FormCalc 7

Thomas Hahn

Max-Planck-Institut für Physik
München

From microsoft.com/en-us/windows7:

Why get Version 7?

- To simplify everyday tasks
- To work the way you want
- To do new things

Automated Diagram Evaluation

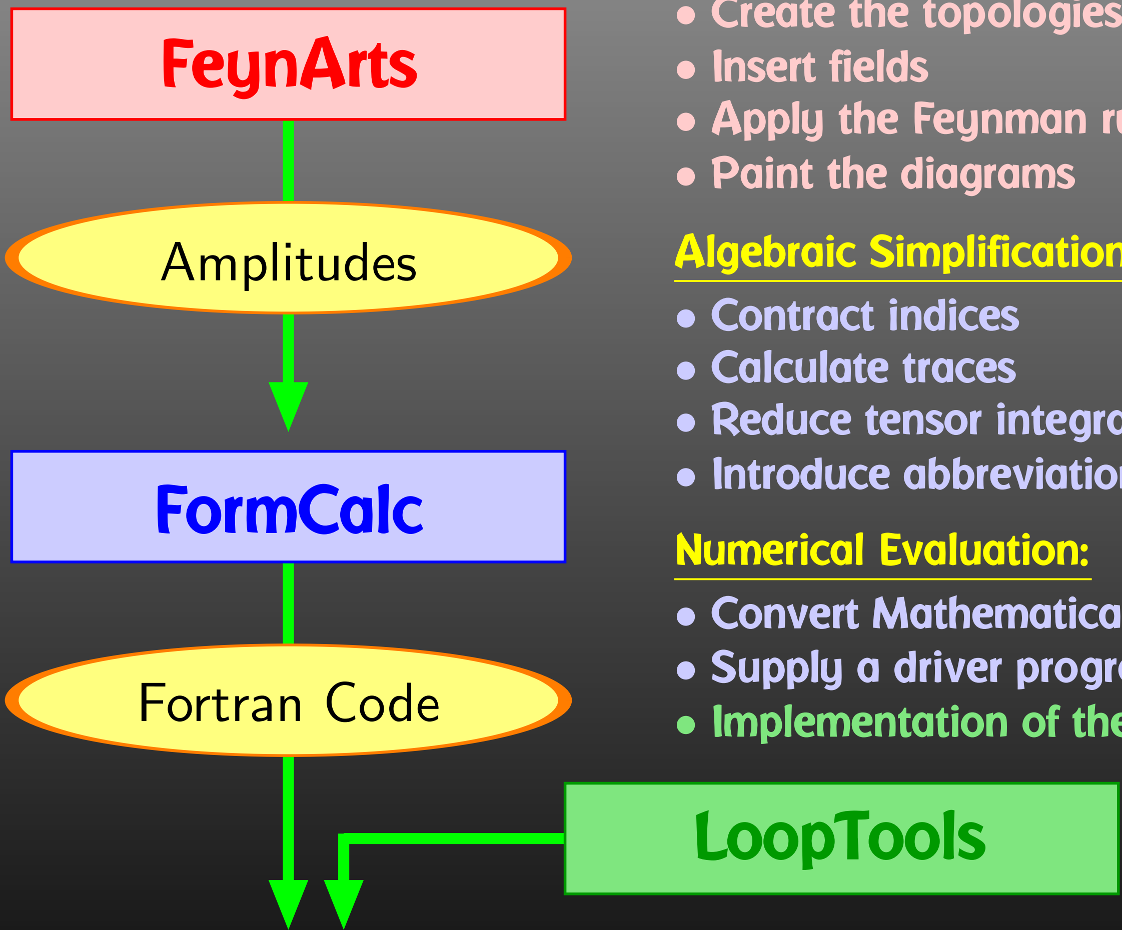


Diagram Generation:

- Create the topologies
- Insert fields
- Apply the Feynman rules
- Paint the diagrams

Algebraic Simplification:

- Contract indices
- Calculate traces
- Reduce tensor integrals
- Introduce abbreviations

Numerical Evaluation:

