

# ANATAR with FORM 5.0

---

**Pooja Mukherjee**

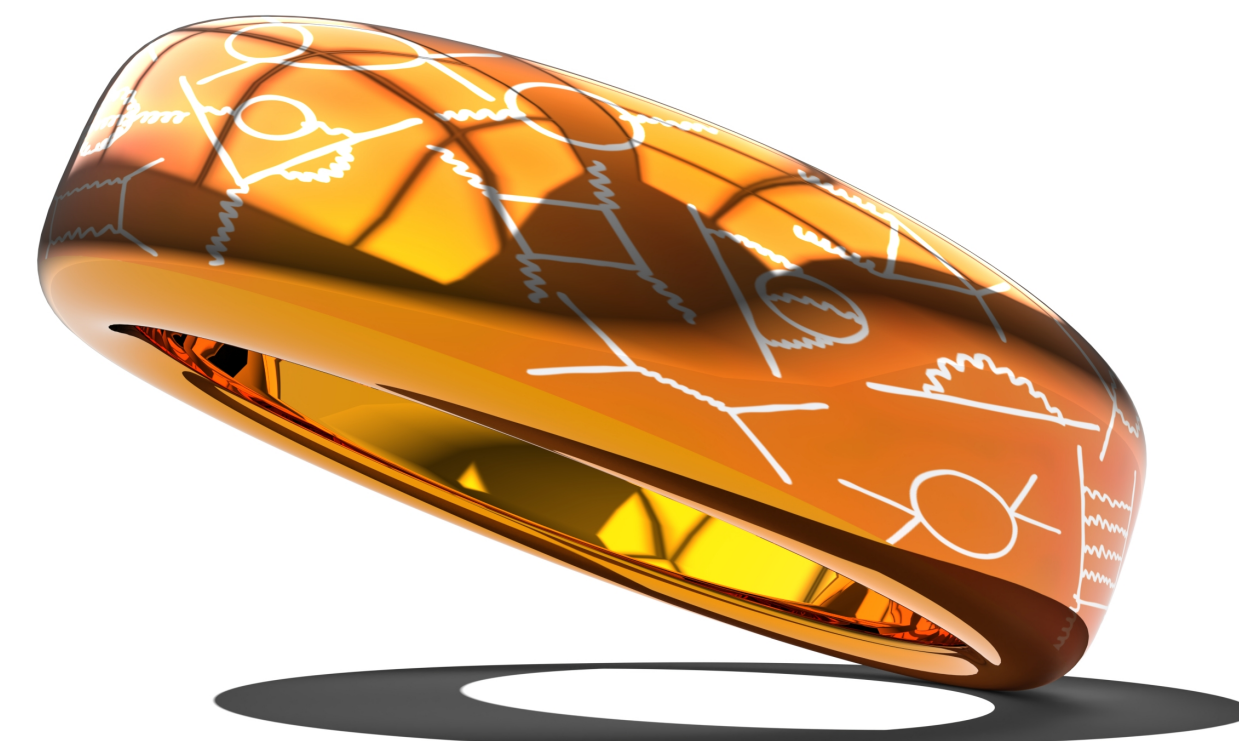
**In collaboration with Prof. Claude Duhr and Dr. Andres Vasquez**

