MARTY

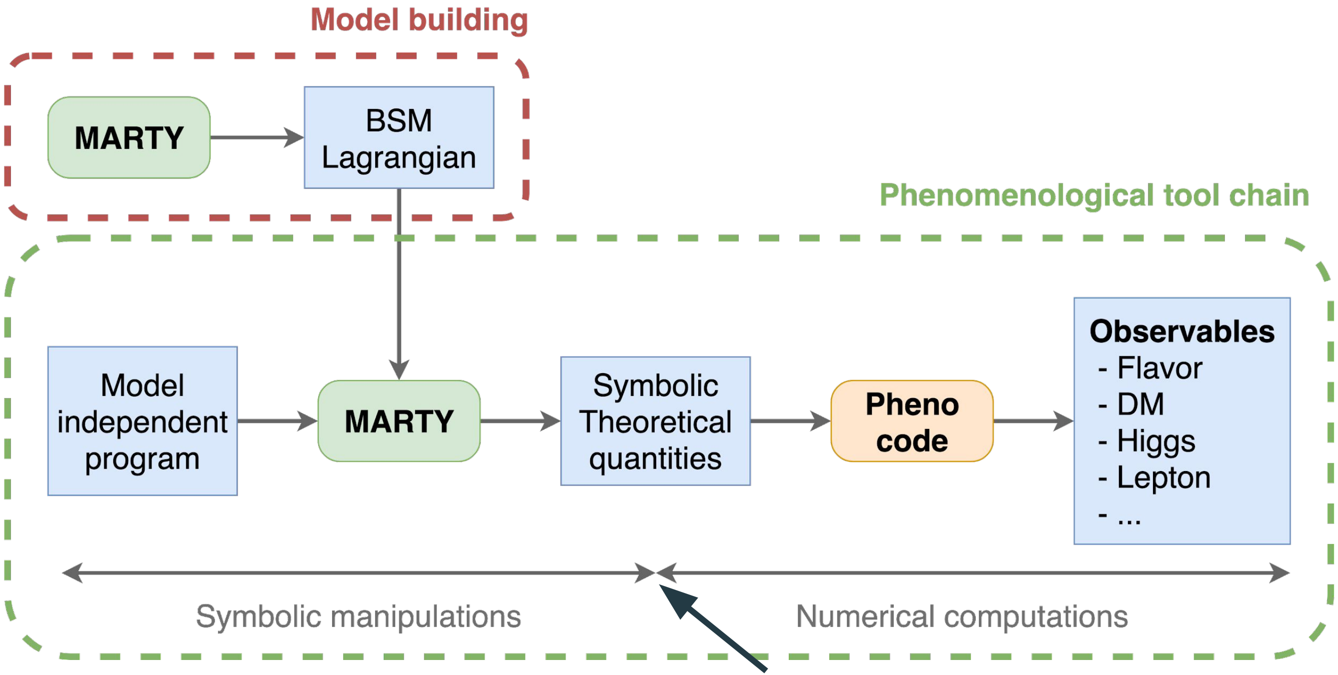
Status, new developments and perspectives

Grégoire Uhlrich 04/05/23

Introduction

I. Introduction
II. Design & Architecture
III. The User Side
IV. What’s New (+ demo)

Purpose - From the Lagrangian to Observables



Code generation: MARTY ⟶ Library of numerical functions

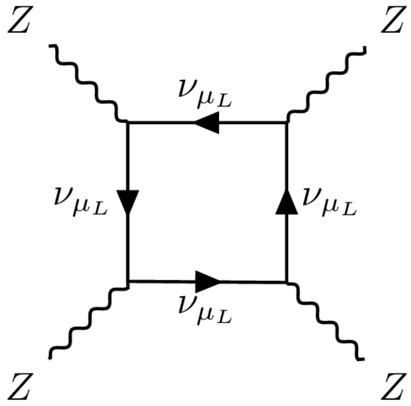
Domain: BSM Models

- **Quantum Field Theory**
  - 4-dimensional Minkowski space-time
  - Spin 0, ½, 1
  - Model building utilities
- **Group theory**
  - Semi-simple Lie groups (SU(N), SO(N), Sp(N), E_6, E_7, E_8, F_4, G_2)
  - Representation theory
  - Algebra generators
  - Simplifications, traces

$$\mathcal{L} = -\frac{1}{4} F_{\mu\nu} F^{\mu\nu} + \left(D^{\mu} H\right)^{\dagger} D_{\mu} H + \mu H^{\dagger} H - \lambda \left(H^{\dagger} H\right)^{2},$$