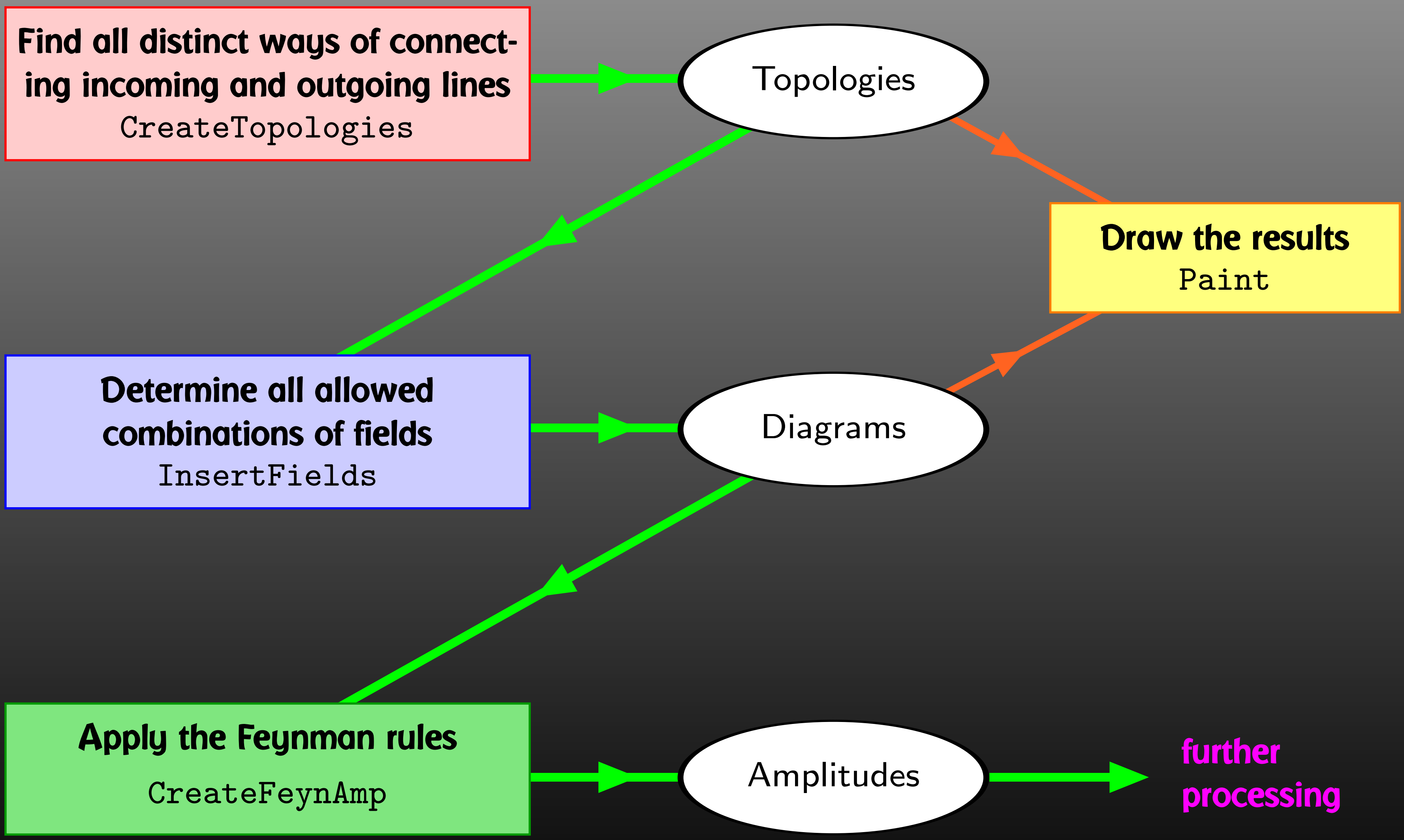
- Convert Mathematica output to Fortran code
- Supply a driver program
- Implementation of the integrals

Symbolic manipulation (Computer Algebra) for the structural and algebraic operations.

Compiled high-level language (Fortran) for the numerical evaluation.

