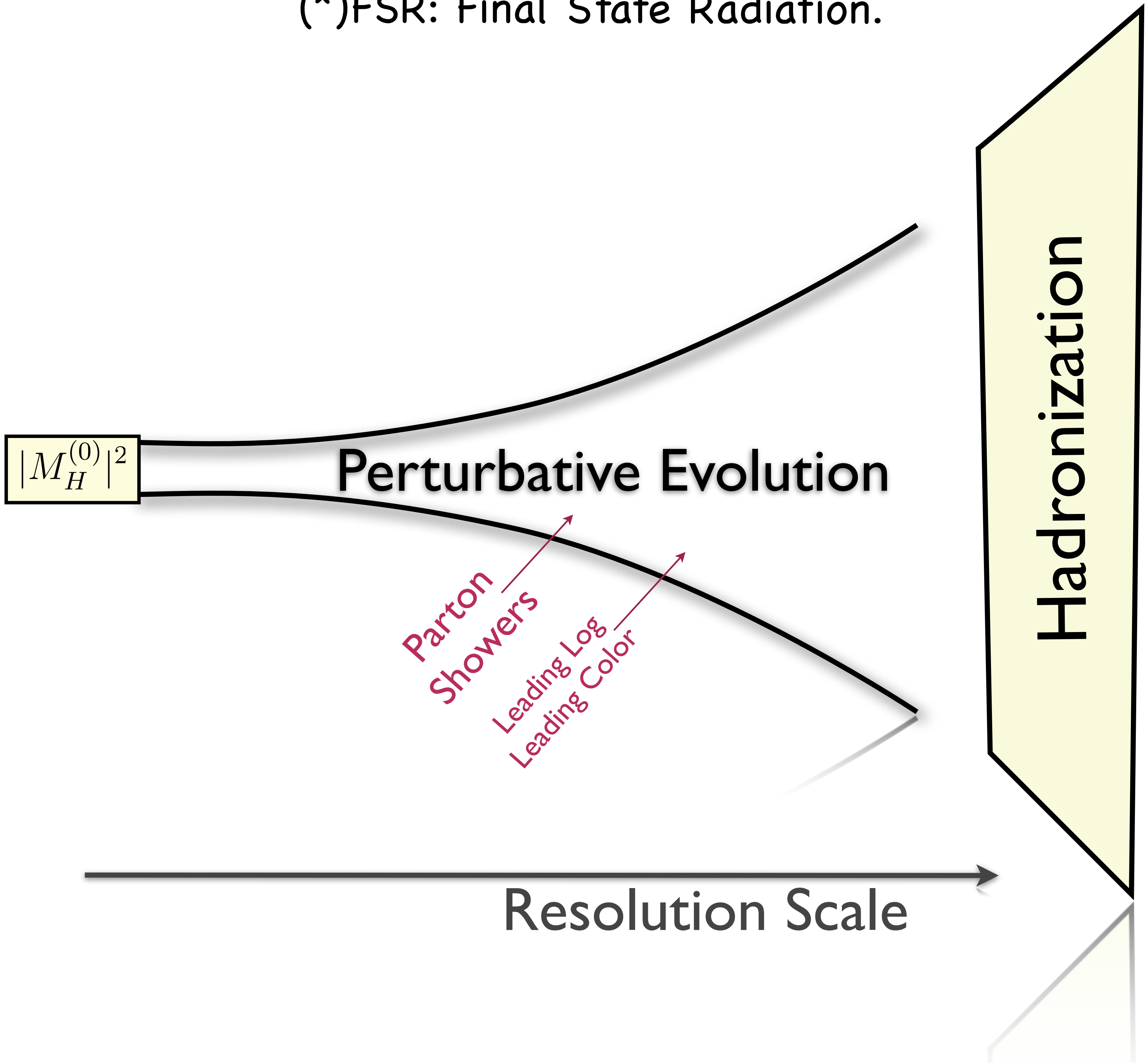


VINCIA collaboration: P. Skands, W. Giele, D. Kosower,
A. Larkoski, J. Lopez-Villarejo; A. Gehrmann-de-Ridder, M. Ritzmann; E. Laenen, L. Hartgring

Modern
Parton Showers

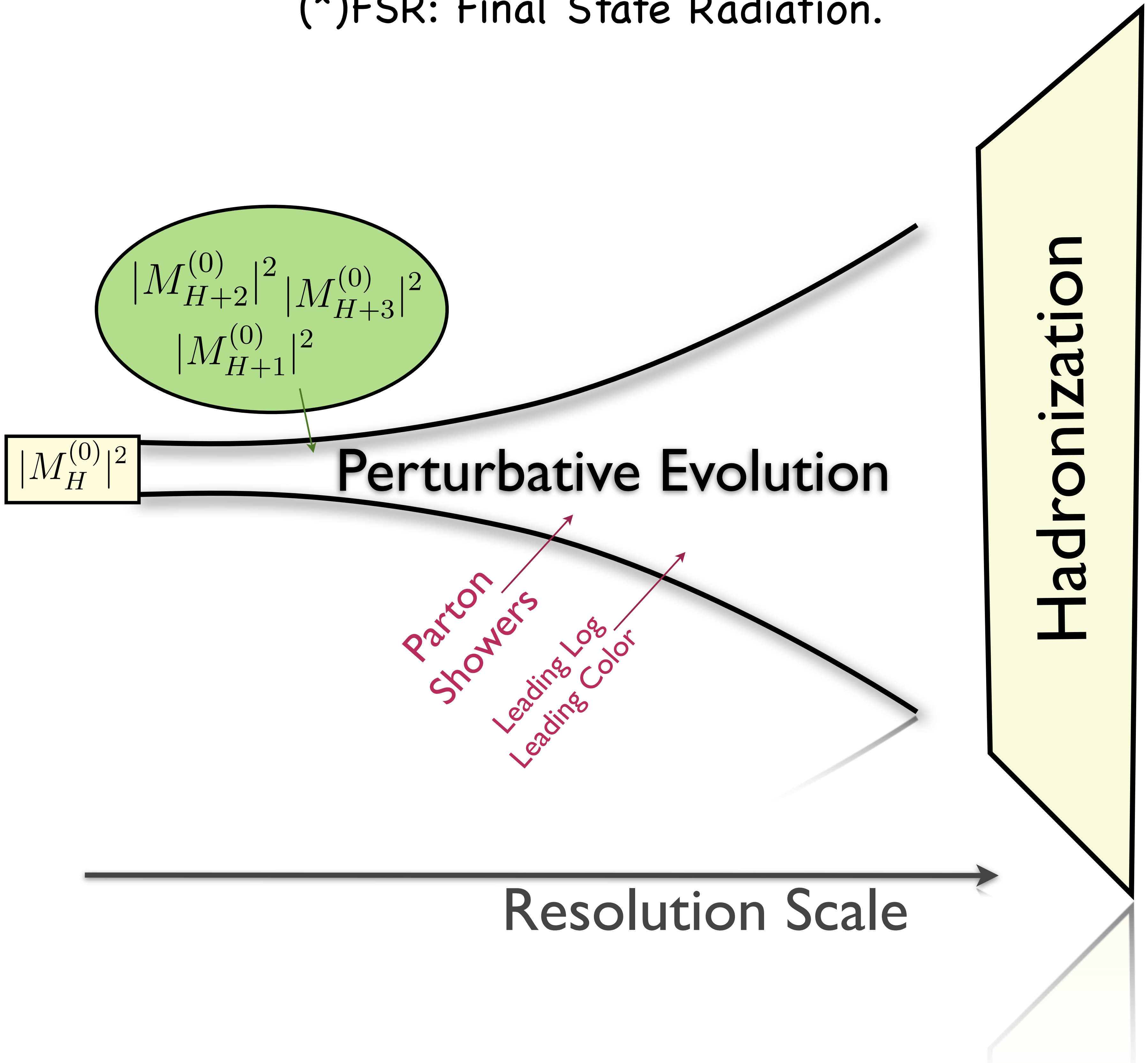
Parton shower evolution (*FSR)

(*)FSR: Final State Radiation.



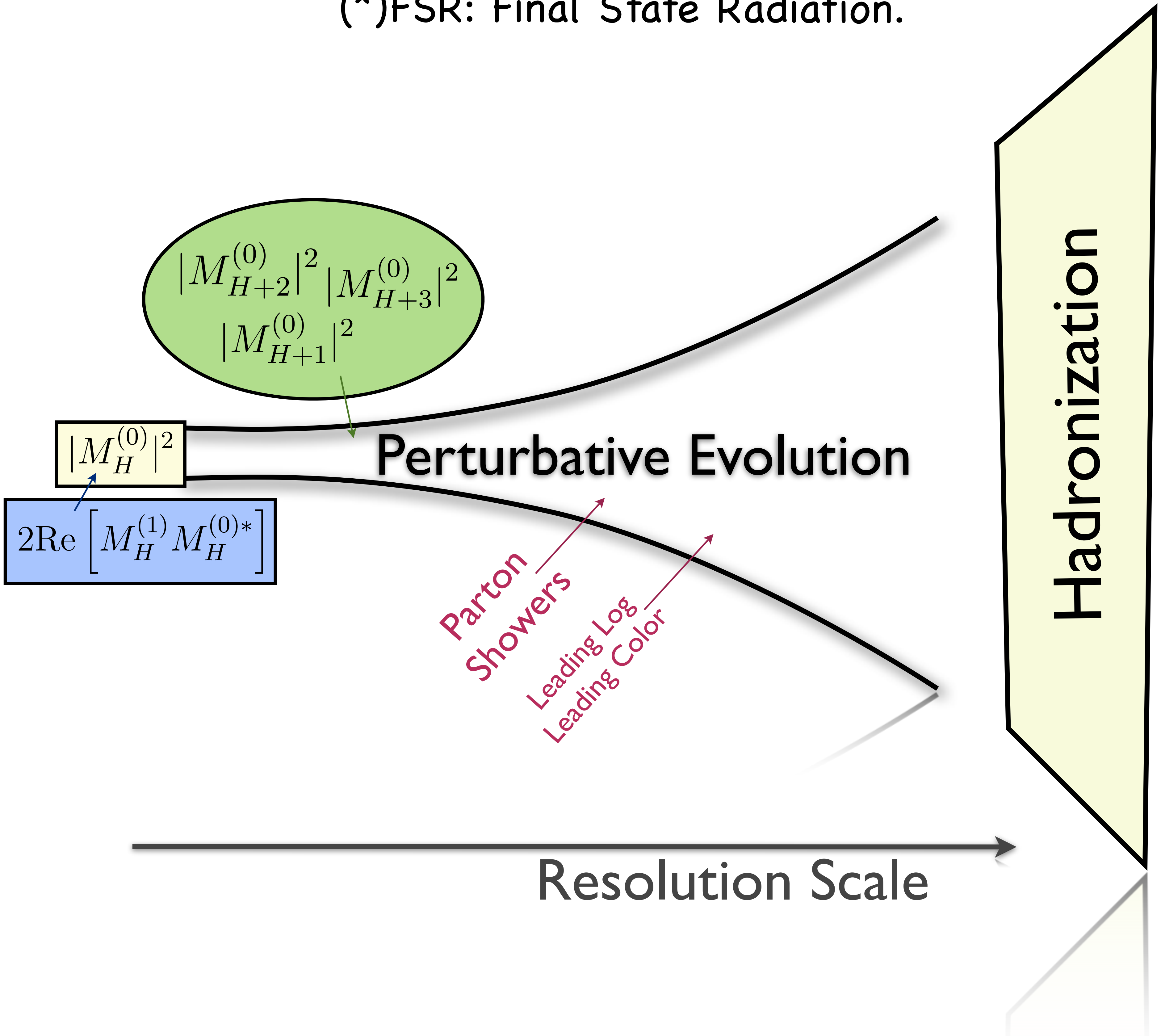
Parton shower evolution (*FSR)

(*)FSR: Final State Radiation.



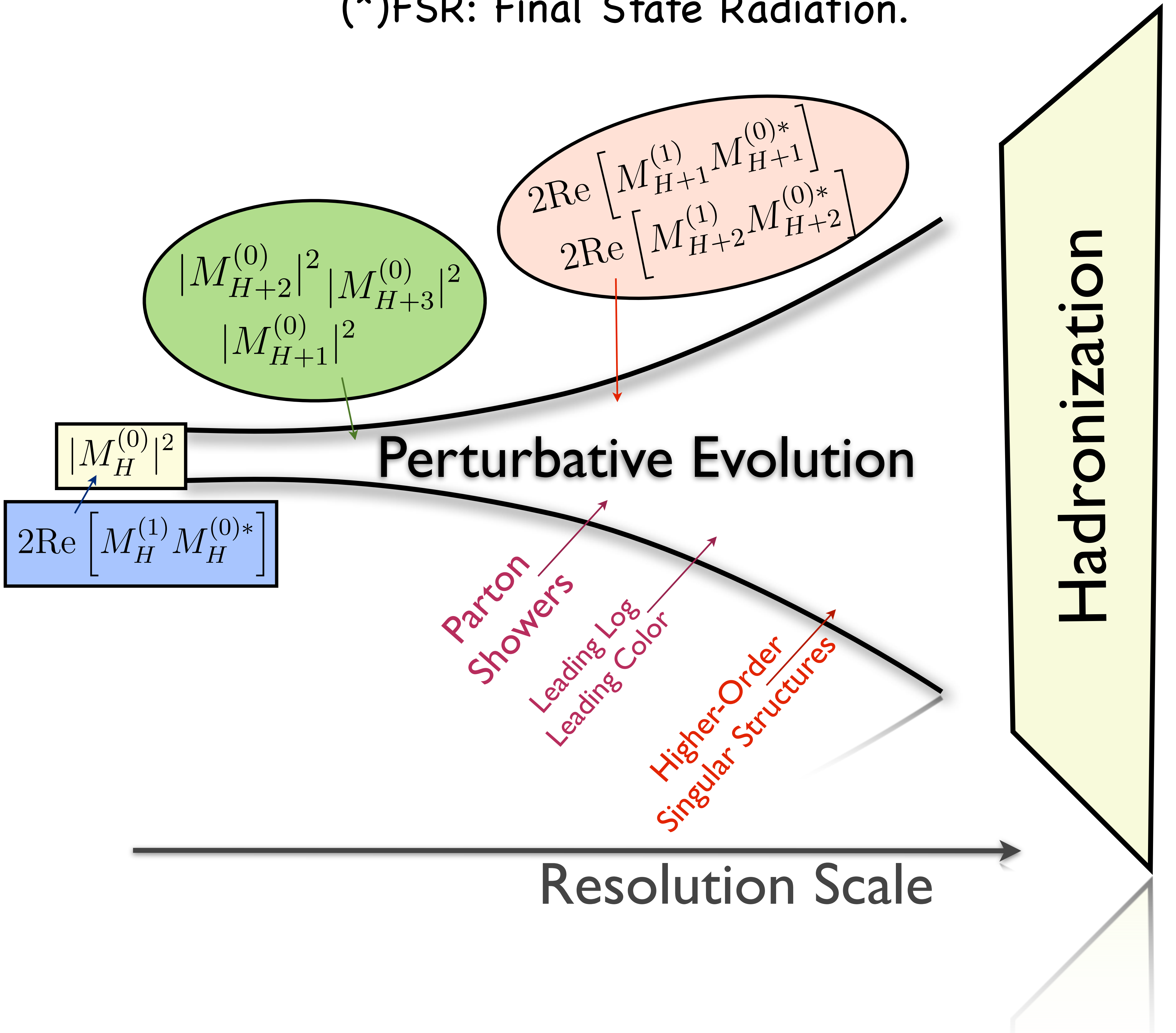
Parton shower evolution (*FSR)

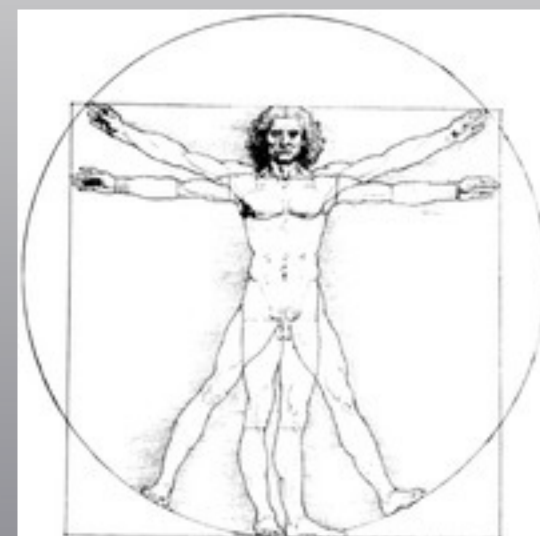
(*)FSR: Final State Radiation.



Parton shower evolution (*FSR)

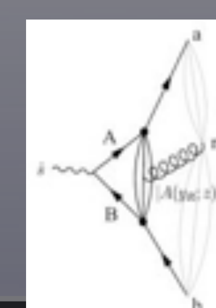
(*)FSR: Final State Radiation.



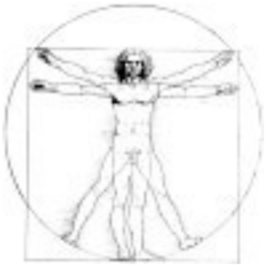


VINCIA

from «the creators» of 

and the  antenna formalism

VINCIA



What is it?

Plug-in to PYTHIA 8 http://vincia.hepforge.org

What does it do?

"(Multiplicatively) Matched Markov antenna showers"

Antenna: VINCIA uses antennae, instead of Altarelli-Paresi splitting kernels.
Markov: markovian condition for the shower; no memory of the path.
Multiplicative matching to exact Matrix Elements.

Extensive *(and automated)* uncertainty estimates

Systematic variations of shower functions, evolution variables, μ_R , etc.

→ *A vector of output weights for each event (central value = unity = unweighted)*

Who is doing it?

Giele, Kosower, Skands (GKS), initiators

+ Collaborations with

A. Larkoski, J. Lopez-Villarejo (sector showers, helicity-dependence),

A. Gehrmann-de-Ridder, M. Ritzmann (mass effects, initial-state radiation),

E. Laenen, L. Hartgring (one-loop corrections)

VINCIA

What is it?

Plug-in to PYTHIA 8 http://vincia.hepforge.org

What does it do?

"(Multiplicatively) Matched Markov antenna showers"

Antenna: VINCIA uses antennae, instead of Altarelli-Paresi splitting kernels.
Markov: markovian condition for the shower; no memory of the path.
Multiplicative matching to exact Matrix Elements.

Extensive *(and automated)* uncertainty estimates

Systematic variations of shower functions, evolution variables, μ_R , etc.

→ A vector of output weights for each event (central value = unity = unweighted)

Who is doing it?

