O'Mega: An Optimizing Matrix Element Generator

Thorsten Ohl

Darmstadt University of Technology, Schloßgartenstraße 9, 64289 Darmstadt, Germany
`ohl@hep.tu-darmstadt.de`

Abstract. I sketch the architecture of *O'Mega*, a new optimizing compiler for tree amplitudes in quantum field theory. O'Mega generates the most efficient code currently available for scattering amplitudes for many polarized particles in the standard model. A complete infrastructure for physics beyond the standard model is provided.

INTRODUCTION

Current and planned experiments in high energy physics can probe processes with many tagged—potentially polarized—particles in the final state. The combinatorial explosion of the number of Feynman diagrams contributing to scattering amplitudes for many external particles calls for the development of more compact representations that translate well to efficient and reliable numerical code. In gauge theories, strong numerical cancellations in a redundant representation built from necessarily gauge dependent Feynman diagrams lead to a loss of numerical precision, stressing further the need for eliminating redundancies.

Due to the large number of processes that have to be studied in order to unleash the potential of modern experiments, the construction of these representations must be possible algorithmically on a computer and should not require human ingenuity for each new application.

O'Mega (1) is a compiler for tree-level scattering amplitudes that satisfies these requirements. O'Mega is independent of the target language and can support code in any programming language for which a simple output module has been written. To support a physics model, O'Mega requires as input only the Feynman rules and the relations among coupling constants.

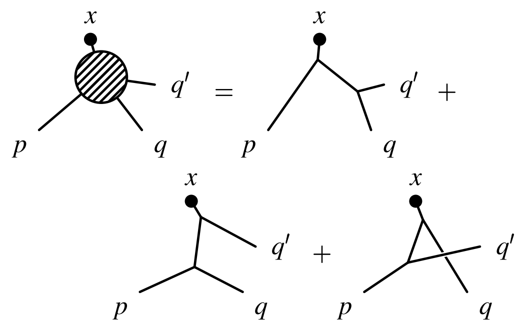
Similar to earlier numerical approaches (2, 3), O'Mega reduces the growth in calculational effort from a factorial of the number of particles to an exponential. The symbolic nature of O'Mega, however, increases its flexibility. Indeed, O'Mega can emulate both (2, 3) and produces code that is empirically at least twice as fast.

1POWS AND KEYSTONES

One Particle Off-shell Wave functions (1POWs) are obtained from Greensfunctions by applying the LSZ reduction formula to all but one line:

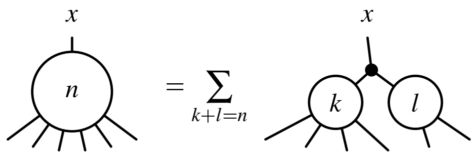
$$W^{q_1,\ldots,q_m}_{p_1,\ldots,p_n}(x) = \langle \phi(q_1),\ldots,\phi(q_m); out|\Phi(x)|\phi(p_1),\ldots,\phi(p_n); in\rangle \,. \quad (1)$$

The 1POW $W^{q,q'}_p(x) = \langle \phi(q),\phi(q'); \mathrm{out}|\Phi(x)|\phi(p); \mathrm{in}\rangle$ in lowest order of ϕ^3-theory, is given—for illustration—by



(2)

At tree-level, the set of all 1POWs for a given set of external momenta can be constructed recursively (4)

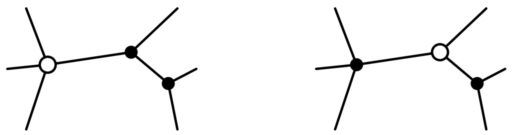


, (3)

where the sum extends over all partitions of the set of n momenta. For all quantum field theories, there are—well defined, but not unique—sets of *Keystones* K (1) such that the sum of tree Feynman diagrams can be expressed as a sparse sum of products of 1POWs without double counting. In a theory with only cubic couplings this is expressed as

$$T = \sum_{i=1}^{F(n)} D_i = \sum_{k,l,m=1}^{P(n)} K^3_{f_kf_lf_m}(p_k,p_l,p_m) W_{f_k}(p_k) W_{f_l}(p_l) W_{f_m}(p_m) \,. \quad (4)$$

The non-trivial problem of avoiding the double counting of diagrams like (the circle denotes the keystone)



has been solved for general theories with vertices of arbitrary degrees (1).