**FORM developers meeting | Jun 25, 2026**



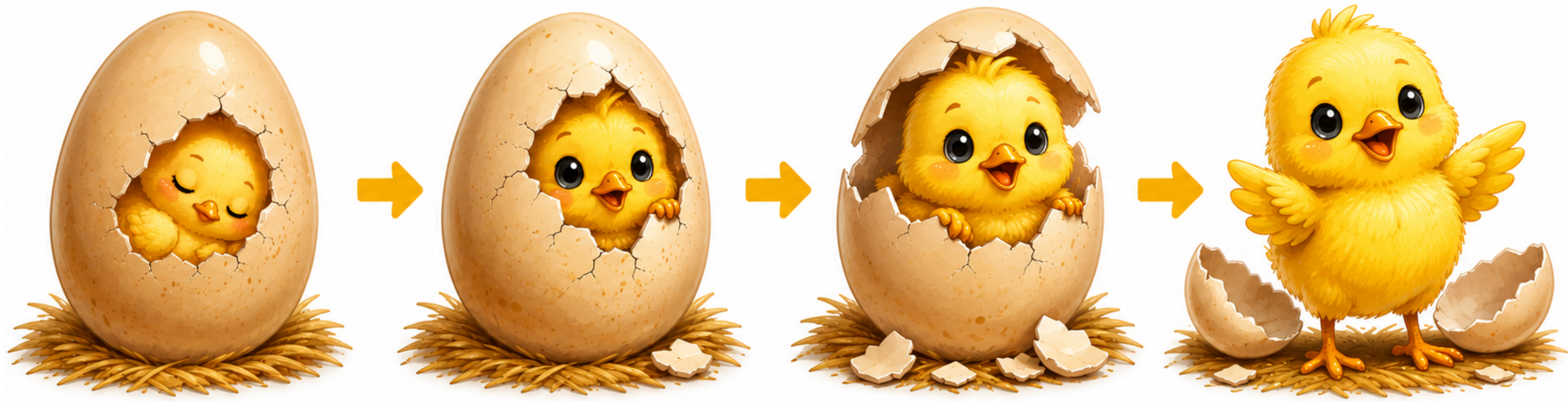
**Universität Hamburg**

**DER FORSCHUNG | DER LEHRE | DER BILDUNG**





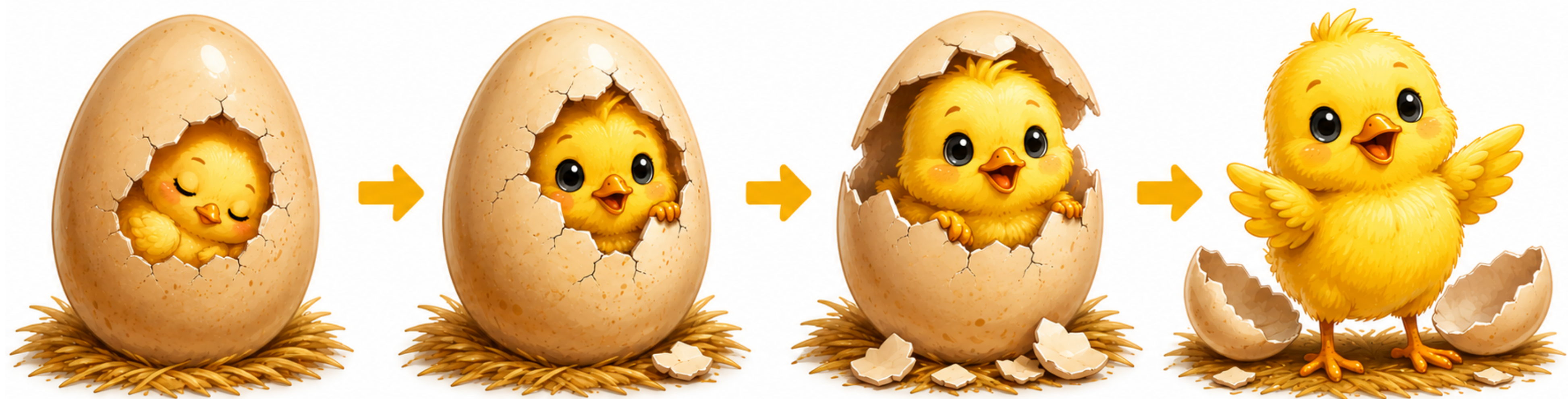
# Anticipated stages of development



P.C. Chatgpt