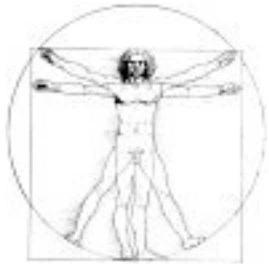
Giele, Kosower, Skands (GKS), initiators

+ Collaborations with

A. Larkoski, J. Lopez-Villarejo (sector showers, helicity-dependence),

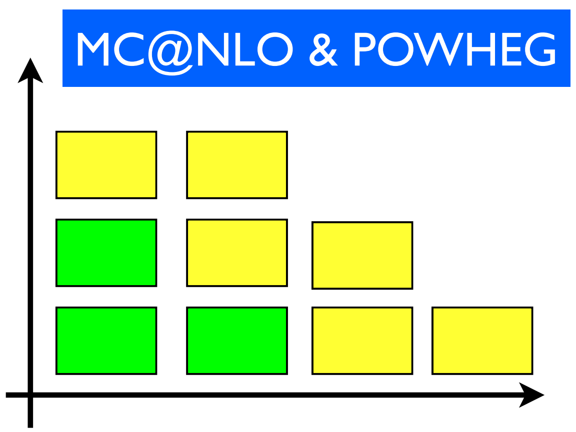
A. Gehrmann-de-Ridder, M. Ritzmann (mass effects, initial-state radiation),

E. Laenen, L. Hartgring (one-loop corrections)

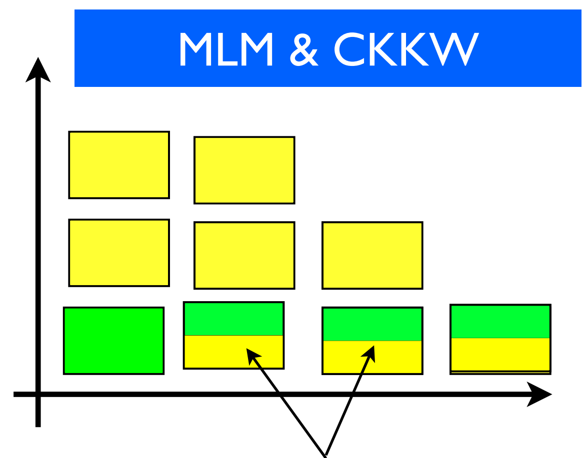


The VINCIA Code

Matching



LO for 1st emission
LL for 2nd emission and beyond



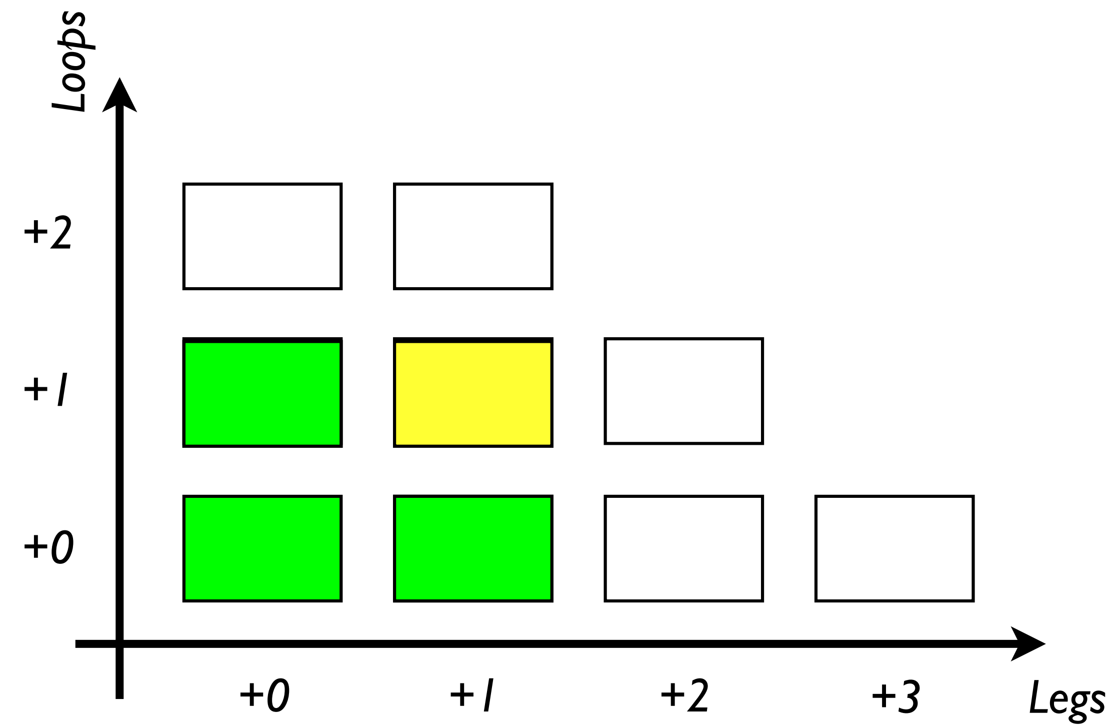
"Matching Scale"
→ hierarchies not matched

Subtraction & Slicing

combine different samples for the same event

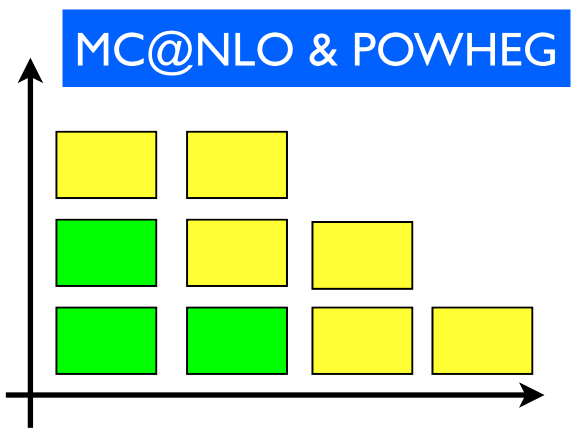
Multiplicative Markov

'on-the-fly' correction

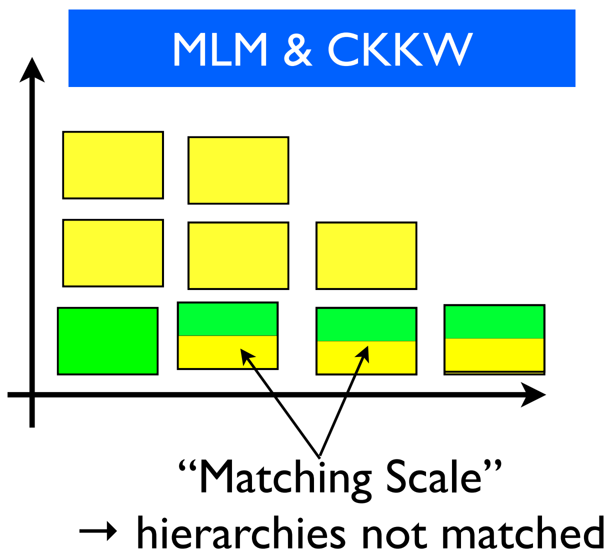


The VINCIA Code + PYTHIA 8

Matching



LO for 1st emission
LL for 2nd emission and beyond

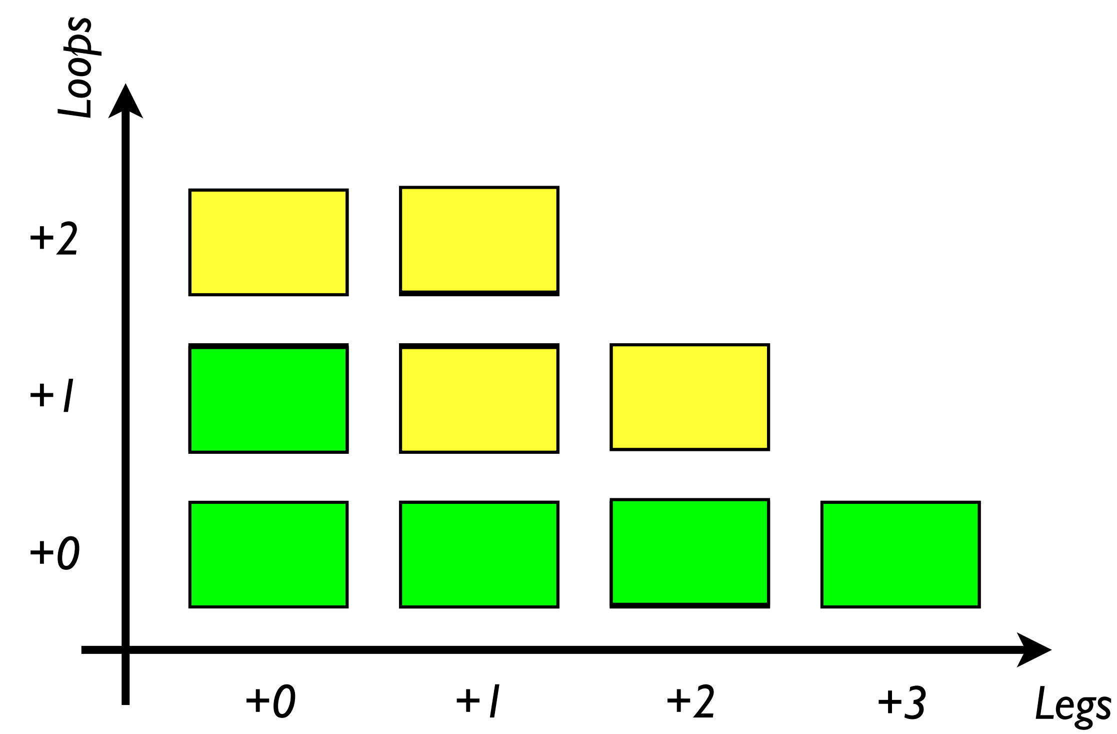


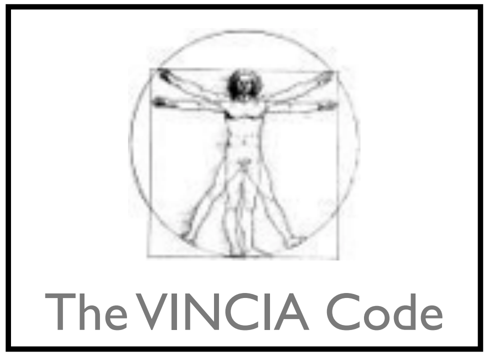
Subtraction & Slicing

combine different samples
for the same event

Multiplicative Markov

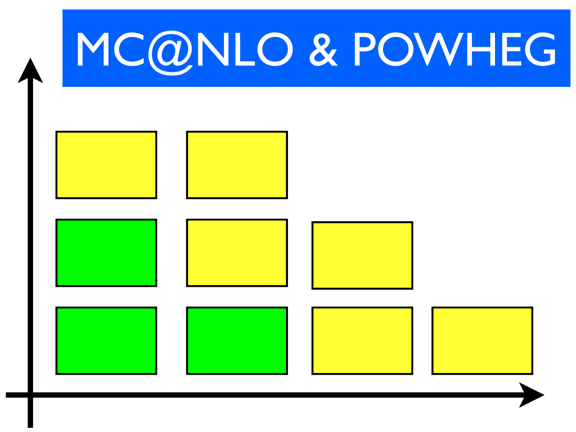
'on-the-fly' correction



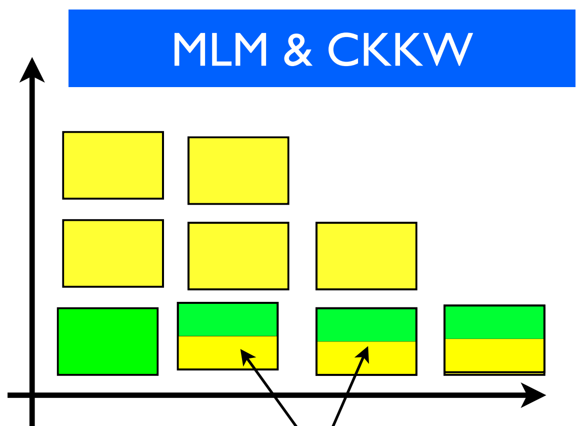


+

Matching



LO for 1st emission
LL for 2nd emission and beyond



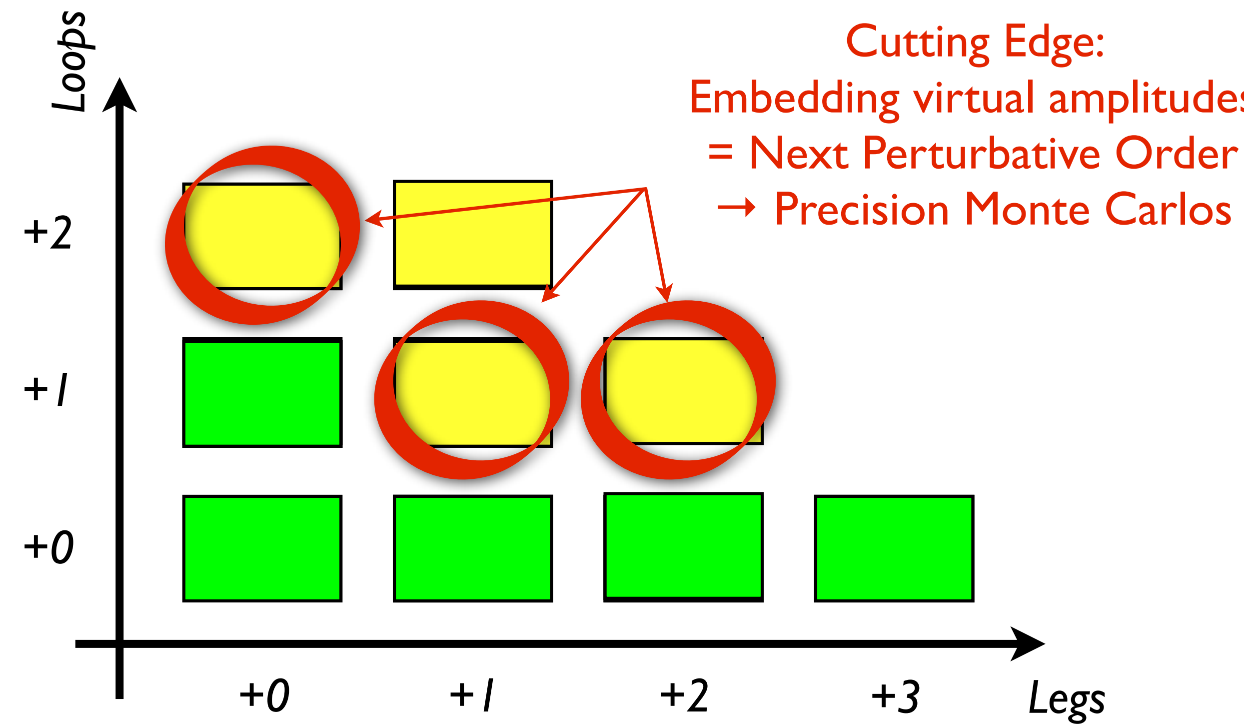
"Matching Scale"
→ hierarchies not matched

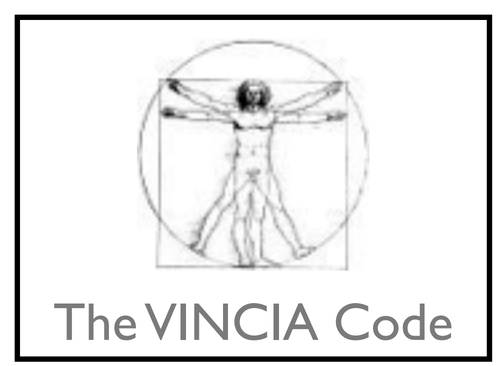
Subtraction & Slicing

combine different samples
for the same event

Multiplicative Markov

'on-the-fly' correction

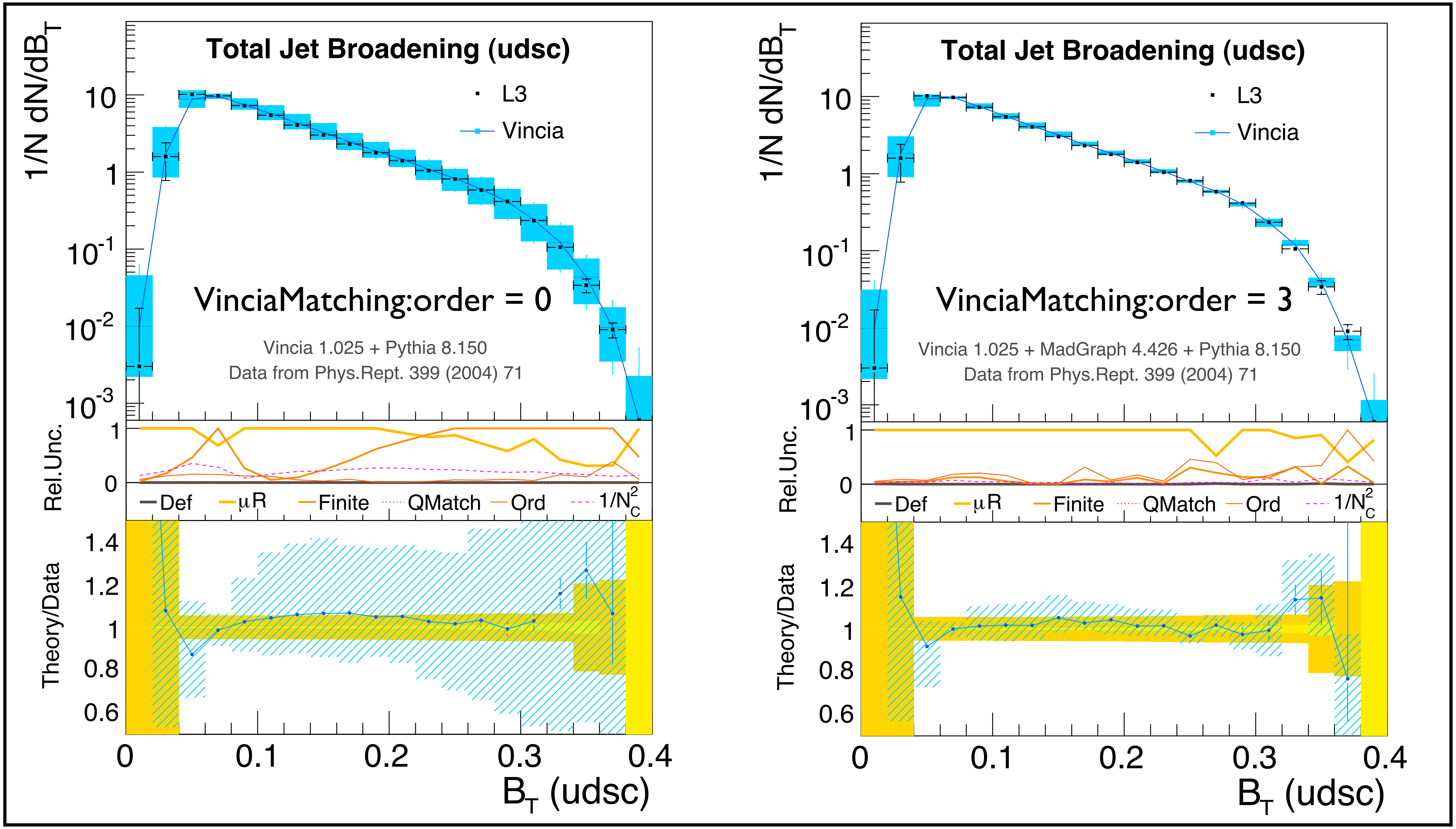




+

Note: other teams working on alternative strategies
with similar goals

Quantifying Precision

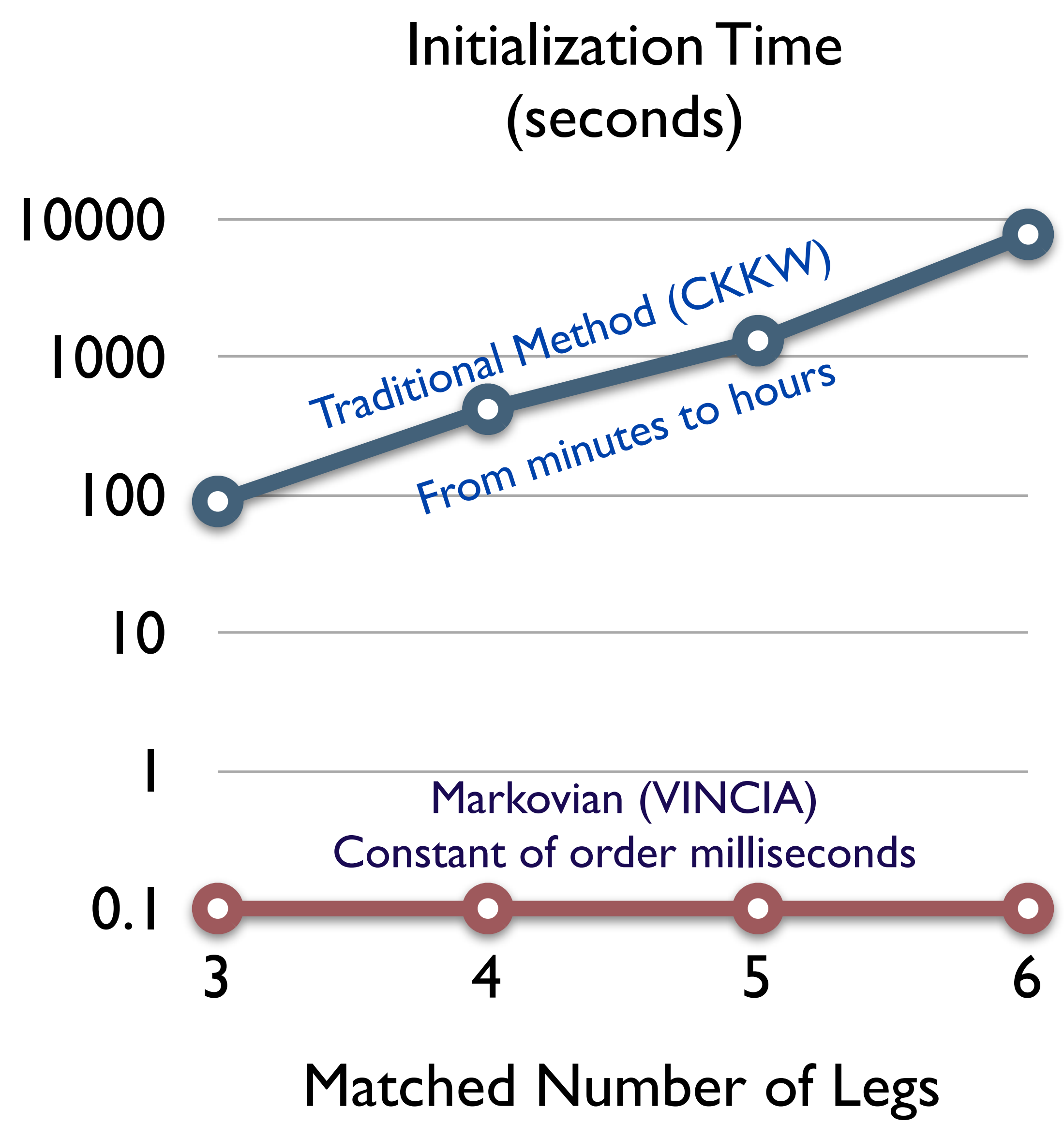


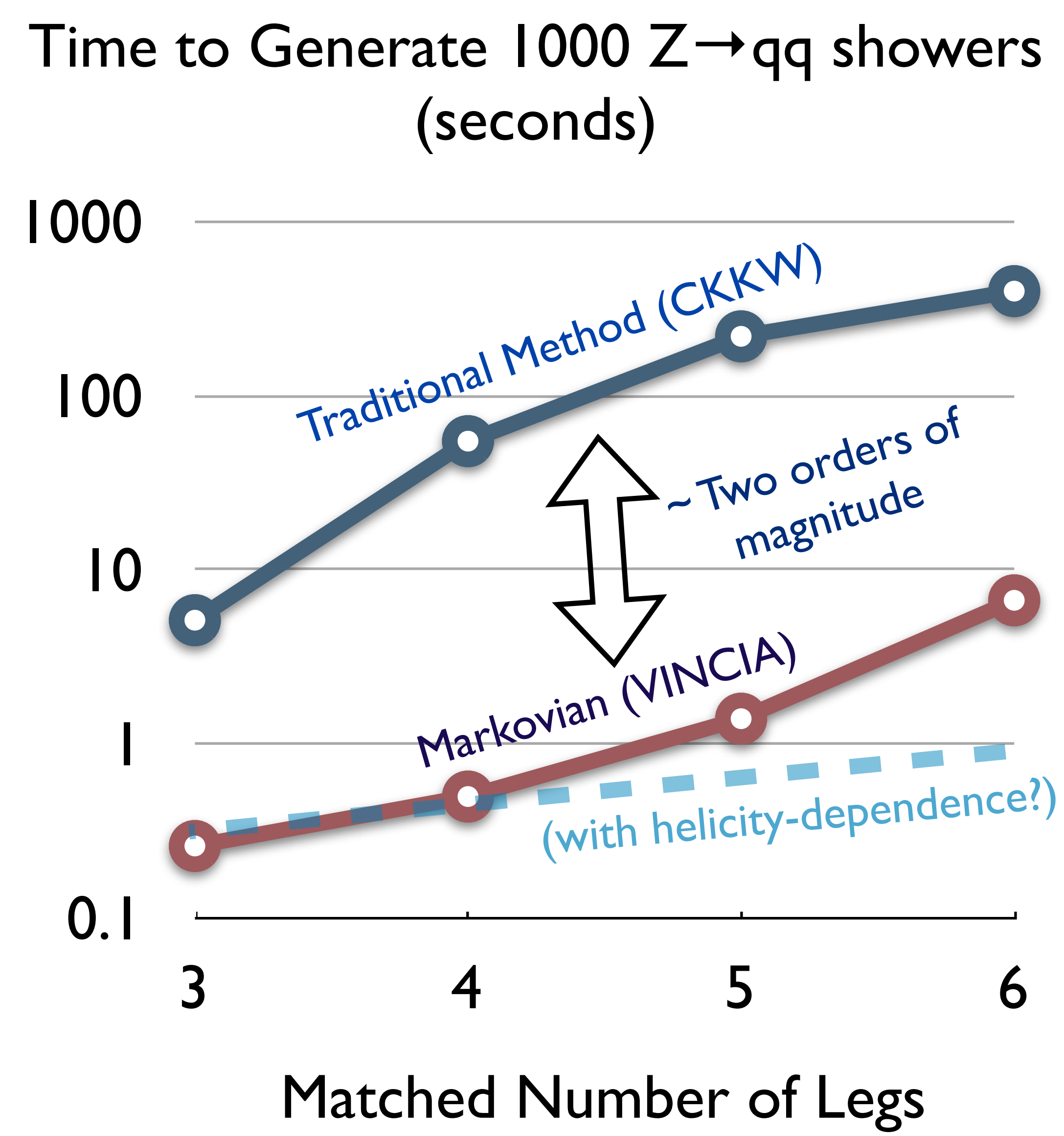
Note: VINCIA so far only developed for final-state radiation (fragmentation)
Initial State under development