Domain: Calculations

- Fully **symbolic and automated**
- Up to 5 external particles, at **1-loop**
- **Amplitudes, squared amplitudes, Wilson coefficients**
- **Simplifications**
  - Dirac algebra
  - Group algebra
  - Tensor reduction for momentum integrals
  - Dimensional regularization
  - Equations of motion (Dirac equation)
  - Definition of abbreviations
  - ...



Software Ecosystem

MARTY Feature	Other providers
Symbolic computations	✸Mathematica
Representation theory (groups, algebras)	LieART✸
Feynman rules calculations	FeynRules,✸ LanHEP
Diagram finding, diagram rendering	FeynArts,✸ CalcHEP/CompHEP, MadGraph5_aMC@NLO
(Squared) amplitude calculation	FORM + FormCalc,✸ CalcHEP/CompHEP, MadGraph5_aMC@NLO
Wilson coefficient calculation	FormFlavor✸
Code generation	FormCalc✸
Spectrum generator generator	SARAH/FlexibleSUSY✸

Design & Architecture

I. Introduction
II. Design & Architecture
III. The User Side
IV. What’s New (+ demo)

Guiding principles

1. **Generality** (as much as possible) i.e. not specialized for any of the following:

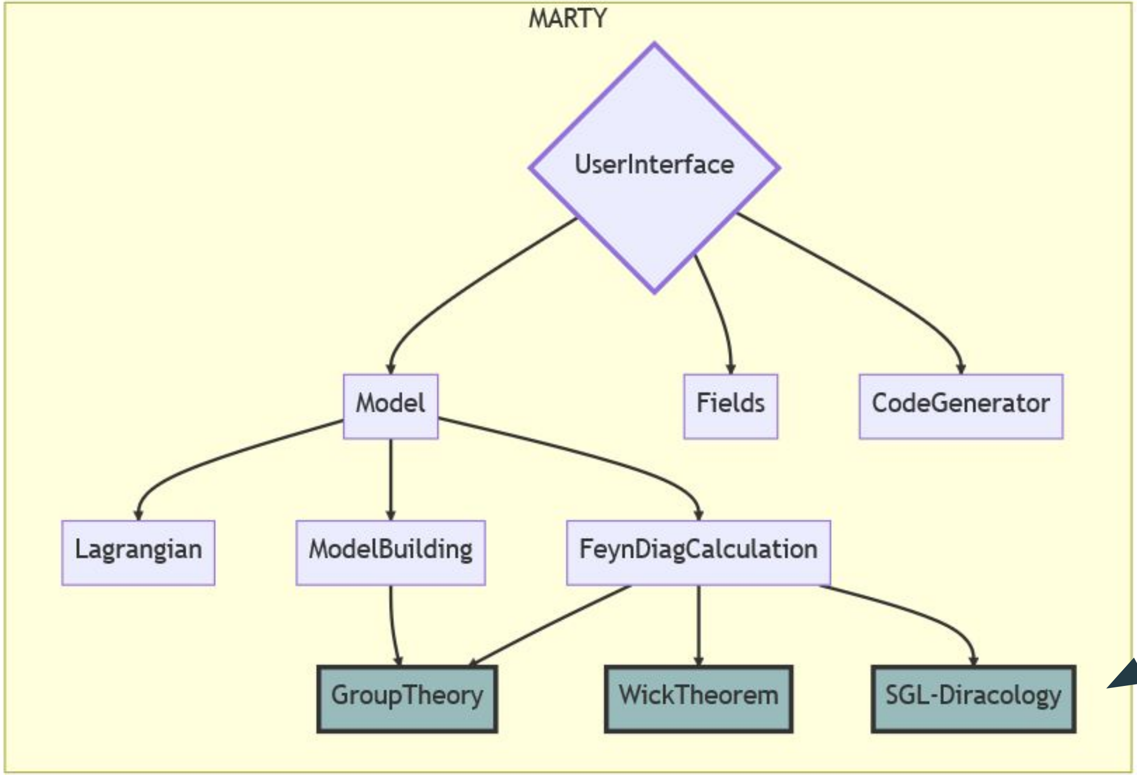
- BSM scenarios
- Particle types (e.g. spin)
- Process types (decay, 2 ⟶ 2, tree-level, one-loop)