# Anticipated stages of development



P.C. Chatgpt



Current stage



# AN Automated Tool for Higher order Amplitude geneRation



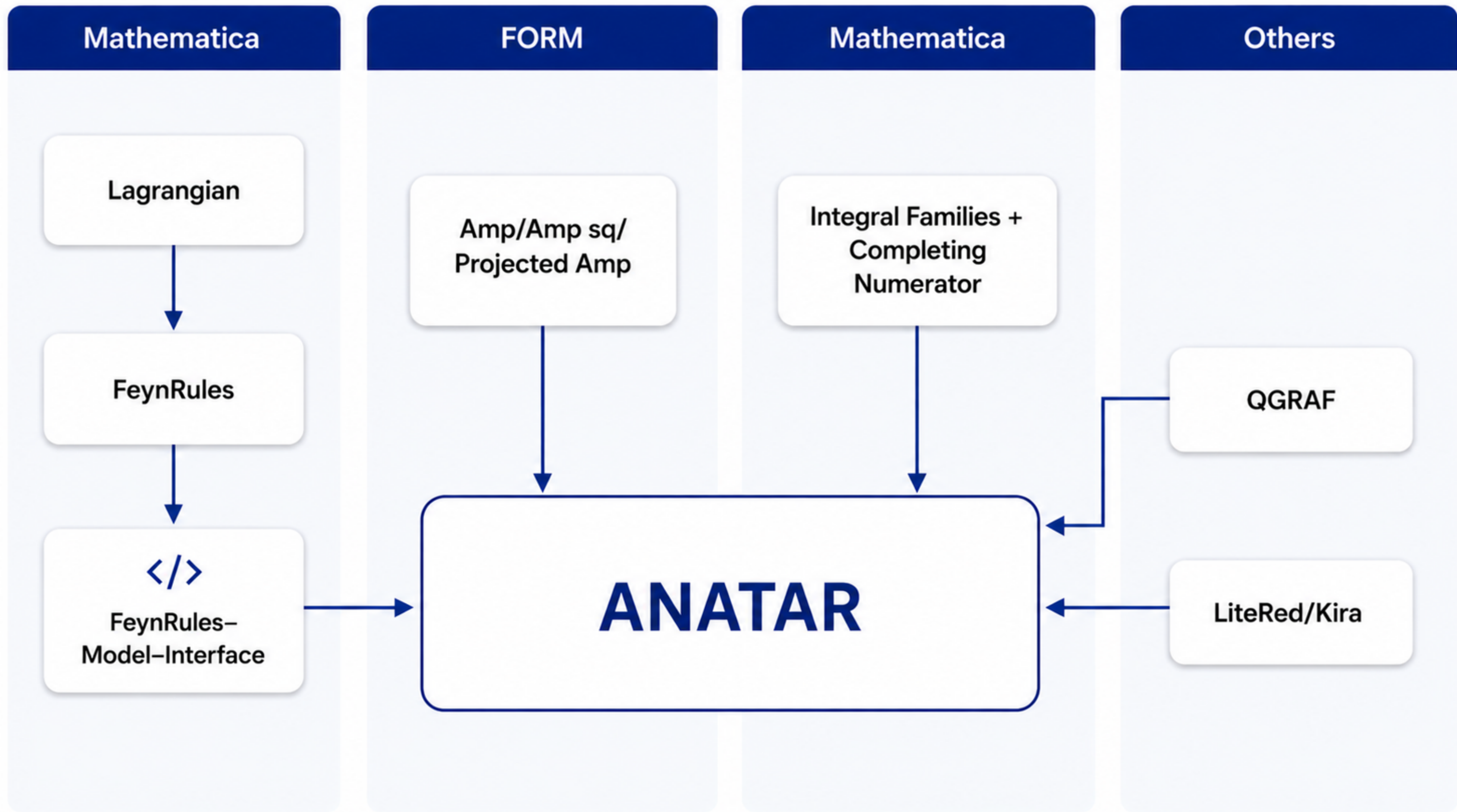
P.C. Andres & Paarth

<https://github.com/ANATAR-hep/ANATAR>

[JHEP 04 (2026) 199]