SPEED

Efficient Matching with Sector Showers
J. Lopez-Villarejo & PS : JHEP 1111 (2011) 150

(Why we believe Multiplicative Markov is the method of choice for complex problems)





Z→qq *(q=udscb)* + shower. Matched and unweighted. Hadronization off

gfortran/g++ with gcc v.4.4 -O2 on single 3.06 GHz processor with 4GB memory

Generator Versions: Pythia 6.425 *(Perugia 2011 tune)*, Pythia 8.150, Sherpa 1.3.0, Vincia 1.026 *(without uncertainty bands, NLL/NLC=OFF)*

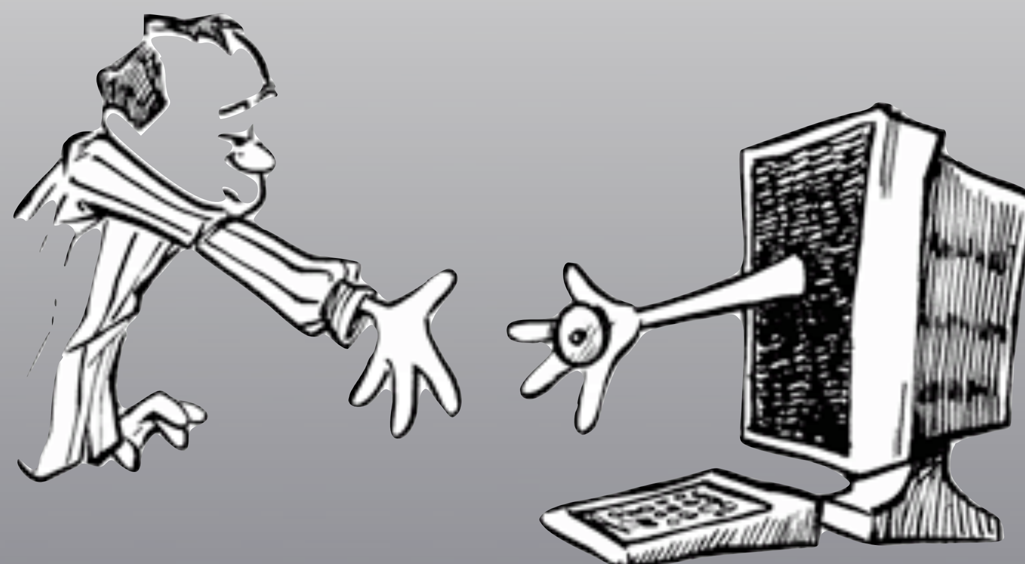
Future prospects

- ISR (ongoing), NLO-multileg matching (ongoing), subleading log shower... Big challenges!
- Extend the team? (computer scientists, theoretical physicists)
- Keep building a free, opensource and transparent code.

Future prospects

- ISR (ongoing), NLO-multileg matching (ongoing), subleading log shower... Big challenges!
- Extend the team? (computer scientists, theoretical physicists)
- Keep building a free, opensource and transparent code.

➡ **Improve documentation for developers.**



Documenting big simulation codes in C++ (*for* *developers*)

J. J. Lopez-Villarejo

Motivation

- Big simulation codes involve collaborations of several authors
  - in distant locations,
  - with different expertise levels.
- New powerful physics features to be added → «pure» physicists.
- Issues with speed and capacity of computers → «pure» programmers.
- Approaching the standards of coding in professional (for-profit) sectors: blueprints first!.
- **What does the code do at a single glance? → transparency**

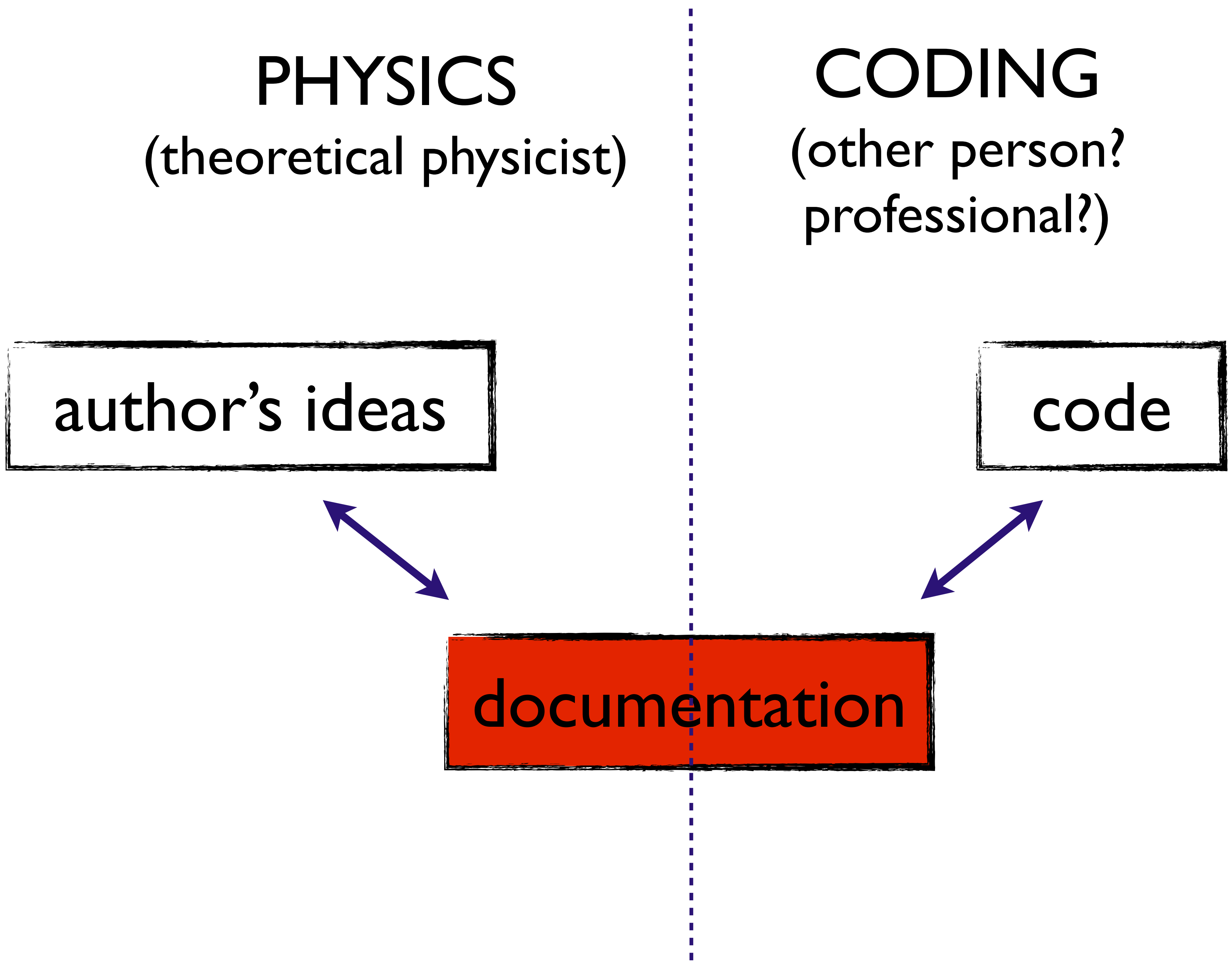
Identified strategies

- Increase modularity (e.g., by effective use of object-orientation). Only one (a few) ‘small’ entities have to be modified each time.
- Improve documentation for authors: an intermediary stage between the physics idea (paper) and the actual realization (code)

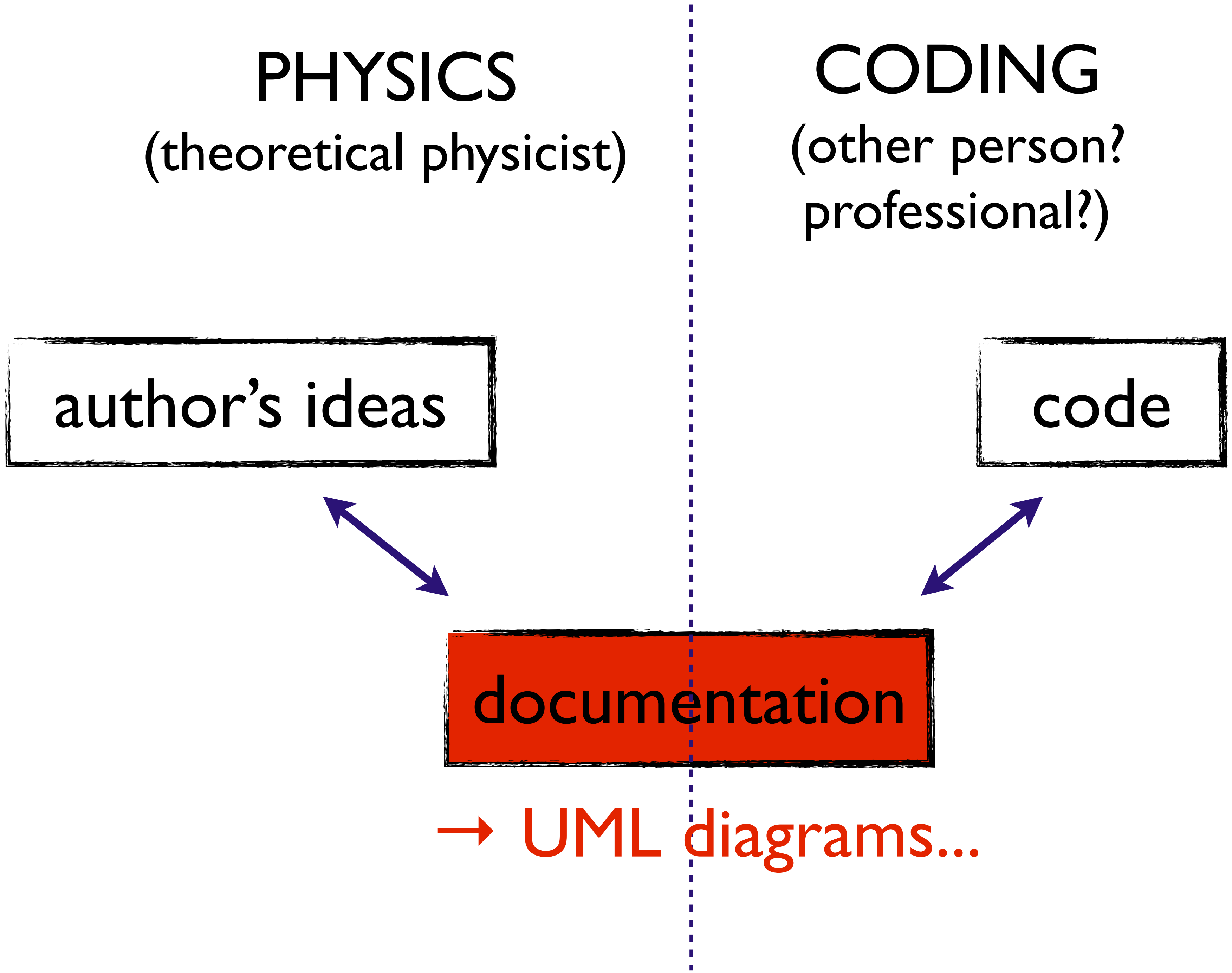
Identified strategies

- Increase modularity (e.g., by effective use of object-orientation). Only one (a few) ‘small’ entities have to be modified each time.
- Improve documentation for authors: an intermediary stage between the physics idea (paper) and the actual realization (code)

The Role of Documentation

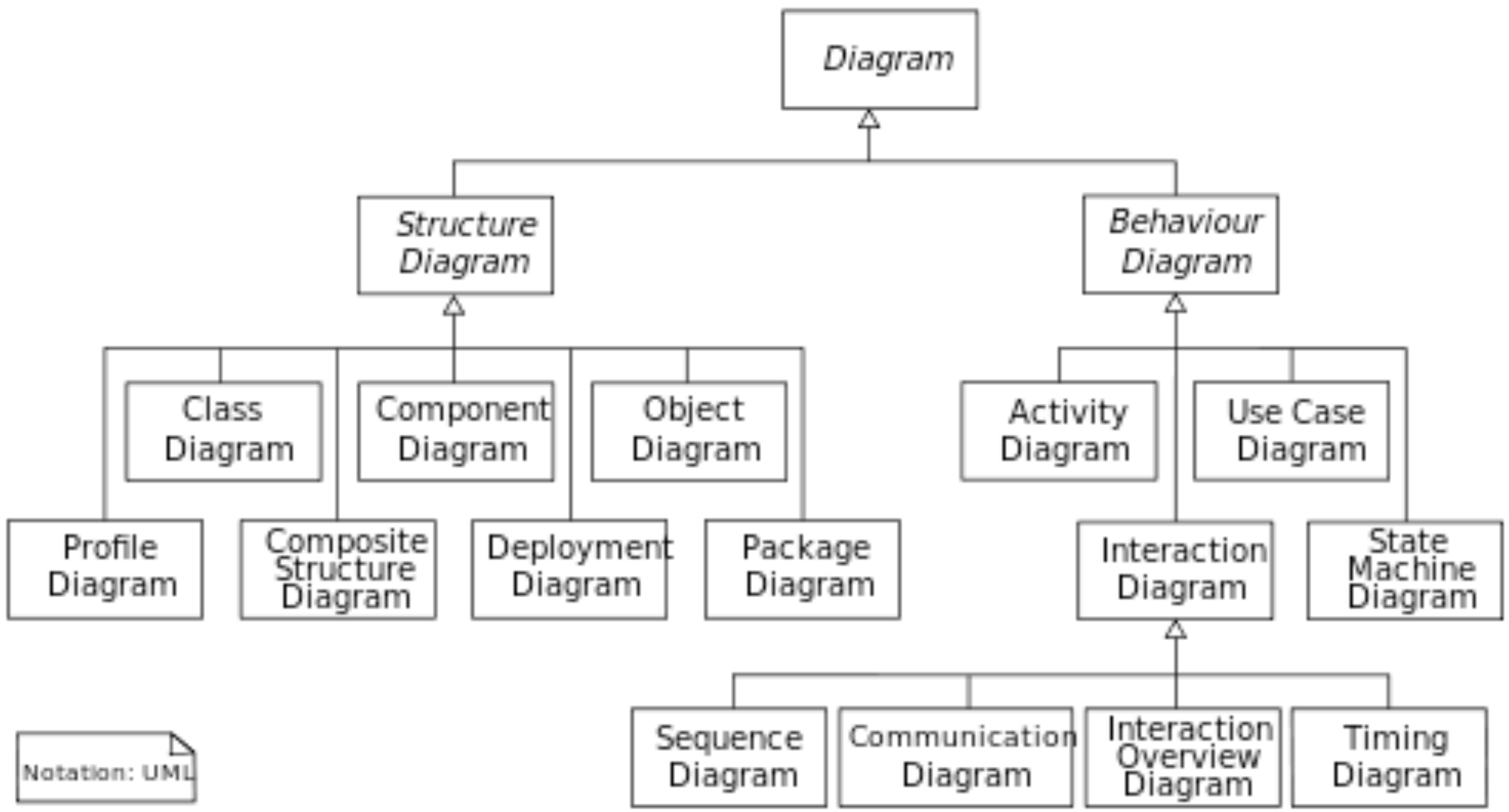


The Role of Documentation



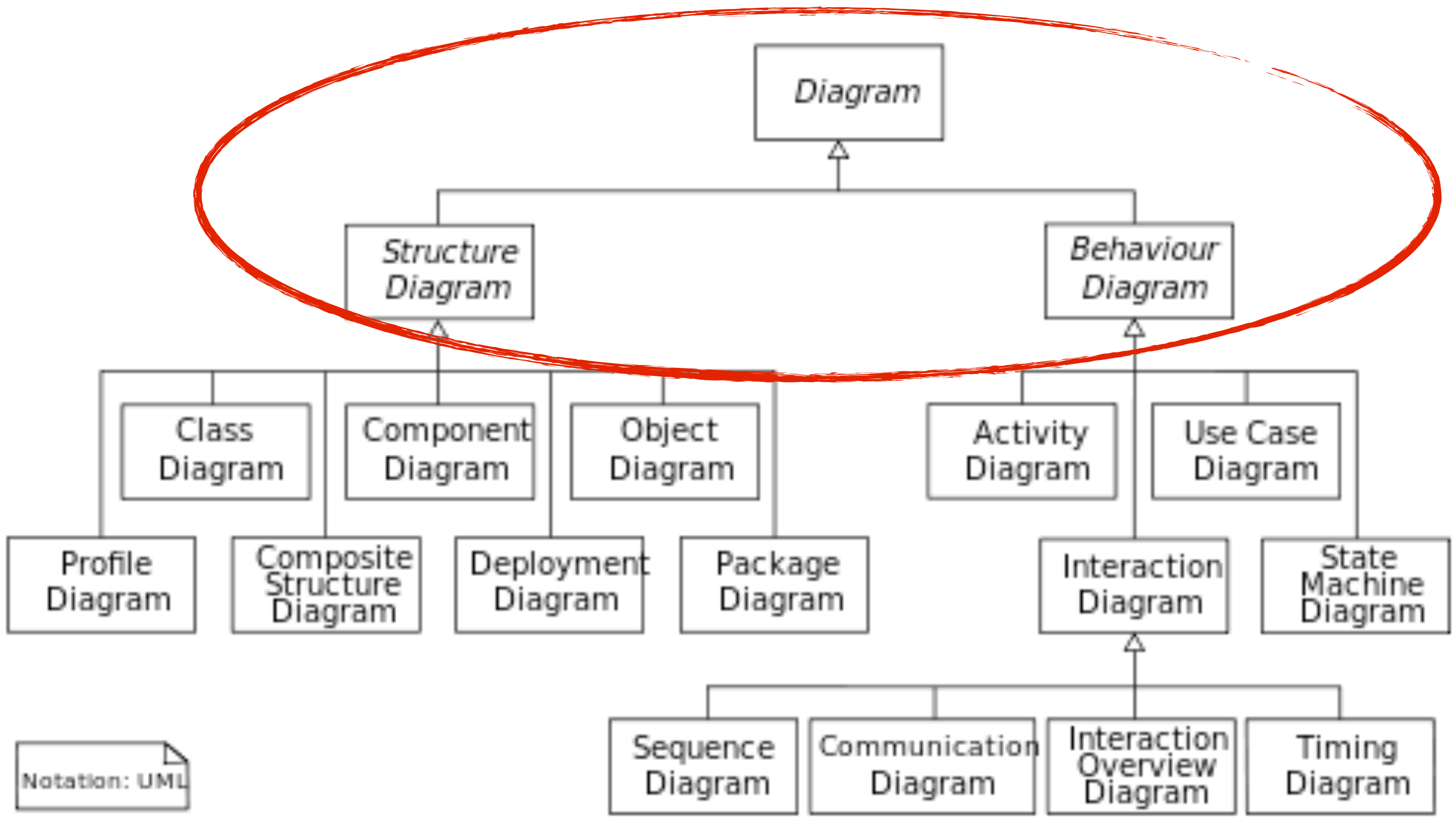
UML Diagrams

The Unified Modeling Language (UML) offers a standard way to visualize a system's architectural blueprints