The number of distinct momenta that can be formed from n external momenta is $P(n) = 2^{n-1} - 1$. Therefore, the number of tree 1POWs grows exponentially with the number of external particles and not with a factorial, as the number of Feynman diagrams $F(n) = (2n-5)!! = (2n-5) \cdot \ldots 5 \cdot 3 \cdot 1$. The equations sketched in Eqs. (3) and (4) for cubic couplings can be generalized to vertices of any order (1).

Even for vector particles and to all orders in renormalized perturbation theory, the 1POWs are 'almost' physical objects and satisfy simple Ward identities in unbroken gauge theories

$$\frac{\partial}{\partial x_\mu} \left\langle \text{out} \middle| A_\mu(x) \middle| \text{in} \right\rangle_{\text{amp.}} = 0 \tag{5}$$

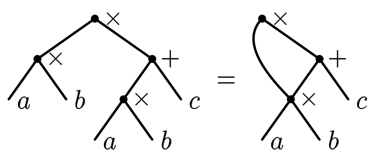
and well as in spontaneously gauge theories

$$\frac{\partial}{\partial x_\mu} \left\langle \text{out} \middle| W_\mu(x) \middle| \text{in} \right\rangle_{\text{amp.}} = \xi_W m_W \left\langle \text{out} \middle| \phi_W(x) \middle| \text{in} \right\rangle_{\text{amp.}} \tag{6}$$

in R_ξ-gauge. The code for matrix elements can optionally be instrumented by O'Mega with numerical checks of these Ward identities for intermediate lines.

DIRECTED ACYCLICAL GRAPHS

The algebraic expression for the tree-level scattering amplitude in terms of Feynman diagrams is itself a tree. The much slower growth of the set of 1POWs compared to the set of Feynman diagrams shows that this representation is extremely redundant. In this case, *Directed Acyclical Graphs* (DAGs) provide a more efficient representation, as illustrated by a trivial example

$ab(ab+c) =$  , (7)

where one multiplication is saved. The replacement of expression trees by equivalent DAGs is part of the repertoire of optimizing compilers, known as *common subexpression elimination*. Unfortunately, this approach fails for typical expressions appearing in quantum field theory, because of the combinatorial growth of space and time required to find an almost optimal factorization.

However, the recursive definition in Eq. (3) allows to construct the DAG of the 1POWs in Eq. (4) *directly* (1), without having to construct and factorize the Feynman diagrams explicitely.

As mentioned above, there is more than one consistent prescription for constructing the set of keystones (1). The symbolic expressions constructed by O'Mega contain the symbolic equivalents of the numerical expressions computed by (2) (maximally symmetric keystones) and (3) (maximally asymmetric keystones) as special cases.

ALGORITHM

By virtue of their recursive construction in Eqs. (3), tree-level 1POWs form a DAG and the problem is to find the smallest DAG that corresponds to a given tree, (i. e. an given sum of Feynman diagrams). O'Mega's algorithm proceeds in four steps

Grow: starting from the external particles, build the tower of *all* 1POWs up to a given height (the height is less than the number of external lines for asymmetrical keystones and less than half of that for symmetrical keystones) and translate it to the equivalent DAG D.

Select: from D, determine *all* possible *flavored keystones* for the process under consideration and the 1POWs appearing in them.

Harvest: construct a sub-DAG $D^* \subseteq D$ consisting *only* of nodes that contribute to the 1POWs appearing in the flavored keystones.

Calculate: multiply the 1POWs as specified by the keystones and sum the keystones.

By construction, the resulting expression contains *no* more redundancies and can be translated to a numerical expression. In general, asymmetrical keystones create an expression that is smaller by a few percent than the result from symmetrical keystones, but it is not yet clear which approach produces the numerically more robust results.

IMPLEMENTATION

The O'Mega compiler is implemented in O'Caml (5), a functional programming language of the ML family with a very efficient, portable and freely available implementation, that can be bootstrapped on all modern computers in a few minutes.