# The Assembly





# QGRAF Diagram generator

## QGRAF output

```
id a1 = (+1)*
pol(b(-1,p1))*
pol(bbar(-3,p2))*
pol(A(-2,p3))*
pol(A(-4,p4))*
pro(b(1,2,p1-p3))*
pro(b(3,4,-p2+p4))*
pro(Z(5,6,-k1))*
pro(b(7,8,-k1+p1-p3))*
pro(b(9,10,-k2))*
pro(G(11,12,k2-p2+p4))*
pro(b(13,14,-k1-k2))*
vrtx(bbar(2,-p1+p3),b(-1,p1),A(-2,-p3))*
vrtx(bbar(-3,p2),b(3,-p2+p4),A(-4,-p4))*
vrtx(bbar(8,k1-p1+p3),b(1,p1-p3),Z(5,-k1))*
vrtx(bbar(4,p2-p4),b(9,-k2),G(11,k2-p2+p4))*
vrtx(bbar(10,k2),b(13,-k1-k2),Z(6,k1))*
vrtx(bbar(14,k1+k2),b(7,-k1+p1-p3),G(12,-k2+p2-p4));
```

## FORM files

```
#procedure FeynmanRules
.sort

argument;
id ff?Field(x?partTagExt[n],?a) = ff(x,?a,ExtLor[n]);
id ff?Field(x?partTagInt[n],?a) = ff(x,?a,IntLor[n]);
id ff?Fermion(x?partTagExt[n],?a) = ff(x,?a,Spin[n]);
id ff?Fermion(x?partTagInt[n],?a) = ff(x,?a,IntSpin[n]);
id ff?Gluoned(x?partTagExt[n],?a) = ff(x,?a,Gluon[n]);
id ff?Gluoned(x?partTagInt[n],?a) = ff(x,?a,IntGluon[n]);
id ff?Coloured(x?partTagExt[n],?a) = ff(x,?a,Colour[n]);
id ff?Coloured(x?partTagInt[n],?a) = ff(x,?a,IntColour[n]);
endargument;

id vrtx(G0(?a, p1?, Lor1?), G0(?b, p2?, Lor2?), G0(?c, p3?, Lor3?), G0(?d, p4?, Lor4?)) = GC40*im;
id vrtx(G0(?a, p1?, Lor1?), G0(?b, p2?, Lor2?), GP(?c, p3?, Lor3?), GPbar(?d, p4?, Lor4?)) = GC38*im;
id vrtx(GP(?a, p1?, Lor1?), GP(?b, p2?, Lor2?), GPbar(?c, p3?, Lor3?), GPbar(?d, p4?, Lor4?)) = GC39*im;
id vrtx(G0(?a, p1?, Lor1?), G0(?b, p2?, Lor2?), H(?c, p3?, Lor3?), H(?d, p4?, Lor4?)) = GC38*im;
id vrtx(GP(?a, p1?, Lor1?), GPbar(?b, p2?, Lor2?), H(?c, p3?, Lor3?), H(?d, p4?, Lor4?)) = GC38*im;
id vrtx(H(?a, p1?, Lor1?), H(?b, p2?, Lor2?), H(?c, p3?, Lor3?), H(?d, p4?, Lor4?)) = GC40*im;

#procedure Propagators
.sort

id pro(ff?Field(x?,y?partTagInt[n],?a)) = pro(ff(x,y,?a,IntLor[n]));
id pro(ff?Fermion(x?,y?partTagInt[n],p1?,iLor1?,Spin1?,iLor2?)) = pro(ff(x,y,p1,Spin1,IntSpin[n]));
id pro(ff?Fermion(x?,y?partTagInt[n],p1?,iLor1?,Spin1?,Colour1?,iLor2?)) = pro(ff(x,y,p1,Spin1,IntSpin[n],Colour1,IntColour[n]));
id pro(ff?Gluoned(x?,y?partTagInt[n],p1?,iLor1?,Gluon1?,iLor2?)) = pro(ff(x,y,p1,iLor1,iLor2,Gluon1,IntGluon[n]));
id pro(ff?Gluoned(x?,y?partTagInt[n],0,iLor1?,Gluon1?,iLor2?)) = pro(ff(x,y,0,iLor1,iLor2,Gluon1,IntGluon[n]));

*****
* Propagators non dependent on Gauge *
*****

id pro(b(?a, p1?, Spin1?, Spin2?, Colour1?, Colour2?)) = im*Den(p1, MB)*d_(Colour1, Colour2)*(MB*GammaM(1, Spin1, Spin2) + GammaM(p1, Spin1, Spin2));
id pro(c(?a, p1?, Spin1?, Spin2?, Colour1?, Colour2?)) = im*Den(p1, 0)*d_(Colour1, Colour2)*GammaM(p1, Spin1, Spin2);
id pro(d(?a, p1?, Spin1?, Spin2?, Colour1?, Colour2?)) = im*Den(p1, 0)*d_(Colour1, Colour2)*GammaM(p1, Spin1, Spin2);
```