UML Diagrams

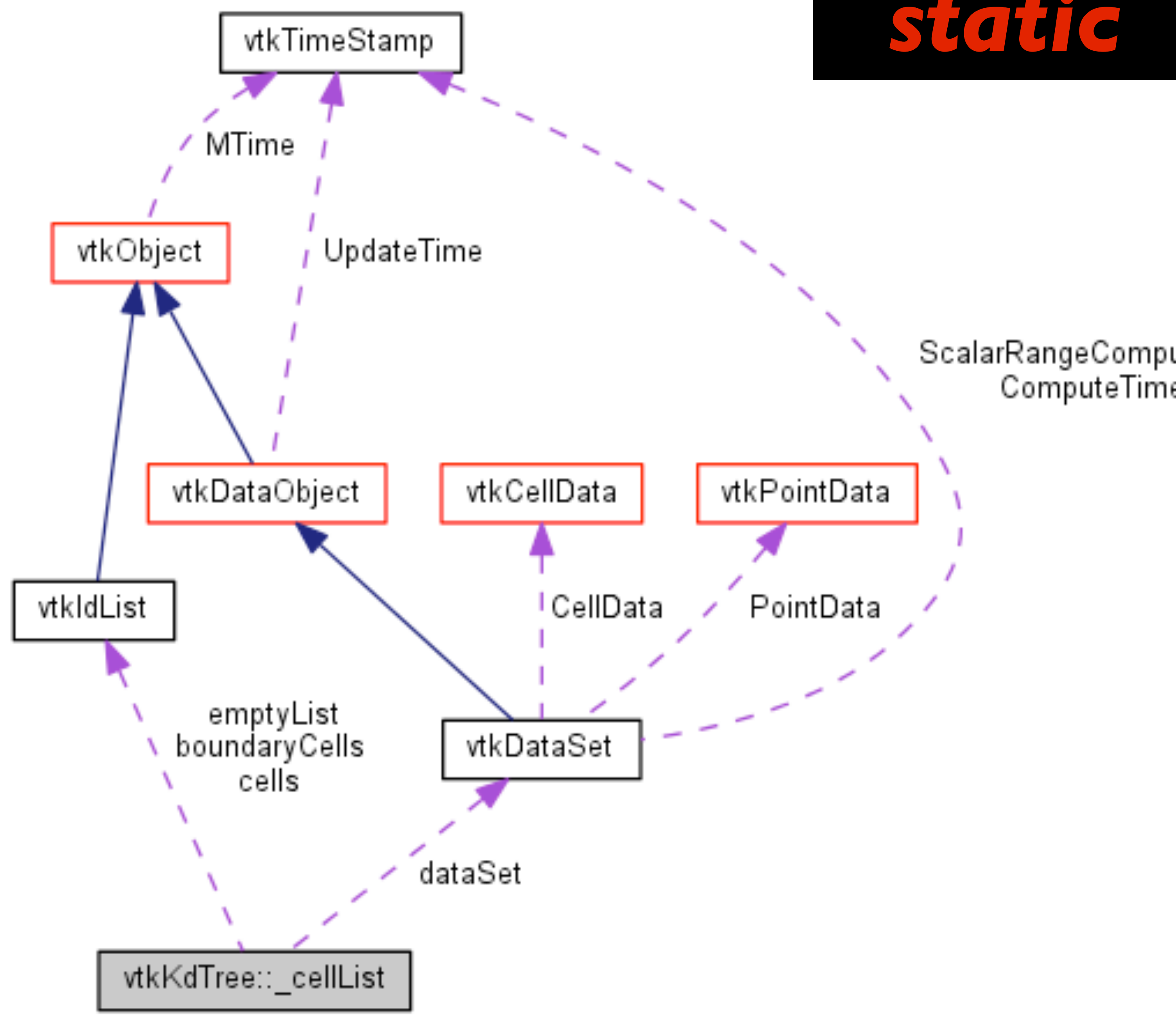
The Unified Modeling Language (UML) offers a standard way to visualize a system's architectural blueprints



UML Structure Diagrams

static

- Doxygen is a free software generating automatically call (structure) diagrams from a C++ code.
- Diagram labels can be modified and extended through comments written directly in the code.



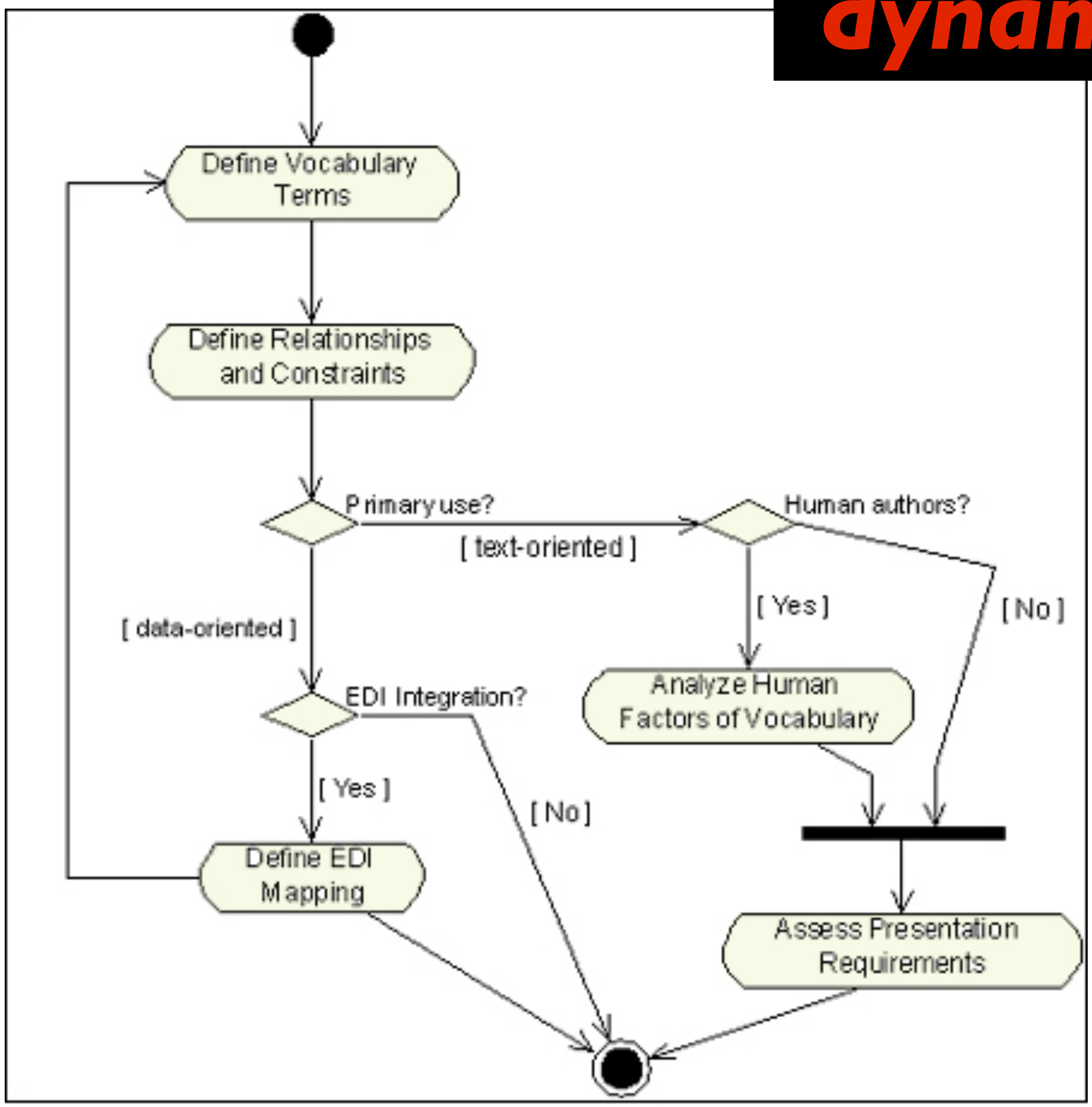
class diagram:
how objects relate to each other?

Doxygen: http://www.doxygen.org/

UML Behavior Diagrams

dynamic!

- Developing free software which will generate automatically high-level workflow diagrams from annotated C++ code.
- User can browse among related activities and zoom in/out (more or less detail).



activity diagram / workflow:
what is the sequence of actions?

Flowgen: public release after testing in VINCIA

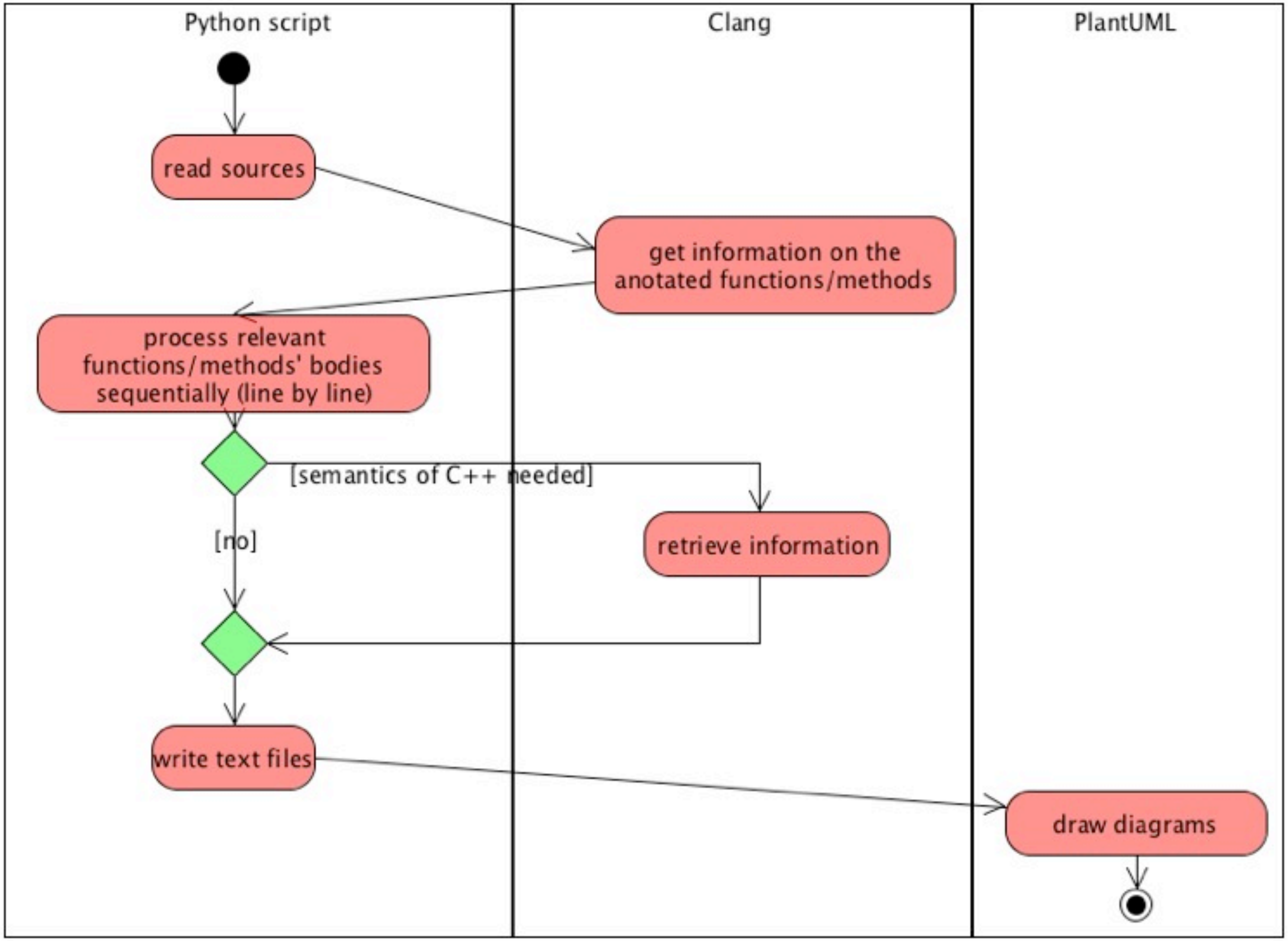
Implementation

Workflow Generator

- A **Python** script controls sequential reading of sources, calls to other elements and writing of the output.
- **Clang** (clang.llvm.org) is used to get the semantics of the source code.
- **PlantUML** (plantuml.sourceforge.net) is used to draw the workflow diagrams.

Flowgen: public release after testing in VINCIA

Workflow generator's workflow



Flowgen: public release after testing in VINCIA

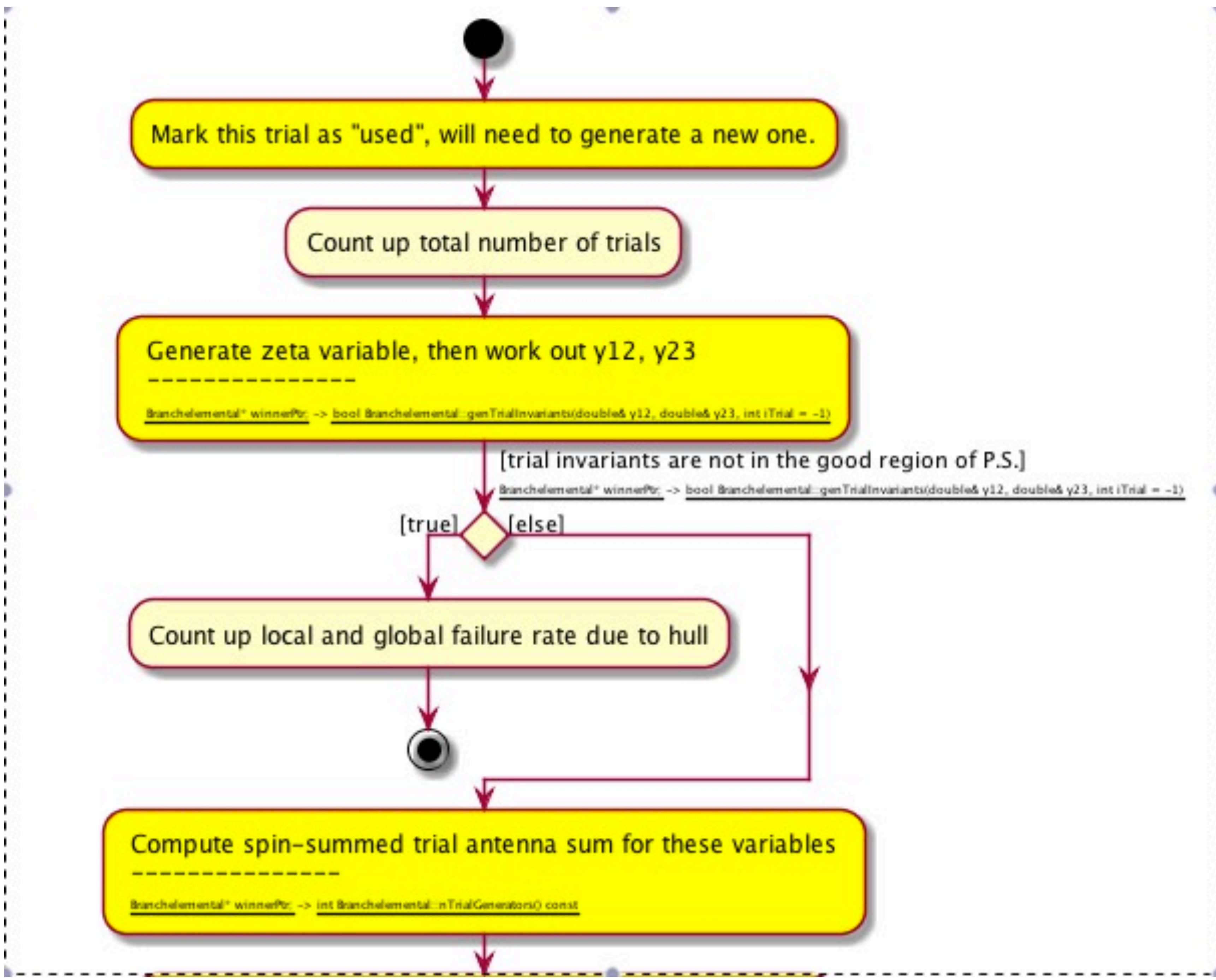
The tool in practice

Flowgen: source input

```
bool VinciaShower::acceptTrial(Event& event) { int iTrial =
winnerPtr->getTrialIndex();
int iAntPhys = winnerPtr->getPhysIndex(iTrial);
bool isSwapped = winnerPtr->getIsSwapped(iTrial);

double qNew = winnerPtr->getTrialScale(iTrial);
double mAnt = winnerPtr->m();
double sAnt = winnerPtr->s();

//$ Mark this trial as "used", will need to generate a new one.
winnerPtr->renewTrial(iTrial);

//$1 Count up total number of trials
++nTrialsSum;
//**********************************************
//$ Generate zeta variable, then work out y12, y23
//**********************************************

double y12, y23;
bool pass = winnerPtr->genTrialInvariants(y12, y23, iTrial);
//$
//$1 [trial invariants are not in the good region of P.S.]
if (! pass) { //$
```

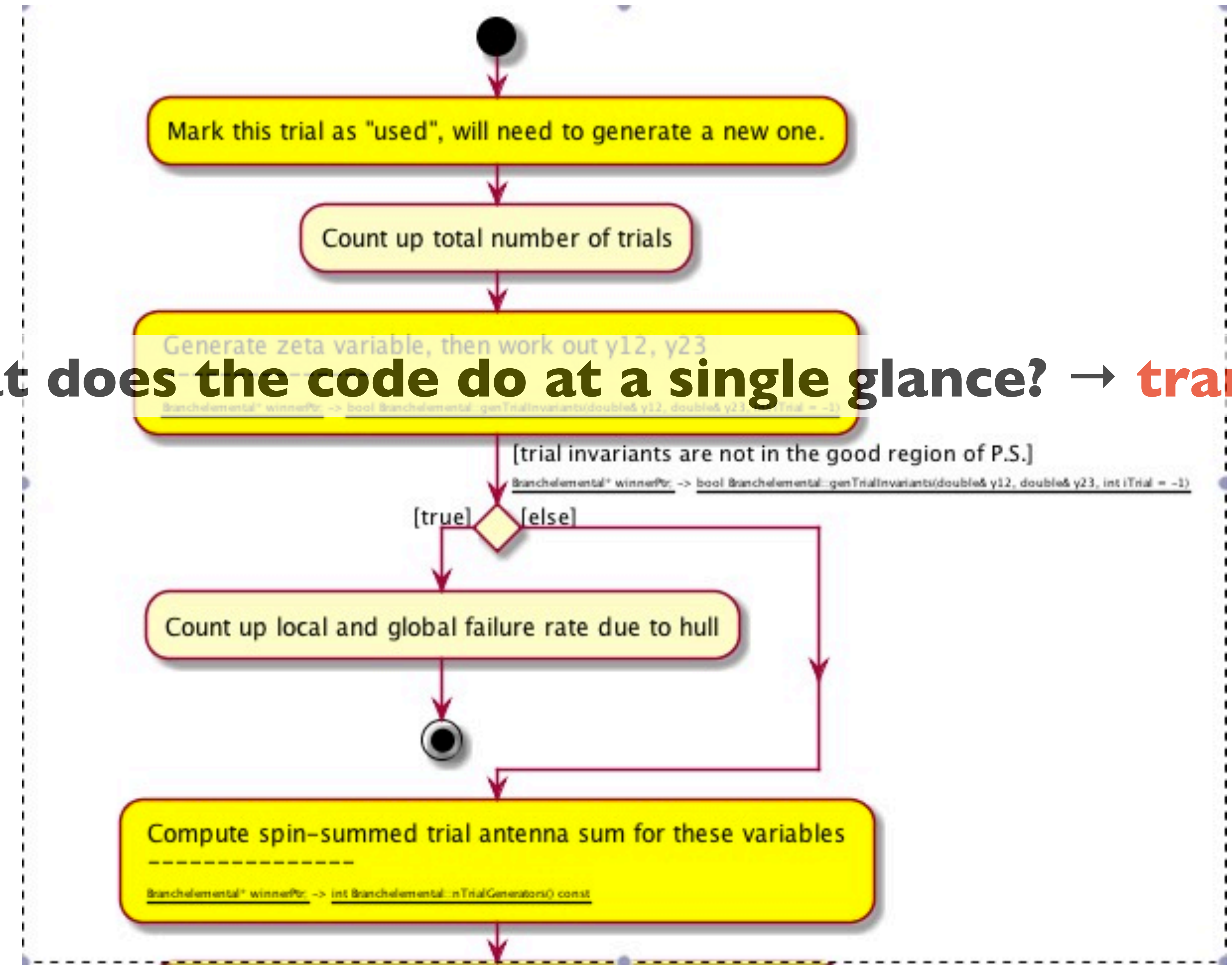
Flowgen: public release after testing in VINCIA

Flowgen: diagram output



Flowgen: public release after testing in VINCIA

Flowgen: diagram output



- **What does the code do at a single glance? → transparency**

Flowgen: public release after testing in VINCIA

Conclusions and Outlook

- We are in the era of precision Monte Carlo parton showers. NLO multileg matching is under development. VINCIA proposes a multiplicative matching approach.
- VINCIA goes for transparency: documenting through high-level workflows of the annotated C++ code.
- Other collaborations may benefit from a similar approach to ours. Flowgen is opensource.