2. **Independence**

- Fully free and open-source code
- Get rid of Mathematica
- Implement a built-in symbolic computation library

3. **Software development standards**

Architecture (simplified)



CSL (symbolic manipulation module) is not shown because ubiquitous

Standalone (or almost) calculation modules

The User Side

I. Introduction
II. Design & Architecture
III. The User Side
IV. What’s New (+ demo)

Get Started

- **Website** (Get Started section: https://marty.in2p3.fr/gettingStarted.html)
- **Examples** in the github repo (https://github.com/docbrown1955/marty-public)
- Suggestions are (very) welcome !

Documentation - How to go further

- **Examples** (https://github.com/docbrown1955/marty-public/tree/master/examples)
- **System tests**
  - MARTY programs in src/ (https://github.com/docbrown1955/marty-public/tree/master/tests/system/src)
  - Numerical app in libsrc/(https://github.com/docbrown1955/marty-public/tree/master/tests/system/libsrc)
- **The manual** (main documentation entry point): https://marty.in2p3.fr/doc/marty-manual.pdf

What's new

I. Introduction
II. Design & Architecture
III. The User Side
IV. What's New (+ demo)

Tree-level widths (v1.5)

- Available for **all MARTY models**
- Decay widths
  - All particles, on demand
  - Tree-level calculation at least
- Generated width function
  - Sums over partial widths
  - Mass threshold mechanism integrated
- Integrated in the spectrum generator

General principle of MARTY's libraries

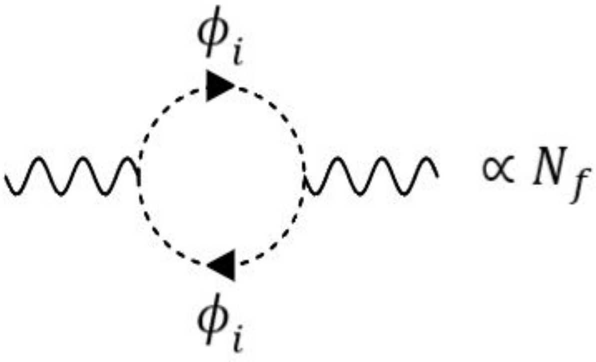
```
// Define input parameters
param_t params;
params.M_W = 80.379;
params.theta_W = 0.49;
// ...

// Calculate the spectrum (optional)
updateSpectrum(params);

// Evaluate numerically the theoretical
// quantities generated by MARTY
cout << f(params) << endl;
cout << g(params) << endl;
```