# FORM Diagram generator

## FORM output

```
expr =
+
  topo_(1)
  *node_(1,1,glu(-p1))
  *node_(2,1,glu(-p2))
  *node_(3,1,t(-p3))
  *node_(4,1,tbar(-p4))
  *node_(5,1,glu(-p5))
  *node_(6,g,glu(p1),glu(p2),glu(-k1))
  *node_(7,g,tbar(p3),t(-k2),glu(k1))
  *node_(8,g,tbar(k2),t(p4),glu(p5))
  *edge_(1,glu(p1),1,6)
  *edge_(2,glu(p2),2,6)
  *edge_(3,tbar(p3),3,7)
  *edge_(4,t(p4),4,8)
  *edge_(5,glu(p5),5,8)
  *edge_(6,glu(k1),6,7)
  *edge_(7,tbar(k2),7,8)
+
  topo_(2)
  *node_(1,1,glu(-p1))
  *node_(2,1,glu(-p2))
  *node_(3,1,t(-p3))
  *node_(4,1,tbar(-p4))
  *node_(5,1,glu(-p5))
  *node_(6,g,glu(p1),glu(p2),glu(-k1))
  *node_(7,g,tbar(-k2),t(p4),glu(k1))
  *node_(8,g,tbar(p3),t(k2),glu(p5))
  *edge_(1,glu(p1),1,6)
  *edge_(2,glu(p2),2,6)
  *edge_(3,tbar(p3),3,8)
  *edge_(4,t(p4),4,7)
  *edge_(5,glu(p5),5,8)
  *edge_(6,glu(k1),6,7)
  *edge_(7,t(k2),7,8)
+
  (3)
```

## FORM files

```
argument node_;
id   ff?Field(p1?ext[n]) = ff(p1,ExtLor[n]);
id   ff?Field(p1?int[n]) = ff(p1,IntLor[n]);
id   ff?Fermion(p1?ext[n]) = ff(p1,Spin[n]);
id   ff?Fermion(p1?int[n]) = ff(p1,IntSpin[n]);
id   ff?Gluoned(p1?ext[n]) = ff(p1,Gluon[n]);
id   ff?Gluoned(p1?int[n]) = ff(p1,IntGluon[n]);
id   ff?Coloured(p1?ext[n]) = ff(p1,Colour[n]);
id   ff?Coloured(p1?int[n]) = ff(p1,IntColour[n]);
endargument;

id   node_(?a,G0(p1?, Lor1?), G0(p2?, Lor2?), G0(p3?, Lor3?), G0(p4?, Lor4?)) = GC40*im;
id   node_(?a,G0(p1?, Lor1?), G0(p2?, Lor2?), GP(p3?, Lor3?), GPbar(p4?, Lor4?)) = GC38*im;
id   node_(?a,GP(p1?, Lor1?), GP(p2?, Lor2?), GPbar(p3?, Lor3?), GPbar(p4?, Lor4?)) = GC39*im;
id   node_(?a,G0(p1?, Lor1?), G0(p2?, Lor2?), H(p3?, Lor3?), H(p4?, Lor4?)) = GC38*im;
```





# Questions

---

◎ Is there option to get the number of diagrams generated ?