Conclusions and Outlook

- We are in the era of precision Monte Carlo parton showers. NLO multileg matching is under development. VINCIA proposes a multiplicative matching approach.
- VINCIA goes for transparency: documenting through high-level workflows of the annotated C++ code.
- Other collaborations may benefit from a similar approach to ours. Flowgen is opensource.

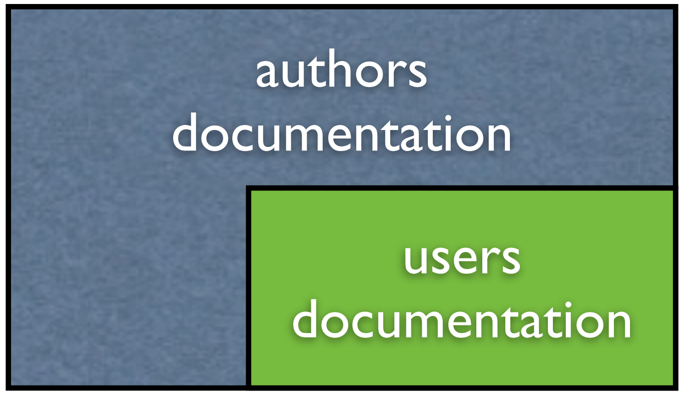
INTERESTED IN THIS PROJECT?

Backup

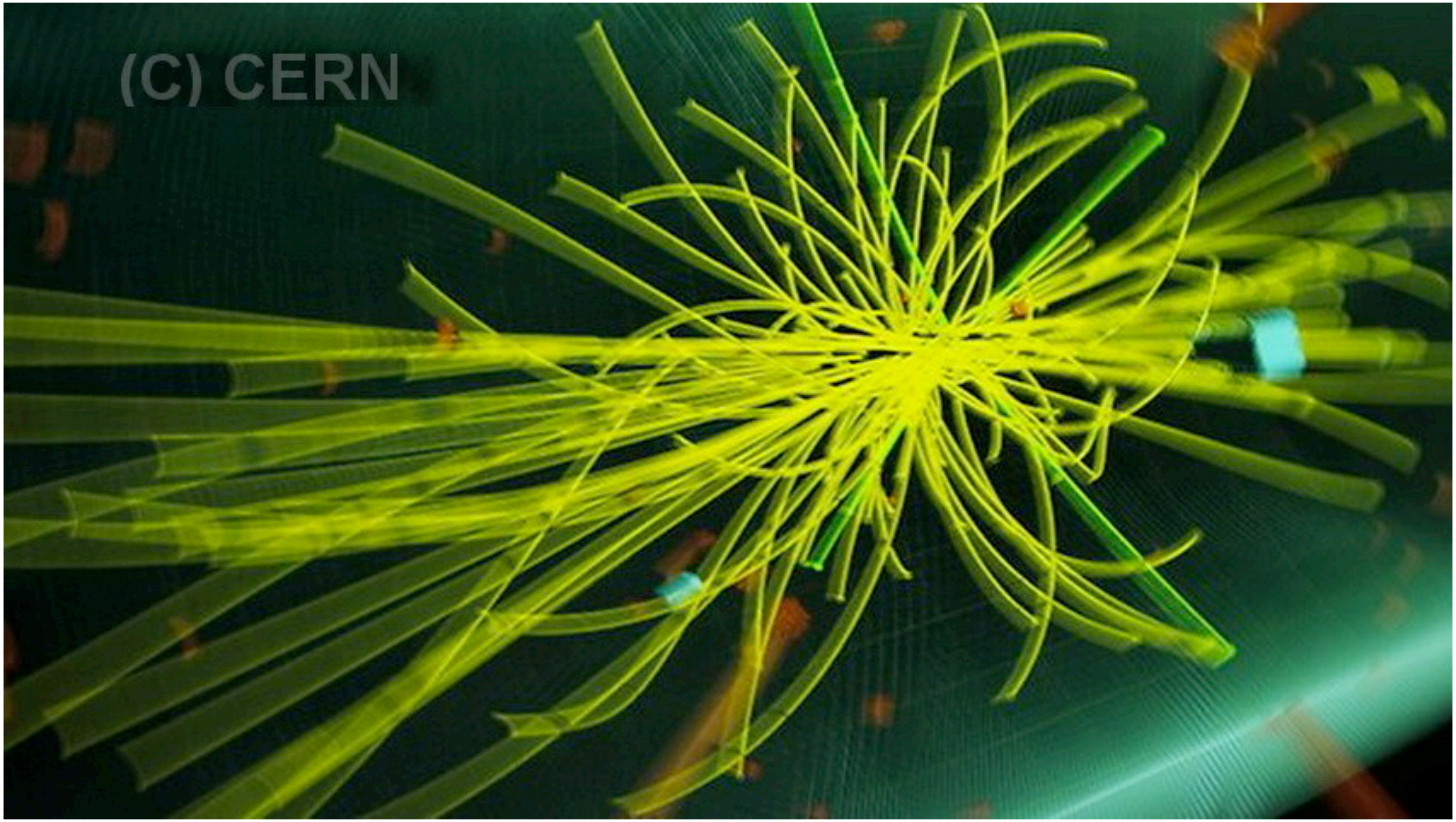
Users documentation

- In VINCIA, keep separate, for the moment.

- A subset of the author's documentation? future possibility.



Flowgen: diagram output



Polarization

Larkoski, Lopez-Villarejo, Skands, arXiv:1301.0933 (2012)

Polarization

Larkoski, Lopez-Villarejo, Skands, arXiv:1301.0933 (2012)

Including helicity information (massless)

Polarization

Larkoski, Lopez-Villarejo, Skands, arXiv:1301.0933 (2012)

Including helicity information (massless)

New spin-dependent antenna functions.

Polarization

Larkoski, Lopez-Villarejo, Skands, arXiv:1301.0933 (2012)

Including helicity information (massless)

New spin-dependent antenna functions.

Advantages:

Polarization

Larkoski, Lopez-Villarejo, Skands, arXiv:1301.0933 (2012)

Including helicity information (massless)

New spin-dependent antenna functions.

Advantages:

- *Treat processes with spin information (observational signatures)*

Polarization

Larkoski, Lopez-Villarejo, Skands, arXiv:1301.0933 (2012)

Including helicity information (massless)

New spin-dependent antenna functions.

Advantages:

- *Treat processes with spin information (observational signatures)*
- Speed gain for matching:

Polarization

Larkoski, Lopez-Villarejo, Skands, arXiv:1301.0933 (2012)

Including helicity information (massless)

New spin-dependent antenna functions.

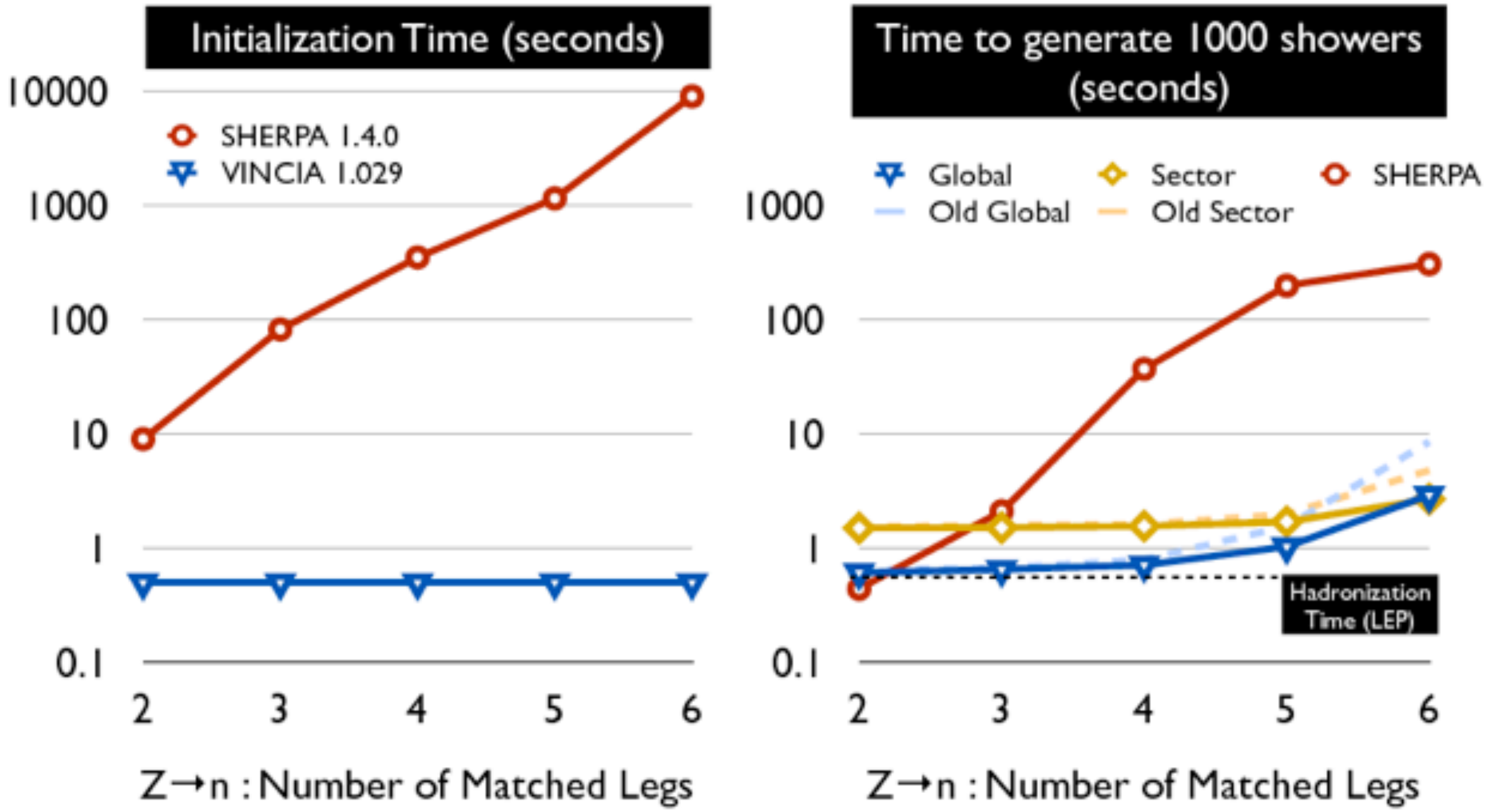
Advantages:

- *Treat processes with spin information (observational signatures)*
- Speed gain for matching:

$$|M_{p1,p2,p3,p4}|^2 = |M_{+,+,+,+}|^2 + |M_{+,+,+,-}|^2 + |M_{+,+,-,-}|^2 + |M_{+,-,-,-}|^2 + \ldots$$

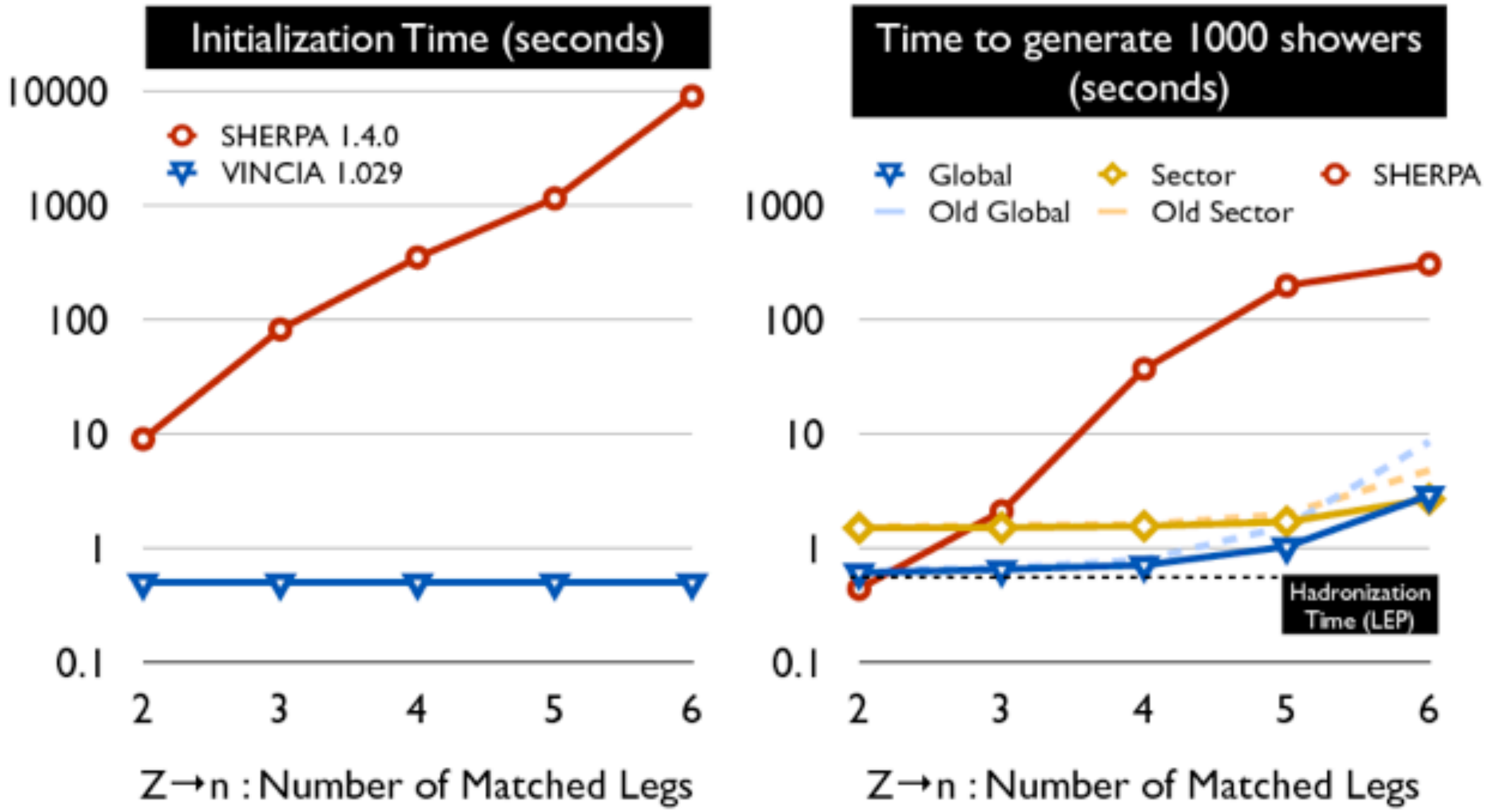
Helicity structures are independent at the level of probabilities

Polarization (speed)



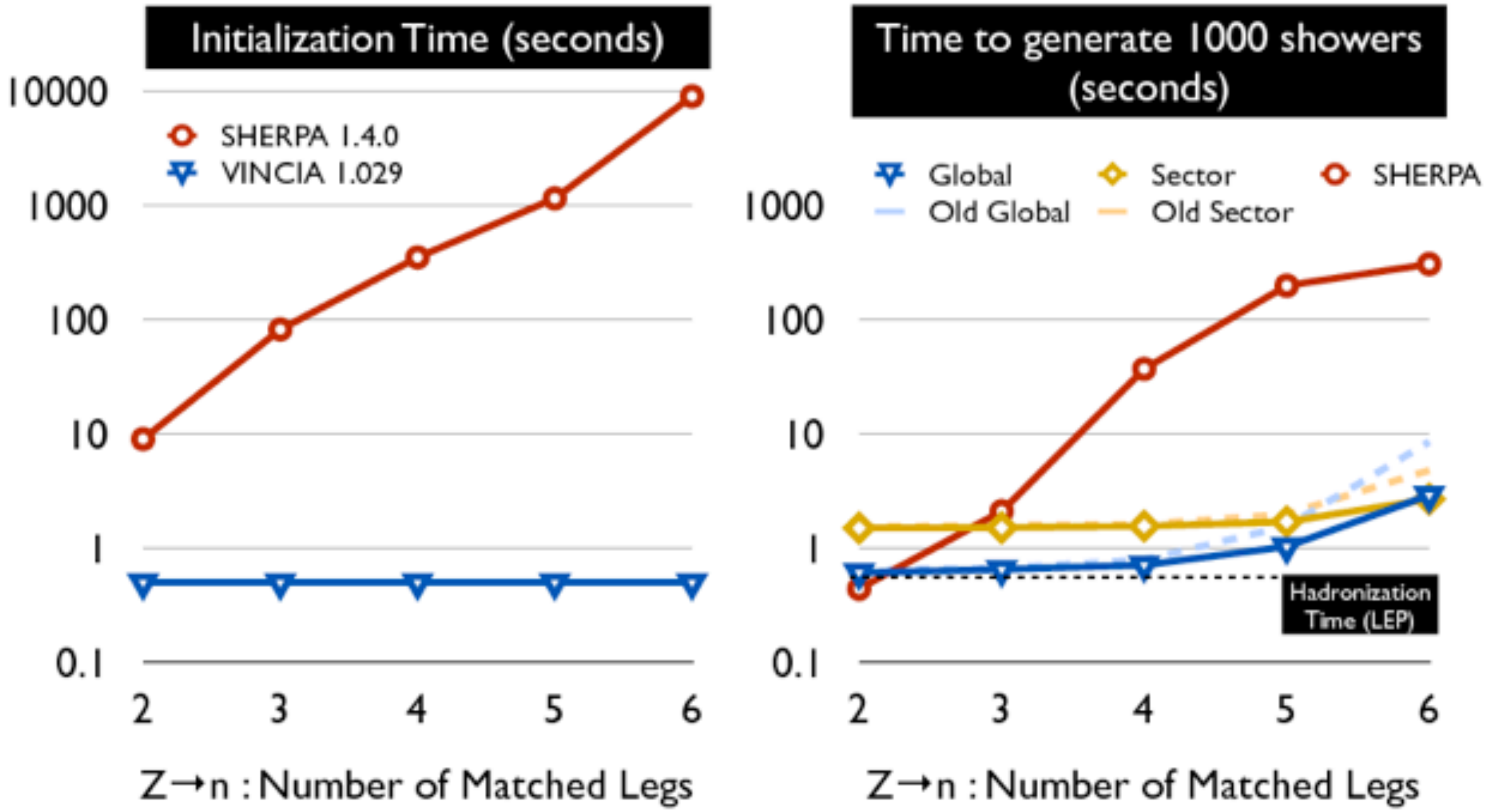
Polarization (speed)

- Speed gain for matching:



Polarization (speed)

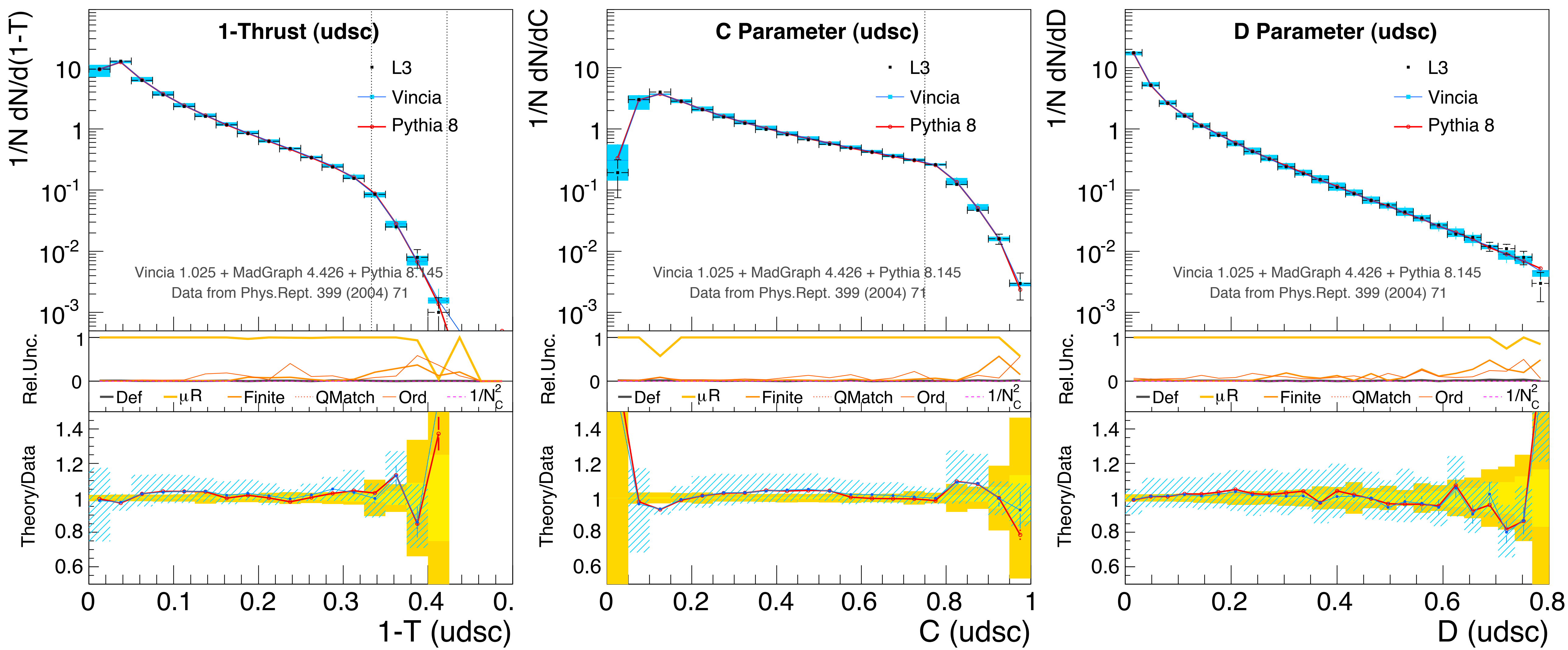
- Speed gain for matching:



$$|M_{p1,p2,p3,p4}|^2 = |M_{+,+,+,+}|^2 + |M_{+,+,+,-}|^2 + |M_{+,+,-,-}|^2 + |M_{+,-,-,-}|^2 + \ldots$$