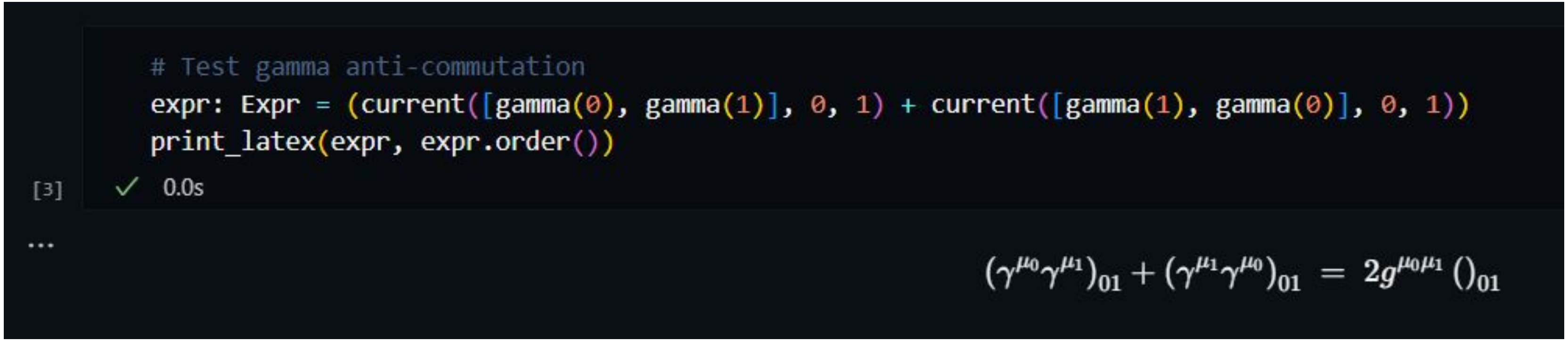
Symbolic flavor dimension (v1.6)

- Breaking change (old MARTY programs may not compile)
- Nf can be let **undefined** (no numerical value)
- No flavor sym. breaking in this case
- **A symbolic Nf is used**
- **Nf is generated as a numerical parameter**



Proof of Concept: Python API for Diracology

- Python interface to MARTY’s module for diracology (SGL)
- Non-official (git branch: https://github.com/docbrown1955/marty-public/tree/api/gamma)
- Demo !



```
# Test gamma anti-commutation
expr: Expr = (current([gamma(0), gamma(1)], 0, 1) + current([gamma(1), gamma(0)], 0, 1))
print_latex(expr, expr.order())
```

[3] ✓ 0.0s

...

$$(\gamma^{\mu_0}\gamma^{\mu_1})_{01} + (\gamma^{\mu_1}\gamma^{\mu_0})_{01} \;=\; 2g^{\mu_0\mu_1}\,()_{01}$$

Conclusion: Room for improvements ?

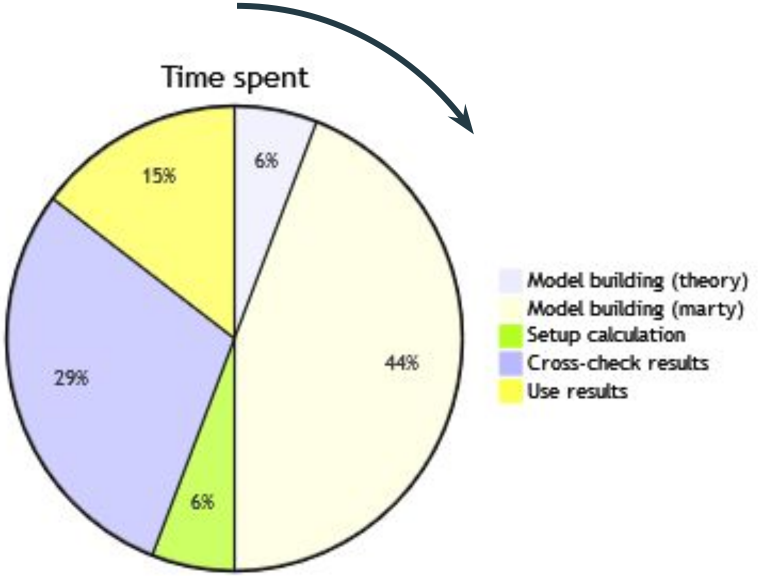
- **User friendly-ness**
  - Hard to understand MARTY’s output
  - Difficult to debug
  - Non-interactive (C++) user interface
- **Interfaces !**
  - Model files (input, output, UFO)
  - Generated numerical libraries unpractical
- **Easy path to get started** and go on on complex models

Questions ?

Backup

General Worlflow

1. **Model building**
    a. Define the model
    b. Implement the model in MARTY
2. **Perform calculations for cross-check**
    a. Implement the calculation
    b. Cross-check
    c. Go back to 1. if necessary
3. **Generate the final results**
    a. Implement the calculation
    b. Exploit the results



(disclaimer: my personal feeling)