◎ If not then alternatively :

```
local expr = diagrams_(QCD, {glu,glu}, {t, tbar, glu},
ext, int, 0, `WithEdges_`+'OnShell_`+'NoTadpole_`+'NoSnail_`);
.sort

print+ss expr;
.sort

#$a = 0;
If(occurs(topo_) ) $a = $a+1;
.sort

#do i = 1, `$a'
*#do i = 1, 1
.sort

bracket topo_;
.sort
local amp`i` = expr[topo_(`i`)];
.sort
```



# Questions

◎ Is there option to get the number of diagrams generated ?

◎ If not then alternatively :

But for this process :

```
local expr = diagrams_(QCD, {glu,glu}, {t, tbar, glu},
ext, int, 0, `WithEdges_`+'OnShell_`+'NoTadpole_`+'NoSnail_`);
.sort

print+ss expr;
.sort

#$a = 0;
If(occurs(topo_) ) $a = $a+1;
.sort

#do i = 1, `$a'
*#do i = 1, 1
.sort

bracket topo_;
.sort
local amp`i` = expr[topo_(`i`)];
.sort
```

```
local expr = diagrams_(QCD, {glu}, {glu},
ext, int, 1, `WithEdges_`+'OnShell_`+'NoTadpole_`+'NoSnail_`);
.sort
```

```
#-

expr =
+
  1/2
  *topo_(1)
  *node_(1,1,glu(-p1))
  *node_(2,1,glu(-p2))
  *node_(3,g,glu(p1),glu(-k1),glu(-k2))
  *node_(4,g,glu(p2),glu(k1),glu(k2))
  *edge_(1,glu(p1),1,3)
  *edge_(2,glu(p2),2,4)
  *edge_(3,glu(k1),3,4)
  *edge_(4,glu(k2),3,4)
-
  topo_(1)
  *node_(1,1,glu(-p1))
  *node_(2,1,glu(-p2))
  *node_(3,g,glu(p1),glu(-k1),glu(-k2))
  *node_(4,g,glu(p2),glu(k1),glu(k2))
  *edge_(1,glu(p1),1,3)
  *edge_(2,glu(p2),2,4)
  *edge_(3,glu(k1),3,4)
  *edge_(4,glu(k2),3,4)
```



# Questions

---

© Can 'WithEdges\_' be put as a default option ?

© Does particles not come under 'set'?

```
set    Field: gh, glu, t,ghbar,tbar;  
[argument node_  
id    ff?Field(p1?ext[n]) = ff(p1,ExtLor[n]);  
endargument;
```

© Is there any alternative for negative momentum sequences ?

```
Set    int: k1,...,k10,-k1,-k2,-k3,-k4,-k5,-k6,-k7,-k8,-k9,-k10;
```



# Questions

---

- ◎ Speeding up the amplitude/amplitude sq./ projected amplitude computation
  - ✦ Done in FORM
  - ✦ Used `id` instead of `local` for every diagram computation
  - ✦ Brackets are used for keeping the relevant pieces and hide the redundant stuff
  - ✦ Is there any suggestions for storing repeated computation temporarily and then substituting back ?

**ToSpectator ?**



# Questions

---

## © Amplitudes to topologies :

- ✦ Done in Mathematica
- ✦ Identifies momentum shift mappings between integral families
- ✦ Then rewriting the numerator in terms of the denominator
- ✦ But denominators not connected via momentum shifts but reducible via partial fractioning are done manually

**FORMapart ?**

**Adam's talk**

**Alibary ?**

**Vitaly's work**





Thank you !