Helicity structures are independent at the level of probabilities

LEP event shapes



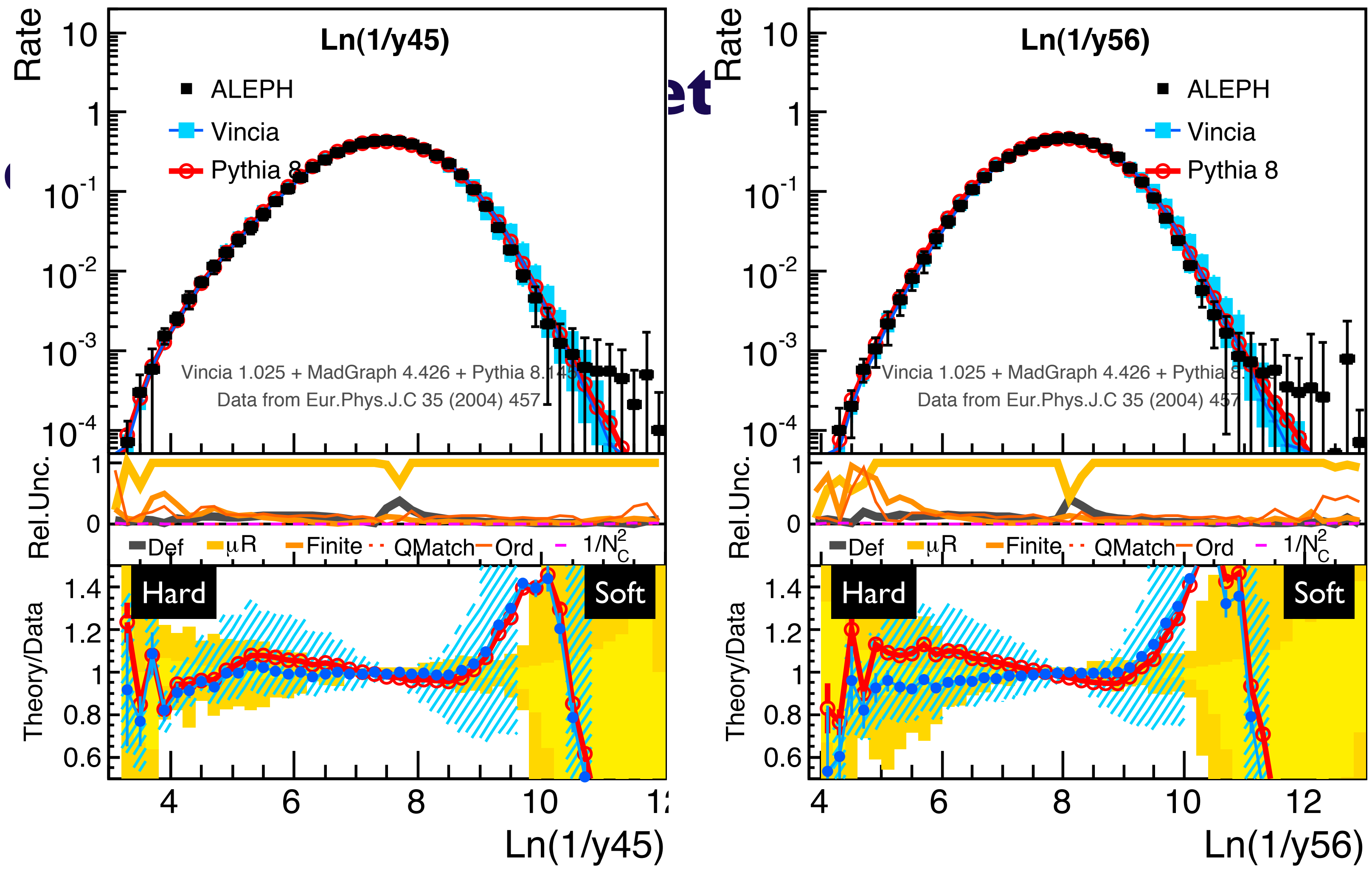
PYTHIA 8 already doing a very good job

VINCIA adds uncertainty bands + can look at more exclusive observables?

Multijet resolution scales

y_{45} = ... **et** ... ~

“scal



4-Jet Angles

4-jet angles

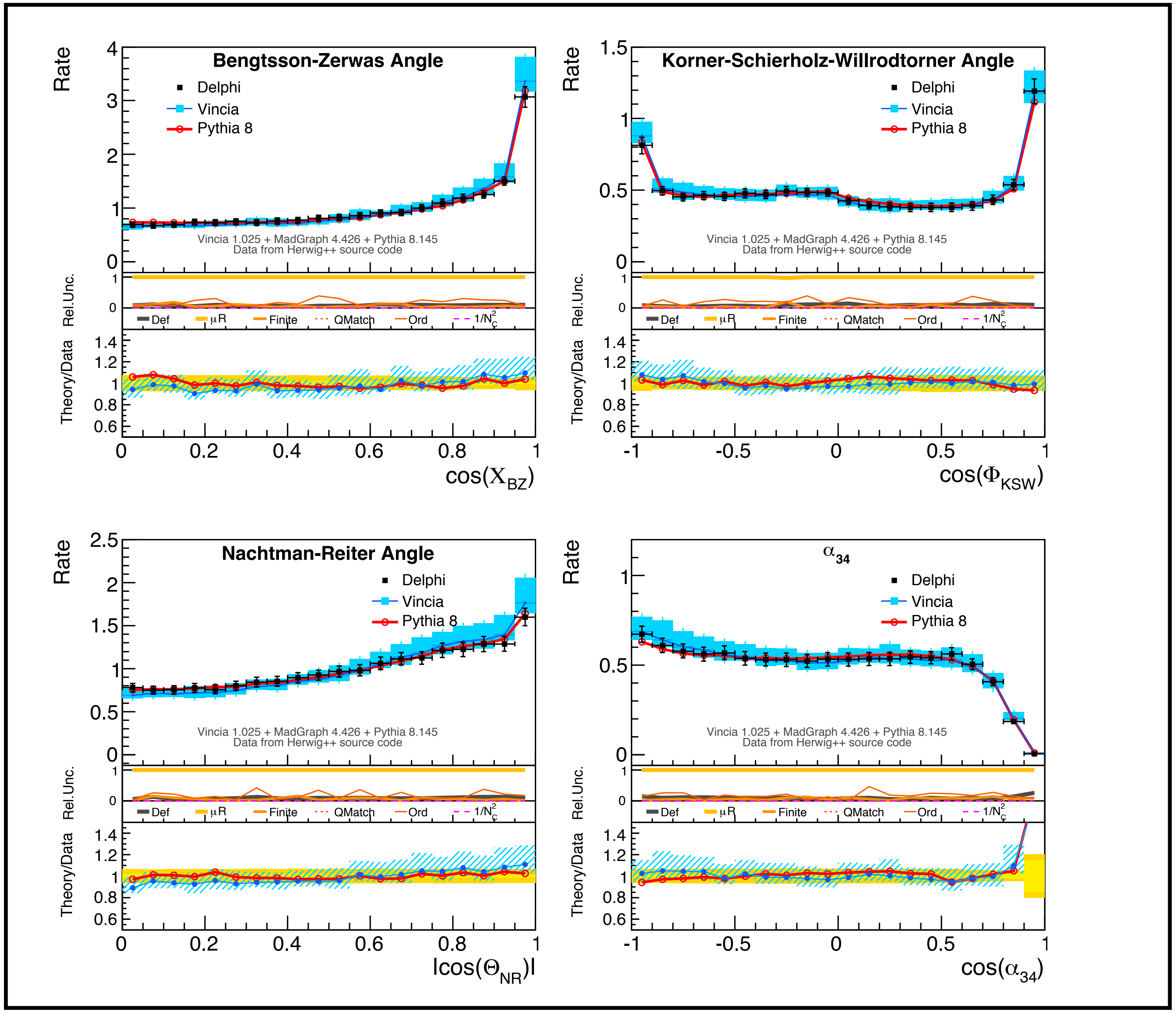
- Sensitive to polarization effects

Good News

- VINCIA is doing reliably well
- Non-trivial verification that shower+matching is working, etc.

Higher-order matching needed?

- PYTHIA 8 already doing a very good job on these observables



Approximations

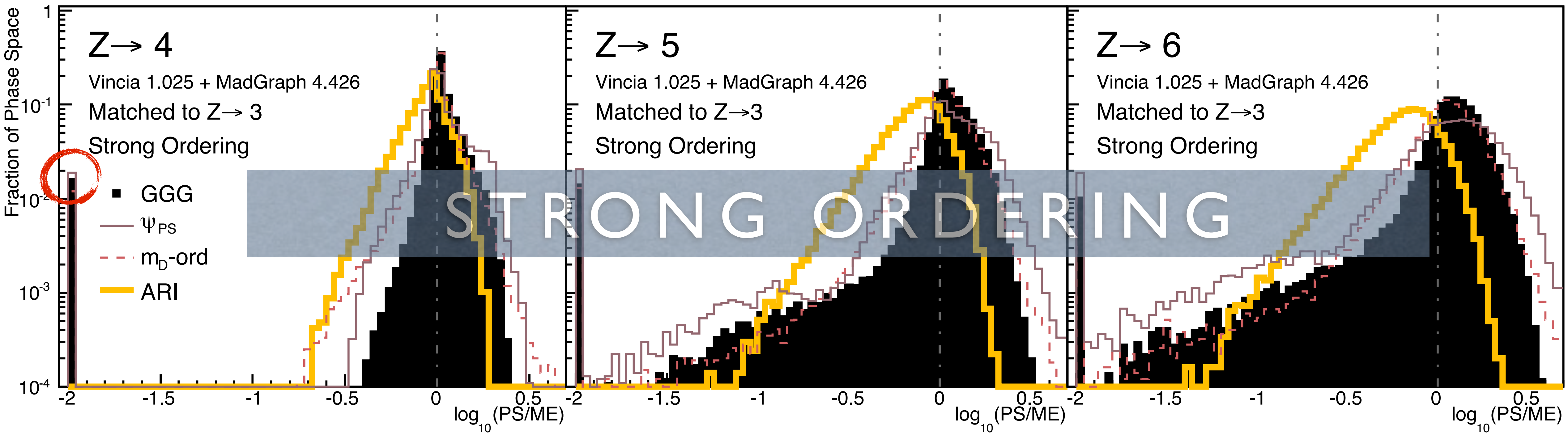
Q: How well do showers do?

Exp: Compare to data. Difficult to interpret; all-orders cocktail including hadronization, tuning, uncertainties, etc

Th: Compare products of splitting functions to full tree-level matrix elements

Plot distribution of **Log_{10}(PS/ME)**

(second order) (third order) (fourth order)



Dead Zone: 1-2% of phase space have no strongly ordered paths leading there*

*fine from strict LL point of view: those points correspond to "unordered" non-log-enhanced configurations

2→4

Generate Branchings *without* imposing strong ordering

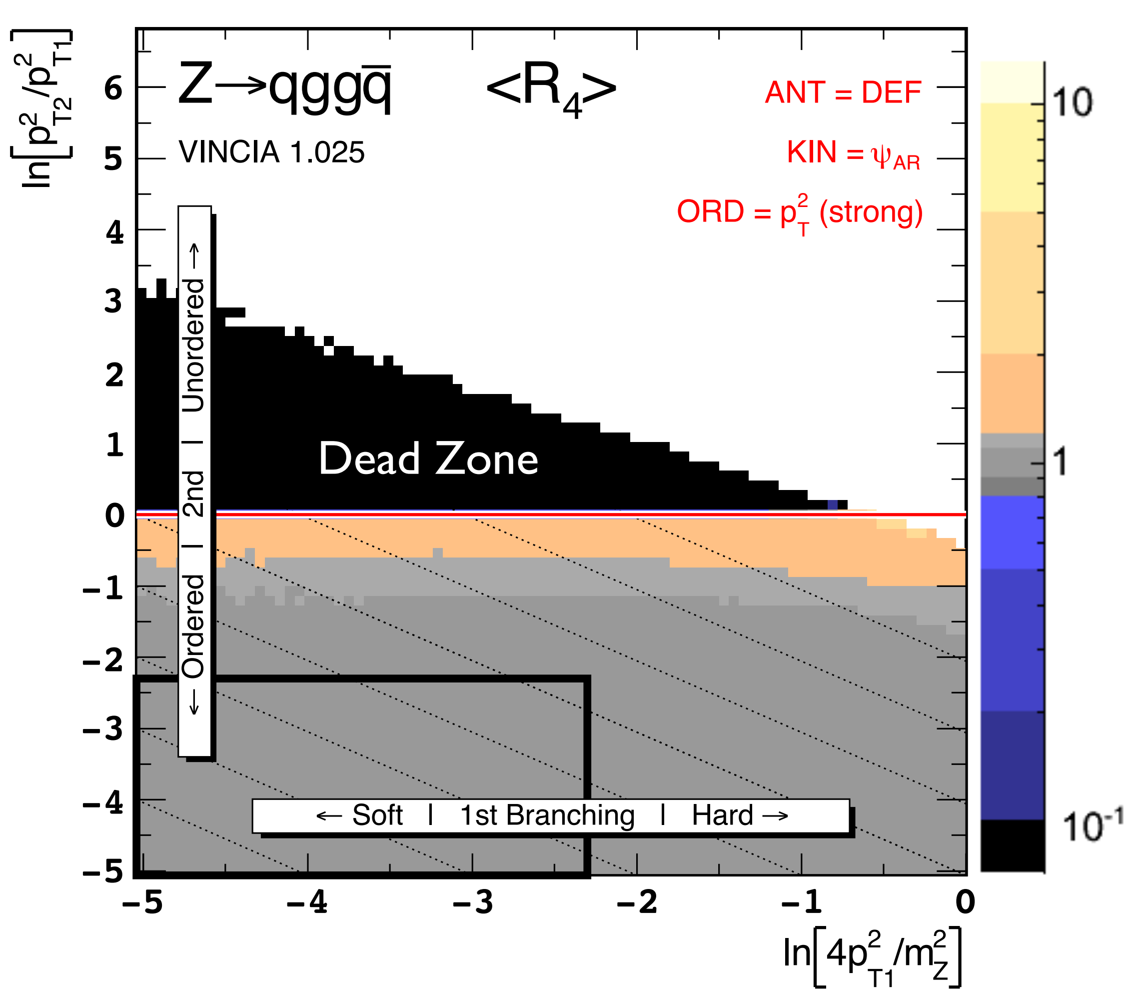
At each step, each dipole allowed to fill its entire phase space

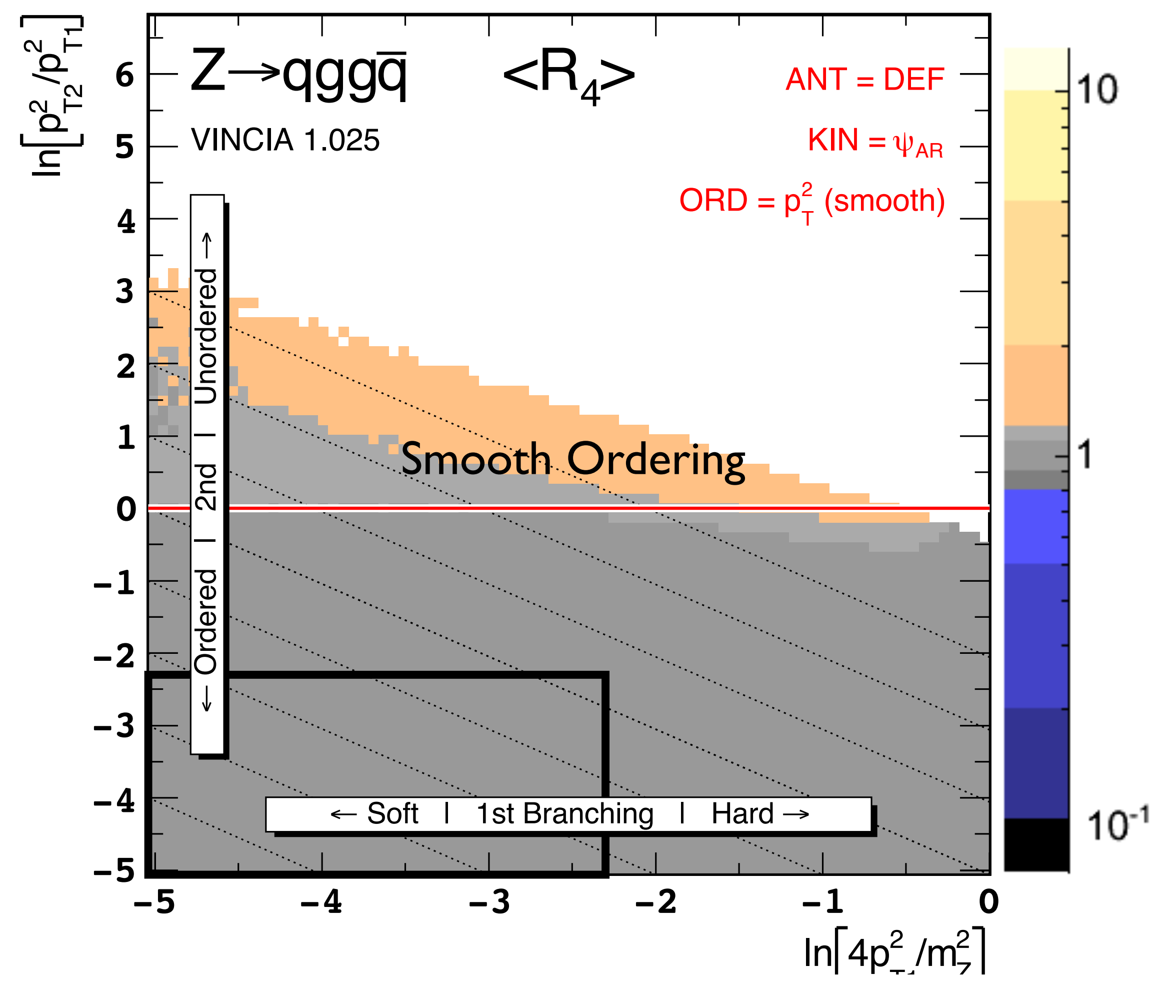
Overcounting removed by matching

+ smooth ordering beyond matched multiplicities

$$\frac{\hat{p}_\perp^2}{\hat{p}_\perp^2 + p_\perp^2} P_{\rm LL}$$

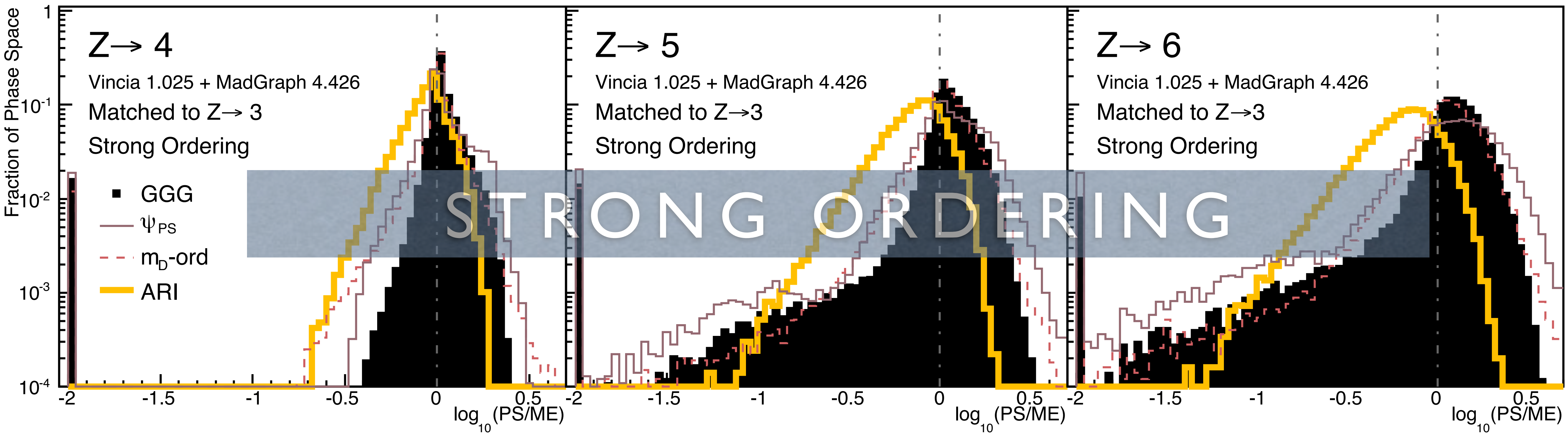
$\hat{p}_\perp^2$ last branching
$p_\perp^2$ current branching