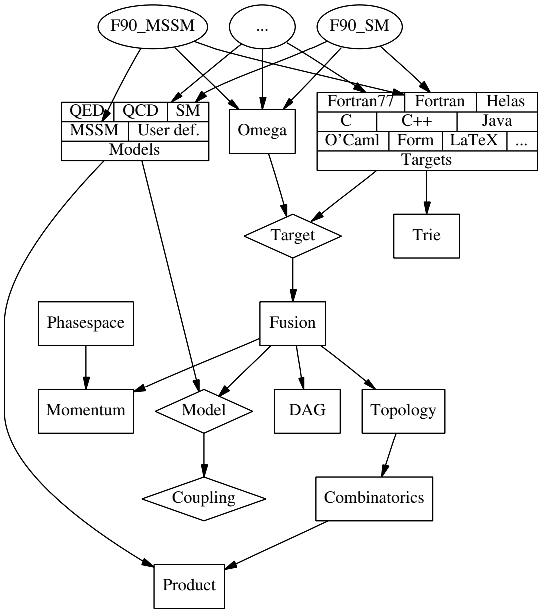


FIGURE 1. Module dependencies in O'Mega. The diamond shaped nodes denote abstract signatures defining functor domains and co-domains. The rectangular boxes denote modules and functors, while oval boxes stand for example applications.

The powerful module system of O'Caml allows an efficient and concise implementation of the DAGs for a specific physics model as a functor application (1). This functor maps from the category of trees to the category of DAGs and is applied to the set of trees defined by the Feynman rules of any model under consideration.

The implementation is concise and efficient simultaneously by exploiting the virtues of persistent data structures (6). Typically, the resources consumed by O'Mega are only a small fraction of the resources required by the compiler for the target language.

The module system of O'Caml has been used to make the combinatorial core of O'Mega demonstrably independent from the specifics of both the physics model and the target language (1), as shown in Figure 1. A Fortran90/95 backend has been realized first, backends for C++ and Java will follow. The complete electroweak standard model has been implemented (the treatment of interfering color amplitudes is still incomplete). Majorana fermions, required by supersymmetric field theories, are available (using (8)) and the MSSM is in preparation.

As mentioned above, the compilers for the target programming language are the slowest step in the generation of executable code. On the other hand, the execution speed of the code is limited by non-trivial vertex evaluations for vectors and spinors, which need $O(10)$ complex multiplications. Therefore, an *O'Mega Virtual Machine* can challenge native code and avoid compilations.

APPLICATIONS

The code generated by the Fortran90/95 backend is the most efficient code available for polarized scattering amplitudes for many particles. The results have been compared with MADGRAPH (7) for many standard model processes and numerical agreement at the level of 10^{-11} has been found with double precision floating point arithmetic. O'Mega generated amplitudes are used in the omnipurpose event generator generator WHIZARD (9). The first complete experimental study of vector boson scattering in six fermion production for linear collider physics (10) has been facilitated by O'Mega and WHIZARD.

Acknowledgments

I thank my collaborators Mauro Moretti and Jürgen Reuter. I thank Wolfgang Kilian for valuable suggestions and for "early adoption" of O'Mega. This research is supported by the German Bundesministerium für Bildung und Forschung (05 HT9RDA) and Deutsche Forschungsgemeinschaft (MA 676/6-1).

REFERENCES

1. M. Moretti, T. Ohl, and J. Reuter, (to be published), `http://heplix.ikp.physik.tu-darmstadt.de/~ohl/omega/`.
2. F. Caravaglios and M. Moretti, Z. Phys. **C74** (1997) 291.
3. A. Kanaki and C. Papadopoulos, DEMO-HEP-2000/01, hep-ph/0002082, February 2000.
4. H. Murayama, I. Watanabe, and K. Hagiwara, KEK Report 91-11, January 1992.
5. Xavier Leroy, *The Objective Caml system, release 3.0, documentation and user's guide*, Technical Report, INRIA, 2000 (`http://caml.inria.fr/ocaml/`).
6. Chris Okasaki, *Purely Functional Data Structures*, Cambridge University Press, 1998.
7. T. Stelzer and W.F. Long, Comput. Phys. Commun. **81** (1994) 357.
8. A. Denner, H. Eck, O. Hahn, and J. Küblbeck, Phys. Lett. **B291** (1992) 278; Nucl. Phys. **B387** (1992) 467.
9. W. Kilian, (to be published), `http://www-ttp.physik.uni-karlsruhe.de/~kilian/whizard/`.
10. R. Chierici and S. Rosati, (to be published).