$|\mathcal{M}|^2 \longrightarrow$ **Cross-sections, Decay rates, . . .**

FeynArts



Find all distinct ways of connecting incoming and outgoing lines
CreateTopologies
Topologies
Determine all allowed combinations of fields
InsertFields
Diagrams
Apply the Feynman rules
CreateFeynAmp
Amplitudes
further processing
Draw the results
Paint

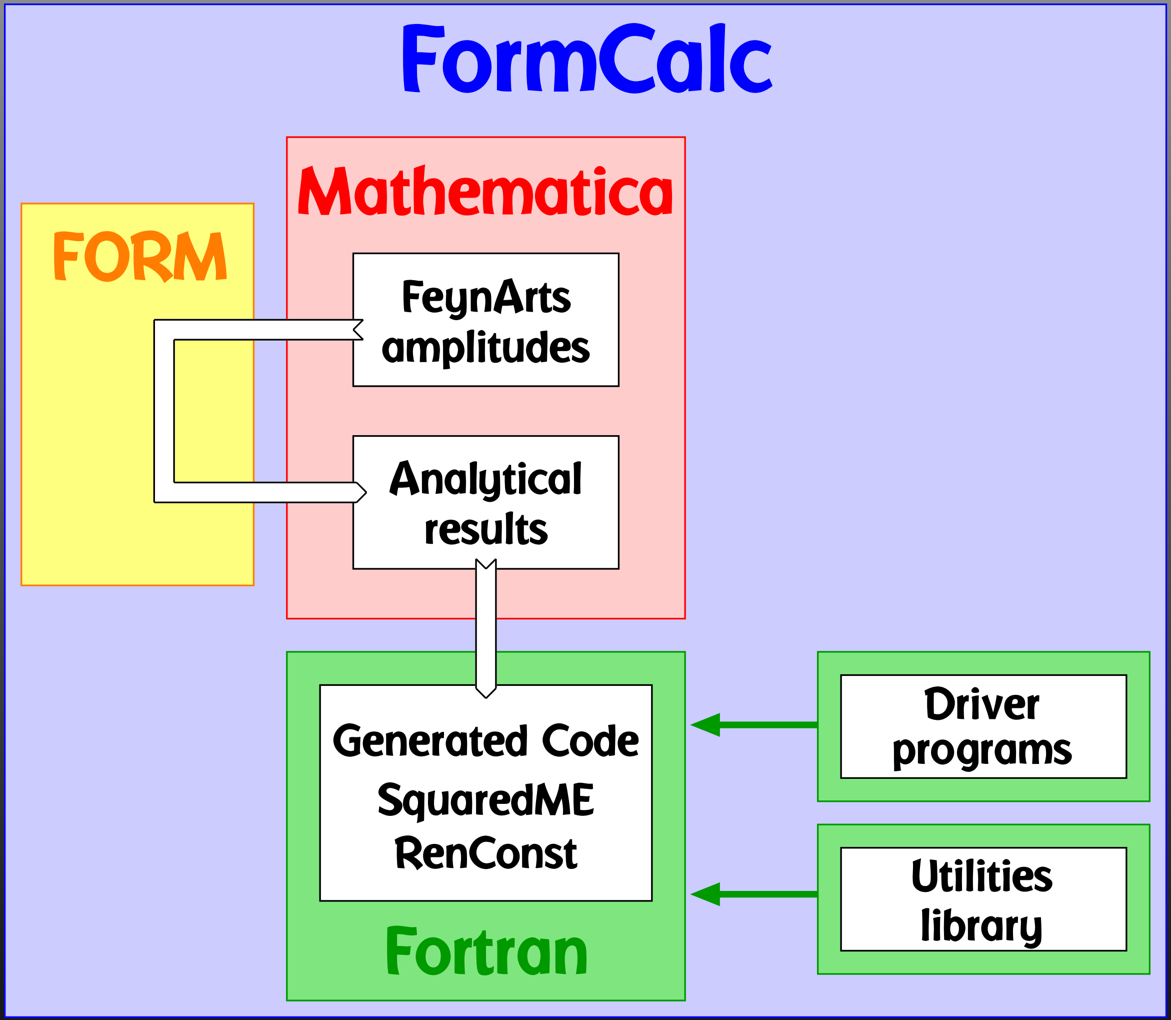
Algebraic Simplification

The amplitudes of `CreateFeynAmp` are in no good shape for direct numerical evaluation.

A number of steps have to be done analytically:

- contract indices as far as possible,
- evaluate fermion traces,
- perform the tensor reduction,
- add local terms arising from D·(divergent integral) (dim reg + dim red),
- simplify open fermion chains,
- simplify and compute the square of SU(N) structures,
- "compactify" the results as much as possible.

FormCalc Internals



FormCalc 7

New Features:

- Analytic tensor reduction,
- Unitarity methods (OPP),
- Improved code generation,
- Command-line parameters for model initialization, MSSM (SM) initialization via FeynHiggs.

Cuba:

- Built-in Parallelization available.