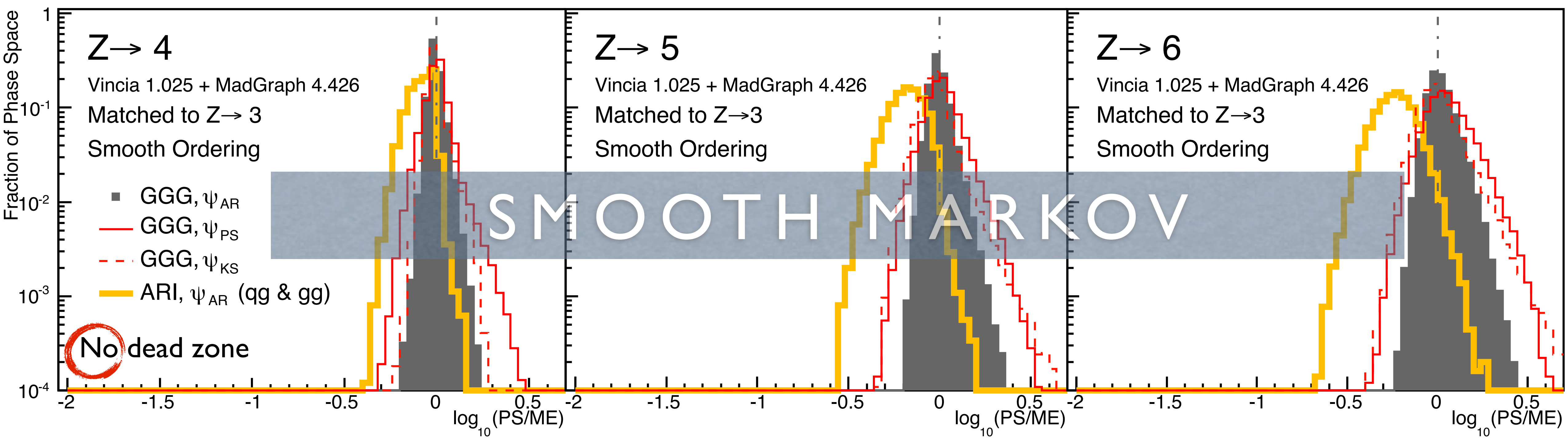


→ Better Approximations

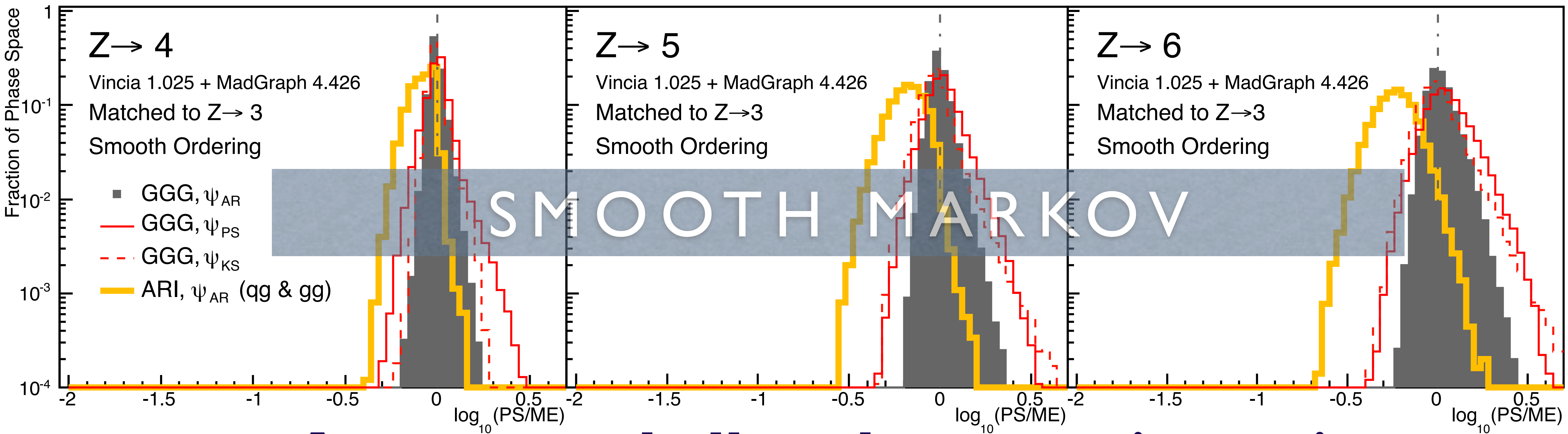
Distribution of **$\log_{10}(PS_{LO}/ME_{LO})$** (inverse ~ matching coefficient)



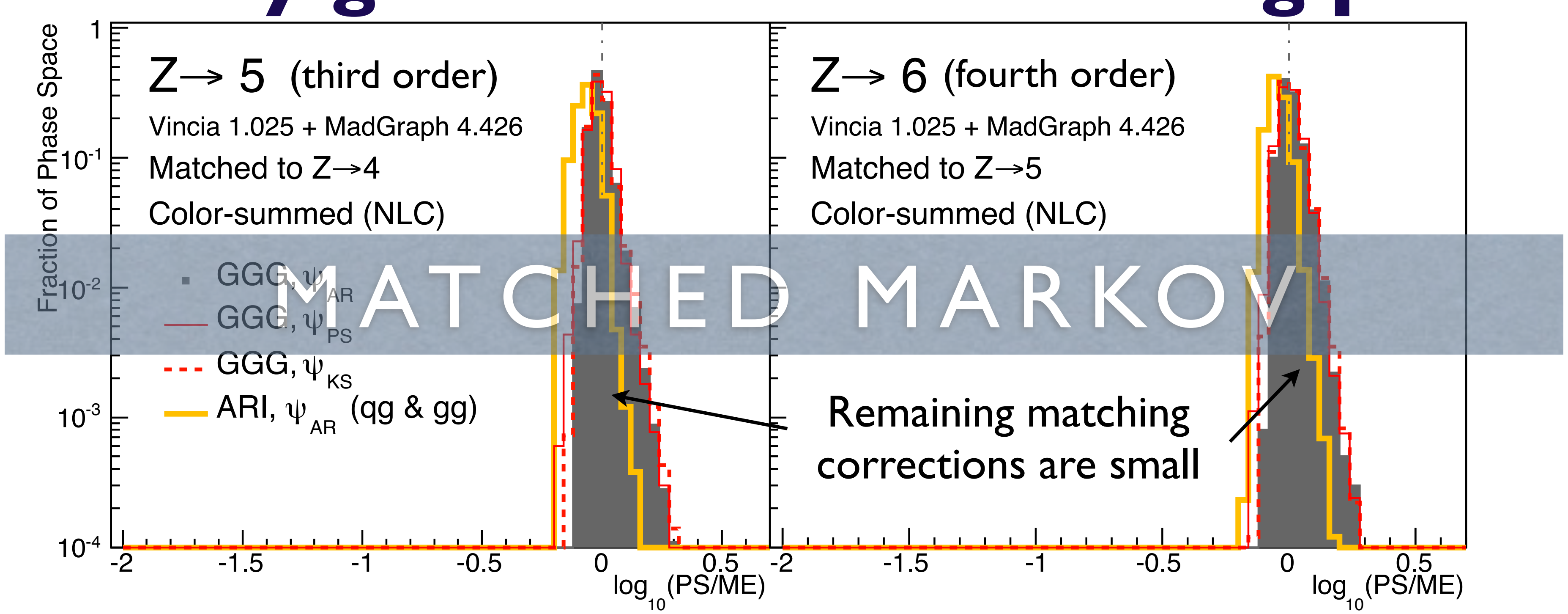
Leading Order, Leading Color, Flat phase-space scan, over **all of phase space** (no matching scale)



+ Matching (+ full colour)

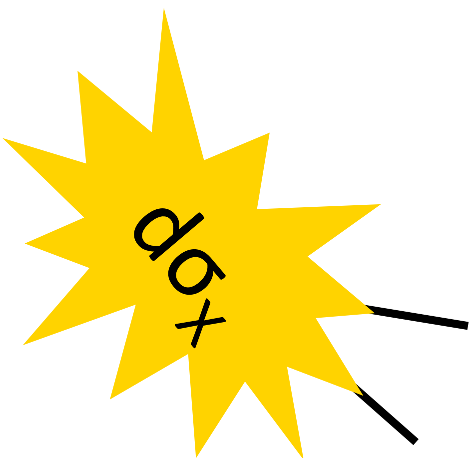


→ **A very good all-orders starting point**



Factorization + Universality

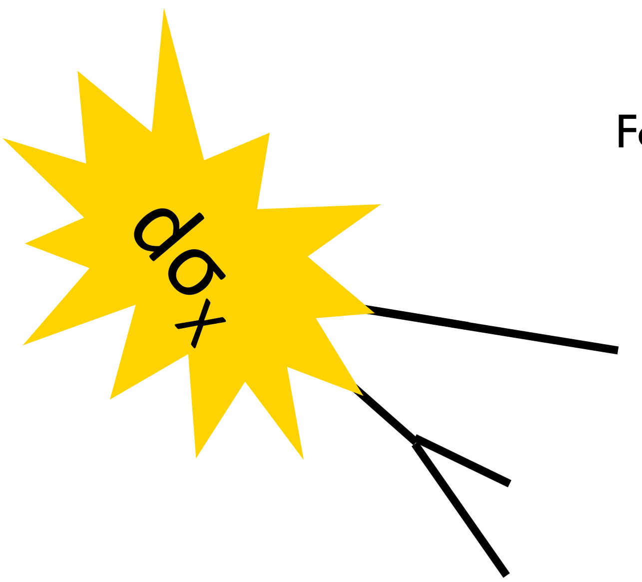
Bremsstrahlung



For any basic process $d\sigma_X$ = ✓ (calculated process by process)

Factorization + Universality

Bremsstrahlung



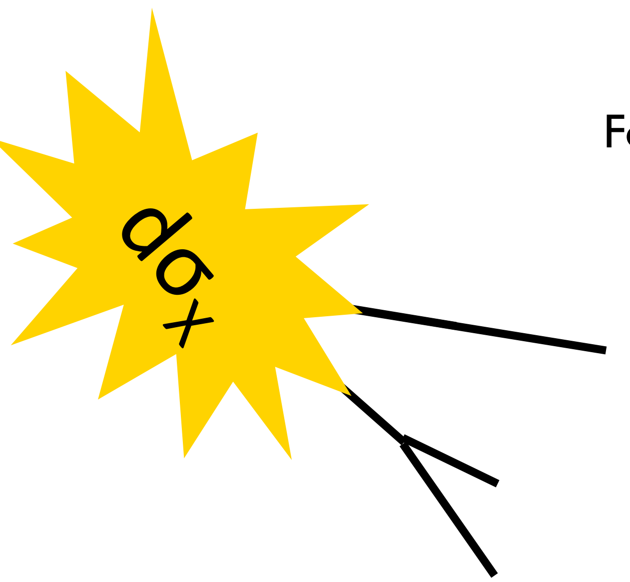
For any basic process $d\sigma_X$ = ✓ (calculated process by process)

$$d\sigma_{X+1} \sim N_C 2 g_s^2 \frac{ds_{i1}}{s_{i1}} \frac{ds_{1j}}{s_{1j}} d\sigma_X$$

Factorization + Universality

Bremsstrahlung

$s_{ij} = (p_i \cdot p_j)^2$



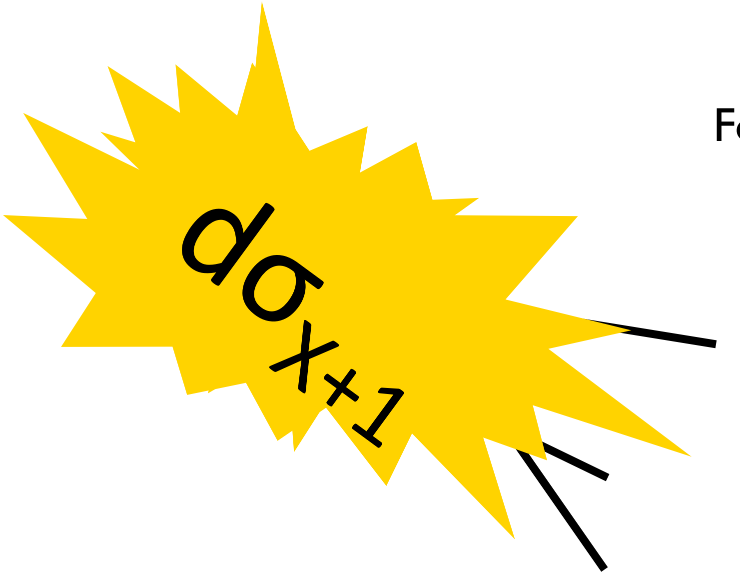
For any basic process $d\sigma_X$ = ✓ (calculated process by process)

$$d\sigma_{X+1} \sim N_C 2 g_s^2 \frac{ds_{i1}}{s_{i1}} \frac{ds_{1j}}{s_{1j}} d\sigma_X$$

Factorization + Universality

Bremsstrahlung

$s_{ij} = (p_i \cdot p_j)^2$



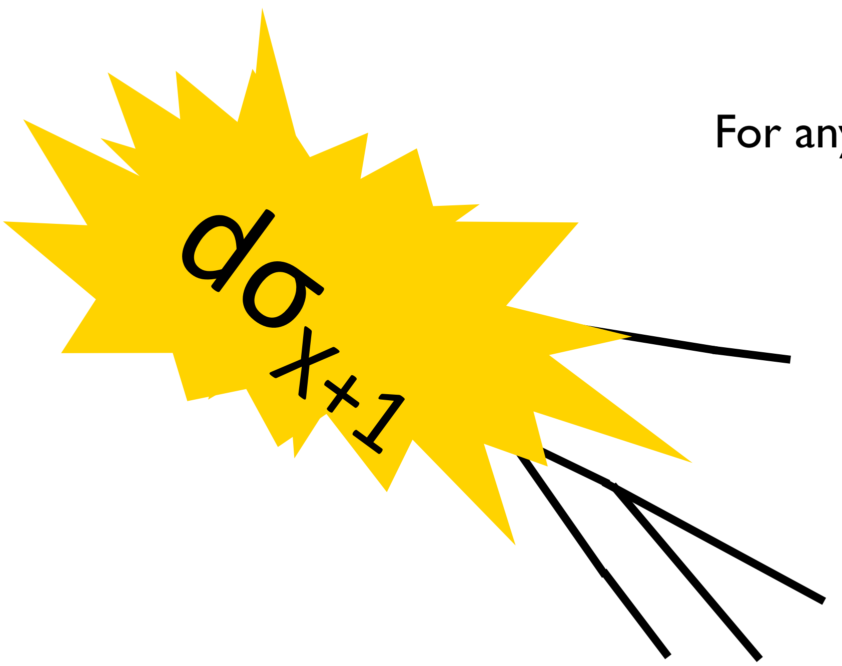
For any basic process $d\sigma_X$ = ✓ (calculated process by process)

$$d\sigma_{X+1} \sim N_C 2 g_s^2 \frac{ds_{i1}}{s_{i1}} \frac{ds_{1j}}{s_{1j}} d\sigma_X \quad ✓$$

Factorization + Universality

Bremsstrahlung

$s_{ij} = (p_i \cdot p_j)^2$



For any basic process $d\sigma_X$ = ✓ (calculated process by process)

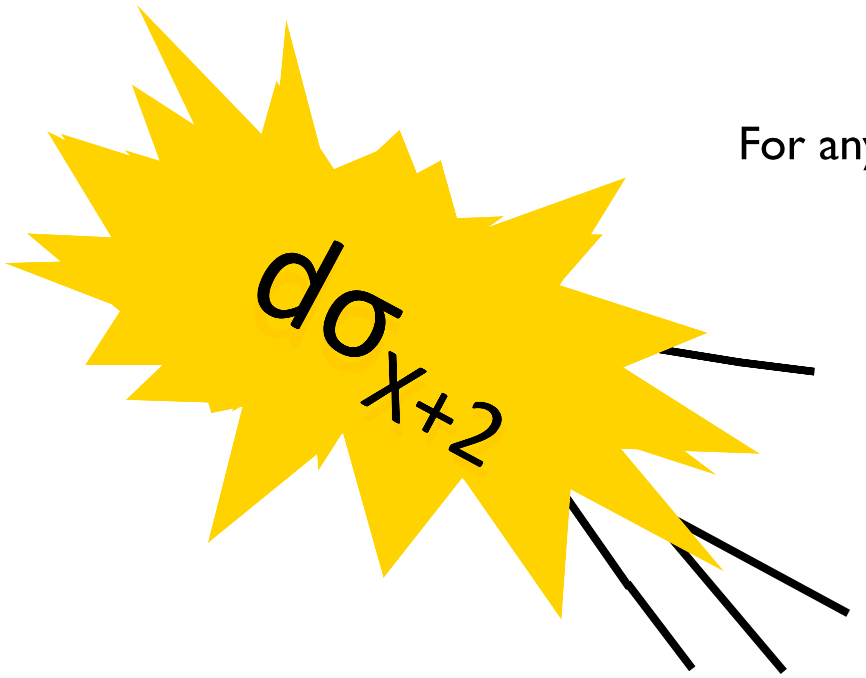
$$d\sigma_{X+1} \sim N_C 2 g_s^2 \frac{ds_{i1}}{s_{i1}} \frac{ds_{1j}}{s_{1j}} d\sigma_X$$ ✓

$$d\sigma_{X+2} \sim N_C 2 g_s^2 \frac{ds_{i2}}{s_{i2}} \frac{ds_{2j}}{s_{2j}} d\sigma_{X+1}$$

Factorization + Universality

Bremsstrahlung

$s_{ij} = (p_i \cdot p_j)^2$



For any basic process $d\sigma_X$ = ✓ (calculated process by process)

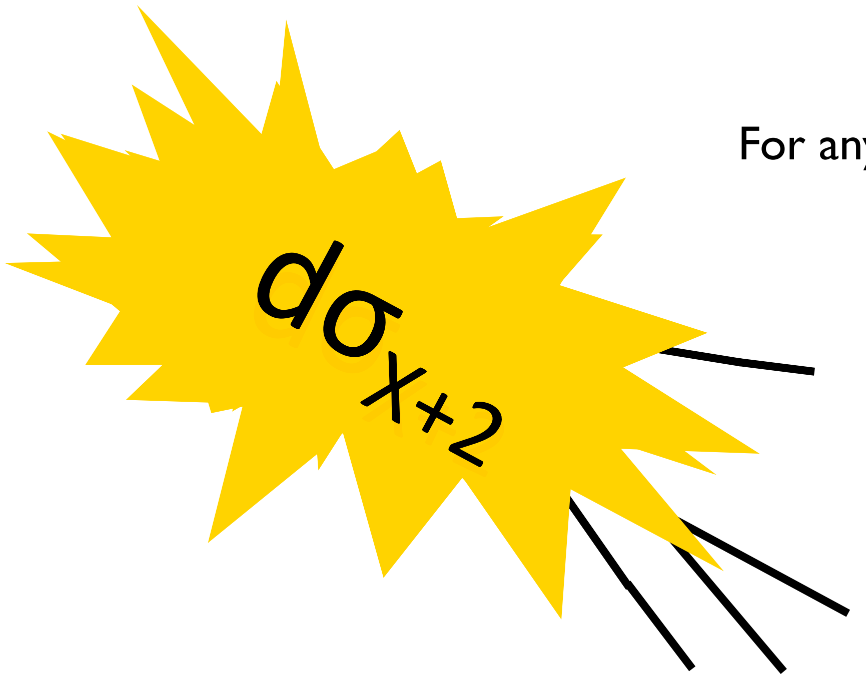
$$d\sigma_{X+1} \sim N_C 2 g_s^2 \frac{ds_{i1}}{s_{i1}} \frac{ds_{1j}}{s_{1j}} d\sigma_X \quad ✓$$

$$d\sigma_{X+2} \sim N_C 2 g_s^2 \frac{ds_{i2}}{s_{i2}} \frac{ds_{2j}}{s_{2j}} d\sigma_{X+1} \quad ✓$$

Factorization + Universality

Bremsstrahlung

$s_{ij} = (p_i \cdot p_j)^2$



For any basic process $d\sigma_X$ = ✓ (calculated process by process)

$$d\sigma_{X+1} \sim N_C 2 g_s^2 \frac{ds_{i1}}{s_{i1}} \frac{ds_{1j}}{s_{1j}} d\sigma_X \quad ✓$$