Analytic Tensor Reduction

Work done in collaboration with S. Agrawal.

Passarino-Veltman reduction is still useful. So far:

- introduction of tensor coefficients in FormCalc, e.g.

$$\int \mathrm{d}^4 q \, \frac{q_\mu q_\nu}{D_0 D_1} \sim B_{\mu\nu} = g_{\mu\nu} B_{00} + p_\mu p_\nu B_{11}$$

- complete reduction to scalars only numerically in LoopTools.

Available now: Analytic Reduction in FormCalc.

```
CalcFeynAmp[..., PaVeReduce -> True]
```

Analytic Tensor Reduction

Reduction formulas from Denner & Dittmaier, hep-ph/0509141. Not straightforward to implement in FORM.

Apart from analytic considerations, this is useful e.g. for low-energy observables, where small momentum transfer may lead to **numerical instabilities in numerical reduction,** as in:

$$B_\mu = p_\mu B_1 \quad \text{for} \quad p \to 0$$

Unless FormCalc finds a way to cancel it immediately, the **inverse Gram determinant appears wrapped in `IGram`** in the output, so is available for further modifications.

Unitarity Methods

Work done in collaboration with E. Mirabella.

We employ the **OPP (Ossola, Papadopoulos, Pittau) methods** as implemented in the two libraries **CutTools** and **Samurai.**

Instead of introducing tensor coefficients, the **numerator is put into a subroutine** which is **sampled by the OPP function,** as in:

$$\varepsilon_1^\mu \varepsilon_2^\nu B_{\mu\nu}(p, m_1^2, m_2^2) = B_{\text{cut}}(2, N, p, m_1^2, m_2^2)$$

where

$$N(q_\mu) = (\varepsilon_1 \cdot q)\,(\varepsilon_2 \cdot q)$$

Unitarity Methods

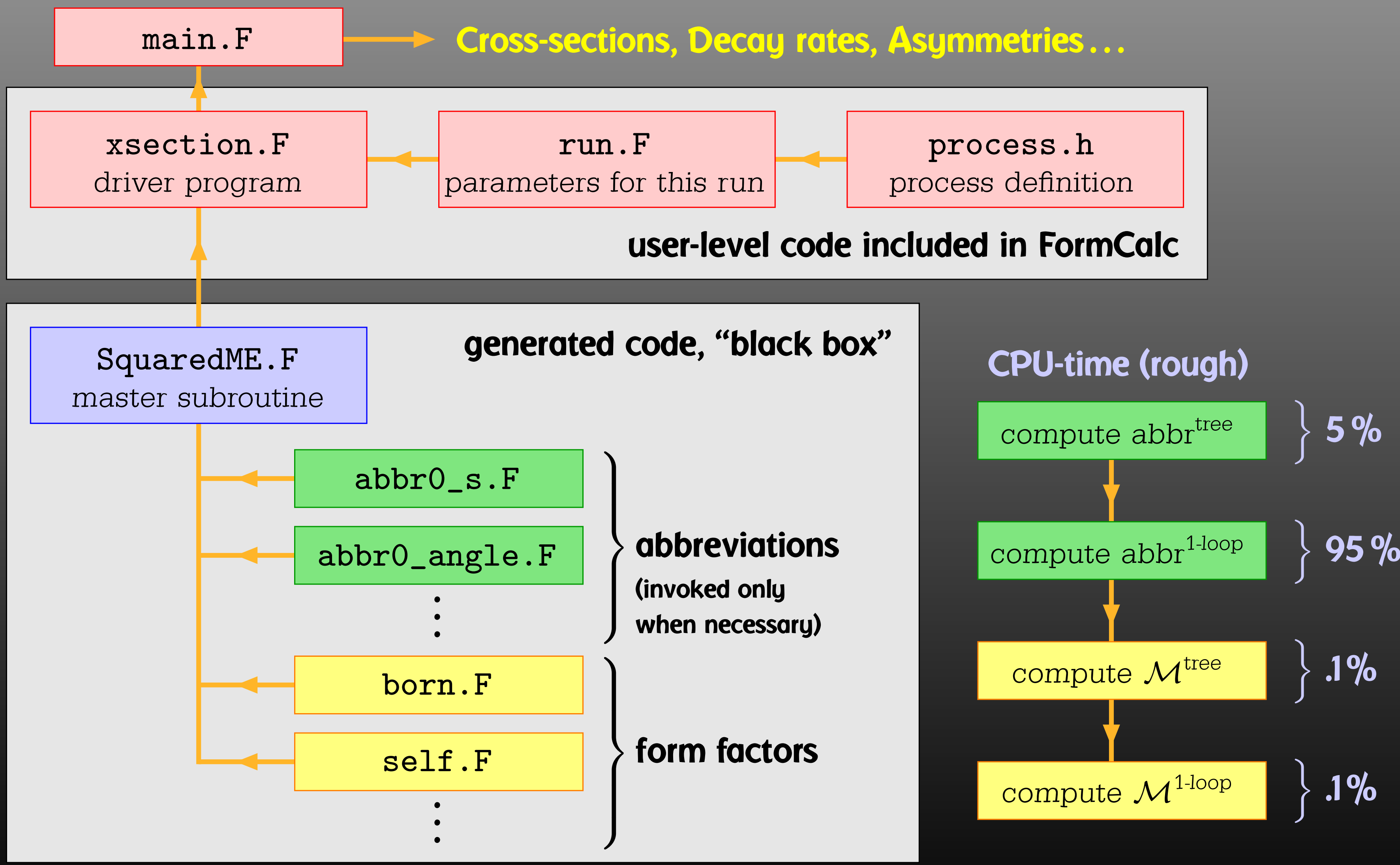
So far tested on a handful of $2 \to 2$ and $2 \to 3$ processes, get agreement to about 10 digits.

Performance somewhat wanting as of now, Passarino–Veltman beats OPP hands down in the processes we looked at.

Currently optimizing performance:

- Option to specify the N in N-point up to which Passarino–Veltman is used, above OPP.
- Minimizing OPP calls to reduce sampling effort – work in progress.
- Already looked into tweaking caching of loop integrals, but pointless: lower-N integrals also needed by OPP.

Numerical Evaluation in Fortran 77



Code generation

Currently: Output in Fortran 77.
Code generator is rather sophisticated by now, e.g.

- Expressions too large for Fortran are split into parts, as in

```
var = part1
var = var + part2
...
```

- High level of optimization, e.g. common subexpressions are pulled out and computed in temporary variables.
- Many ancillary functions make code generation versatile and highly automatable, such that the resulting code needs few or no changes by hand.

Improvements in Code Generation

- Working on **Output in C,** makes integration into C/C++ codes easier and allows for GPU programming.
- Main subroutine `SquaredME.F` is now **sectioned by comments,** to aid **automated substitution** e.g. with `sed`, for example:

```
* BEGIN VARDECL
...
* END VARDECL
```

- Introduced **data types** `Real` **and** `Complex` for better abstraction, can e.g. be changed to different precision.

Command-line parameters for model initialization

Extension of command-line argument parsing:

```
run :arg1 :arg2 ... uuuu 0,1000
```

The ':'-arguments are passed to model initialization code.

Internal routines in `xsection.F` accordingly have additional parameters `argv`, `argc`.

Model Initialization through FeynHiggs

- `model_fh.F` uses **FeynHiggs as Frontend for FormCalc-generated code:**

```
run :fhparameterfile :fhflags uuuu 0,1000
```

- FeynHiggs initializes MSSM (SM) parameters and passes them to FormCalc code.
- No duplication of initialization code.
- Parameters consistent between Higgs-mass computation and cross-section calculation.
- Needs FeynHiggs 2.8.1 or above.

Cuba Parallelization

New version Cuba 3 features built-in parallelization:

- Just **set environment variable** `CUBACORES` to the number of cores one wishes to utilize.
- Uses `fork`/`wait`, with back-communication of results through pipes, thus **no need to write reentrant integrand function,** works identically in Fortran and C/C++.
- Effective on multi-core machines only, no parallelization across network.
- Only the sampling function is presently parallelized, which is the optimal solution for Vegas and Suave, but sub-optimal Divonne and Cuhre. Benchmarking and optimization for the latter is work in progress.

Summary

New Features in FormCalc 7: feynarts.de/formcalc

- Analytic tensor reduction in `CalcFeynAmp`,
- Unitarity (OPP) methods using either the Samurai or CutTools library,
- Improved code generation,
- Command-line parameters for model initialization,
- Initialization of MSSM parameters via FeynHiggs.

Cuba: feynarts.de/cuba

- Built-in Parallelization available simply by setting an environment variable.