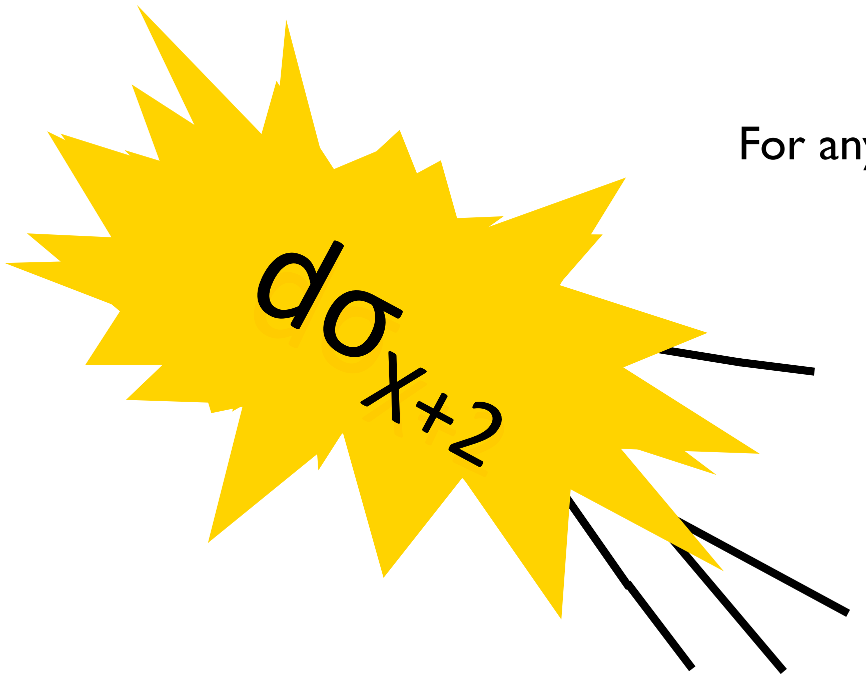
$$d\sigma_{X+2} \sim N_C 2 g_s^2 \frac{ds_{i2}}{s_{i2}} \frac{ds_{2j}}{s_{2j}} d\sigma_{X+1} \quad ✓$$

$$d\sigma_{X+3} \sim N_C 2 g_s^2 \frac{ds_{i3}}{s_{i3}} \frac{ds_{3j}}{s_{3j}} d\sigma_{X+2} \quad \ldots$$

Factorization + Universality

Bremsstrahlung

$s_{ij} = (p_i \cdot p_j)^2$



For any basic process $d\sigma_X$ = ✓ (calculated process by process)

$$d\sigma_{X+1} \sim N_C 2 g_s^2 \frac{ds_{i1}}{s_{i1}} \frac{ds_{1j}}{s_{1j}} d\sigma_X \quad ✓$$

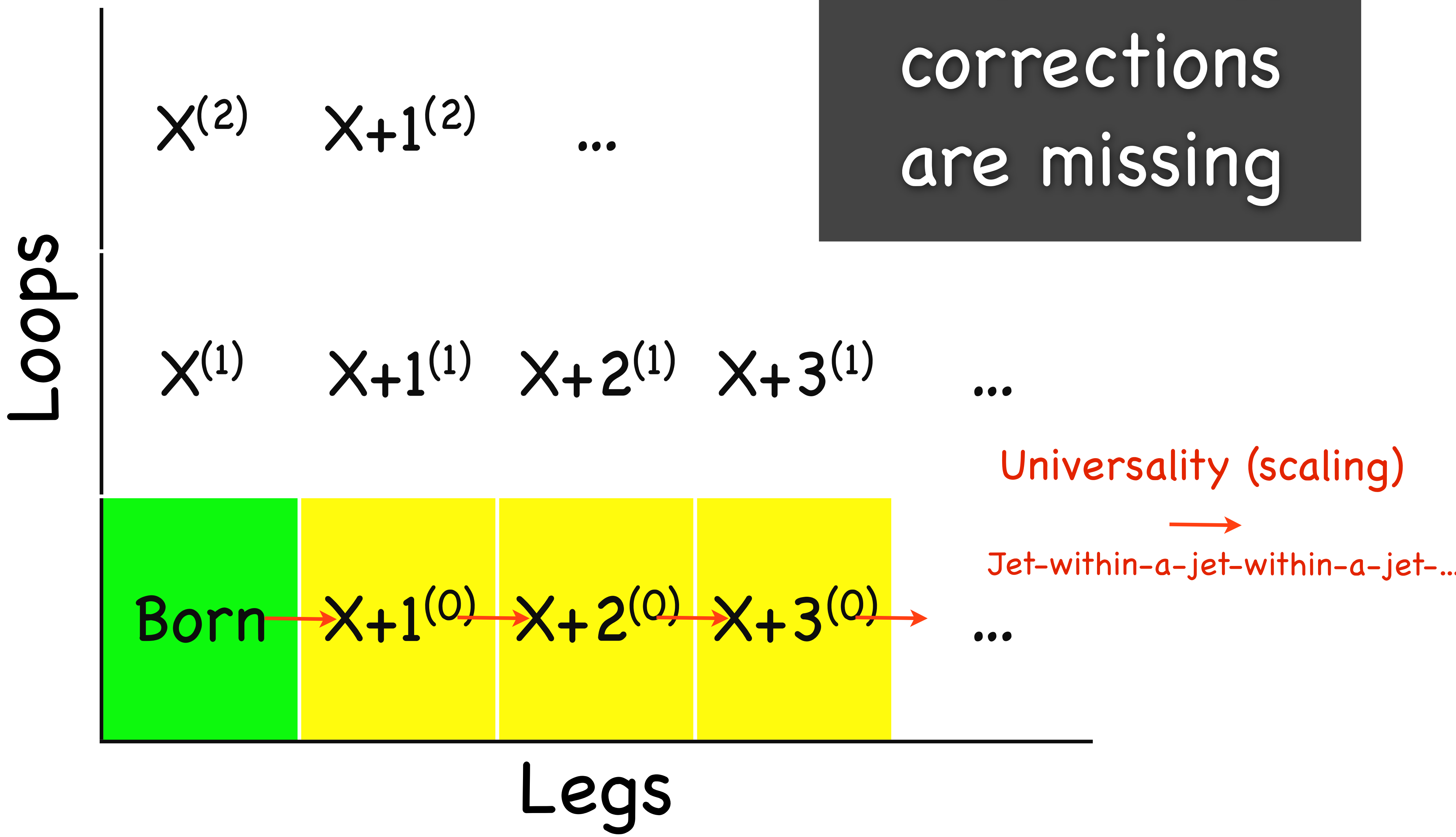
$$d\sigma_{X+2} \sim N_C 2 g_s^2 \frac{ds_{i2}}{s_{i2}} \frac{ds_{2j}}{s_{2j}} d\sigma_{X+1} \quad ✓$$

$$d\sigma_{X+3} \sim N_C 2 g_s^2 \frac{ds_{i3}}{s_{i3}} \frac{ds_{3j}}{s_{3j}} d\sigma_{X+2} \quad \ldots$$

This gives an approximation to infinite-order tree-level cross sections (here "double-log approximation: DLA")

(Running coupling and a few more subleading singular terms can also be included → MLLA, NLL, ...)

Factorization + Universality



The Virtual corrections are missing

Factorization + Universality

Let's squeeze it to get more juice...

$$d\sigma_{X+1} \sim N_C 2 g_s^2 \frac{ds_{i1}}{s_{i1}} \frac{ds_{1j}}{s_{1j}} d\sigma_X$$

Factorization + Universality

Let's squeeze it to get more juice...

$$d\sigma_{X+1} \sim P(Q) d\sigma_X$$

$P(Q) = O(\alpha)$

order α

Factorization + Universality

Let's squeeze it to get more juice...

$$d\sigma_{X+1} \sim P(Q) d\sigma_X$$

$P(Q) = O(\alpha)$

order α

↓

$$d\mathcal{P}_{branch} \sim P(Q) \mathcal{P}_{no-branch} dQ$$

Factorization + Universality

Let's squeeze it to get more juice...

$$d\sigma_{X+1} \sim P(Q) d\sigma_X$$

$P(Q) = O(\alpha)$

order α

↓

$$d\mathcal{P}_{branch} \sim P(Q) \mathcal{P}_{no-branch} dQ$$

↓ $P_{no-branch} + P_{branch} = 1$

Factorization + Universality

Let's squeeze it to get more juice...

$$d\sigma_{X+1} \sim P(Q) d\sigma_X$$

$P(Q) = O(\alpha)$

order α

↓

$$d\mathcal{P}_{branch} \sim P(Q)\mathcal{P}_{no-branch} dQ$$

↓ $P_{no-branch} + P_{branch} = 1$

$$d\mathcal{P}_{no-branch} = -d\mathcal{P}_{branch} \sim -P(Q)\mathcal{P}_{no-branch} dQ$$

Factorization + Universality

Let's squeeze it to get more juice...

$$d\sigma_{X+1} \sim P(Q) d\sigma_X$$

$P(Q) = O(\alpha)$

order α

↓

$$d\mathcal{P}_{branch} \sim P(Q) \mathcal{P}_{no-branch} dQ$$

↓ $P_{no-branch} + P_{branch} = 1$

$$d\mathcal{P}_{no-branch} = -d\mathcal{P}_{branch} \sim -P(Q) \mathcal{P}_{no-branch} dQ$$

→ $\mathcal{P}_{no-branch} = e^{-\int P(Q')dQ'}$

all orders in α

Factorization + Universality

Let's squeeze it to get more juice...

$$d\sigma_{X+1} \sim P(Q) d\sigma_X$$

$P(Q) = O(\alpha)$

order α

↓

$$d\mathcal{P}_{branch} \sim P(Q) \mathcal{P}_{no-branch} dQ$$

↓ $P_{no-branch} + P_{branch} = 1$

$$d\mathcal{P}_{no-branch} = -d\mathcal{P}_{branch} \sim -P(Q) \mathcal{P}_{no-branch} dQ$$

→ $\mathcal{P}_{no-branch} = e^{-\int P(Q')dQ'}$ all orders in α

Exponentiation

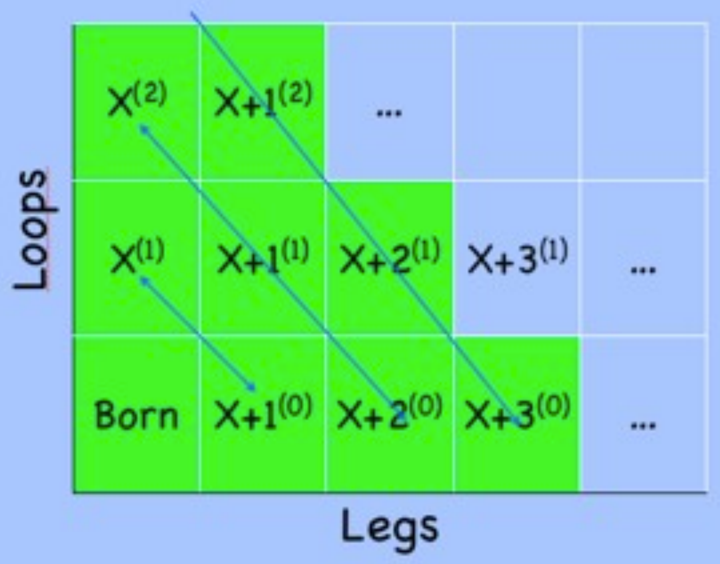
Unitarity

(*)Unitarity: Conservation of probability.

fixed order pQCD

KLN Theorem

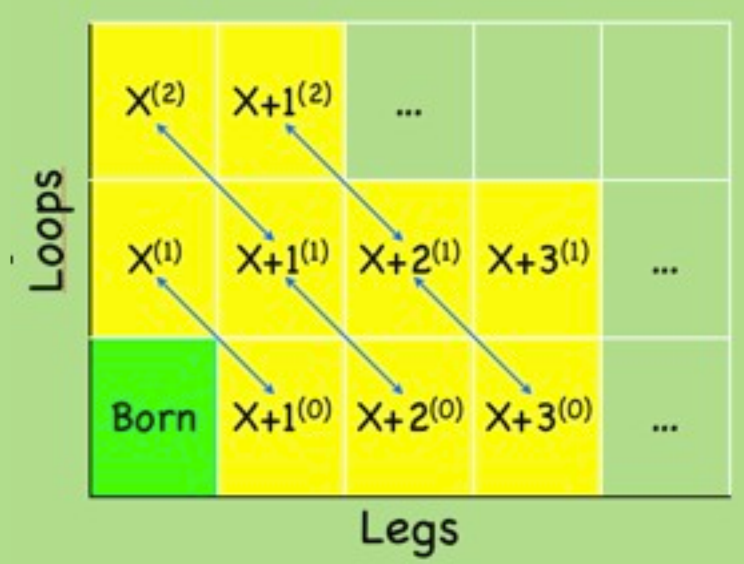
Relates Loops and Real emission.
Cancels IR divergences at each order



shower pQCD

Imposed by Event *evolution*:

When (X) branches to (X+1):
Gain one (X+1). Lose one (X).



$\frac{\mathrm{d}\sigma}{\mathrm{d}\Omega}$? Divide and conquer

Factorization → Split the problem into pieces

+ Quantum mechanics → Probabilities: $\frac{d\sigma}{d\Omega} \sim \mathcal{P}_{event}$

$$\mathcal{P}_{event} \approx \mathcal{P}_{hard/pQCD} \otimes \mathcal{P}_{soft/Had}$$

$\frac{d\sigma}{d\Omega}$? Divide and conquer

Factorization → Split the problem into pieces

+ Quantum mechanics → Probabilities: $\frac{d\sigma}{d\Omega} \sim \mathcal{P}_{event}$

$$\mathcal{P}_{event} \approx \mathcal{P}_{hard/pQCD} \otimes \mathcal{P}_{soft/Had}$$

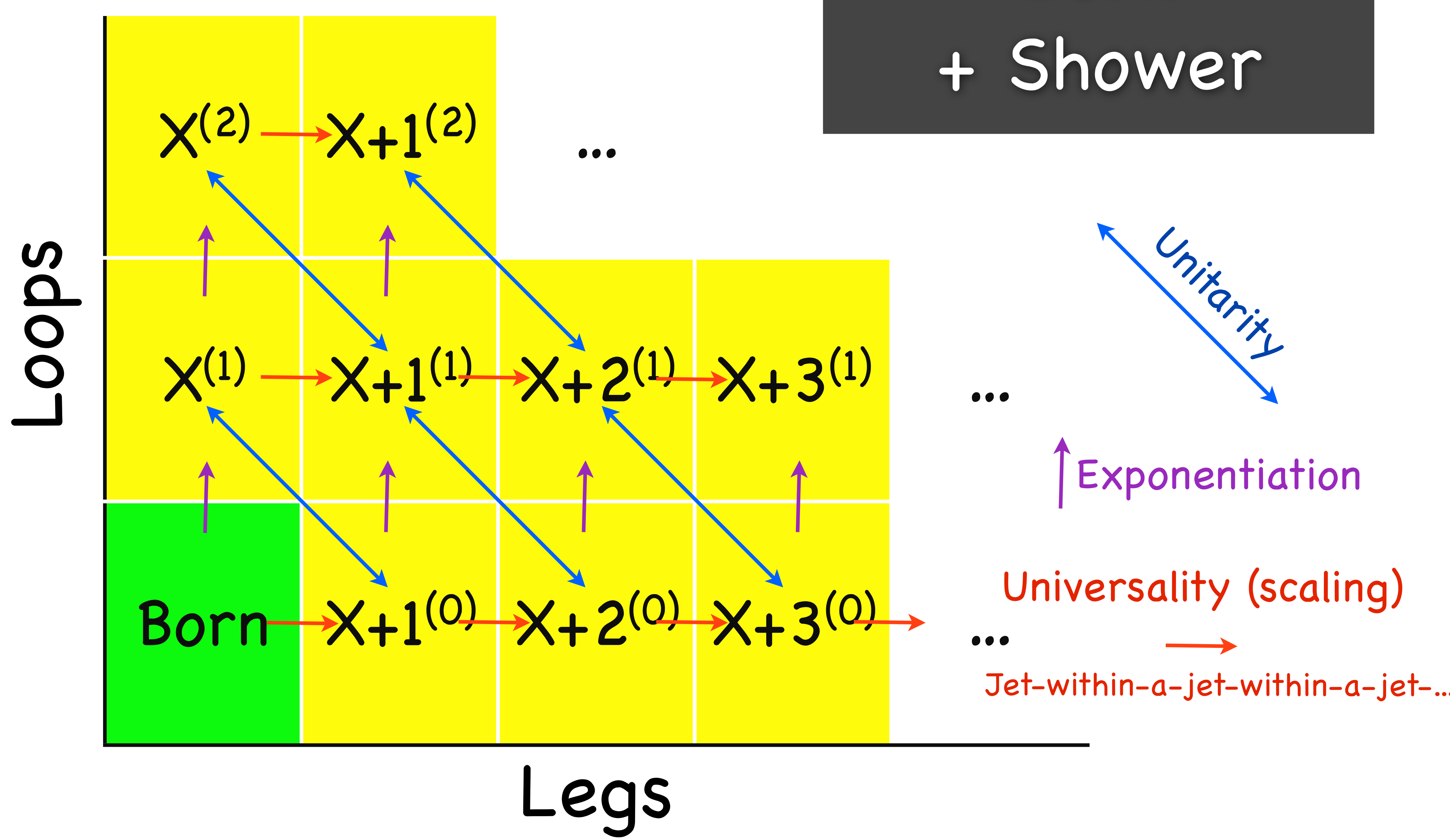
Parton shower
pQCD

Bootstrapped pQCD

(*) Bootstrapping: refers to a self-sustaining process that proceeds without external help

Resummation

Born + Shower



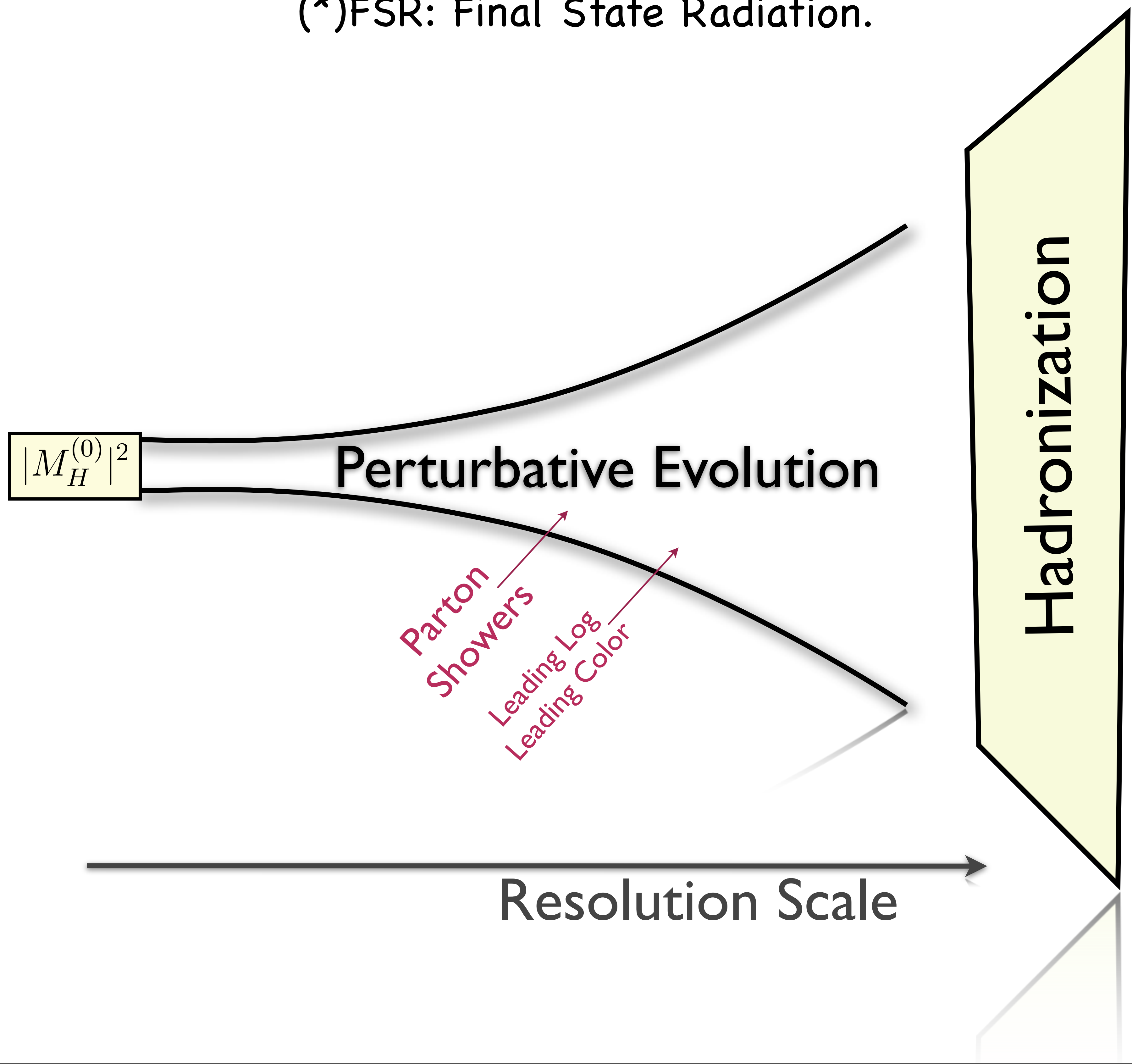
Analogy: Radioactive decay

Radioactive decay		Parton shower
→ *Evolves in time, t*	⟷	→ *Evolves* in resolution Q ~ virtuality, energy, …
→ *evolution equation with kernel:* *P(t)=λ, decay constant*	⟷	→ *evolution equation with kernel:* *P(Q)=*$\frac{d\sigma_{X+1}}{d\sigma_X}$ *which depends on evolution variable*
→ *Markov: probability to decay is independent of process*	⟷	→ *Markov: probability to branch is independent of process, only depends on Q (a priori).*
→ *Probability not to evolve (no-decay) is exponential, exp{-λt}*	⟷	→ *Probability not to evolve (no-branch) is exponential, exp{-∫P(Q')dQ'}*

What does a basic parton shower do?

Parton shower evolution (*FSR)

(*)FSR: Final State Radiation.



pQCD with parton showers

Resummation

Hard process (Born)
+ Shower