Math.Comput.Sci. (2015) 9:391–408
DOI 10.1007/s11786-015-0236-y

Mathematics in Computer Science

Graph Grammars, Insertion Lie Algebras, and Quantum Field Theory

Matilde Marcolli · Alexander Port

Received: 11 March 2015 / Accepted: 14 May 2015 / Published online: 13 August 2015

Abstract Graph grammars extend the theory of formal languages in order to model distributed parallelism in theoretical computer science. We show here that to certain classes of context-free and context-sensitive graph grammars one can associate a Lie algebra, whose structure is reminiscent of the insertion Lie algebras of quantum field theory. We also show that the Feynman graphs of quantum field theories are graph languages generated by a theory dependent graph grammar.

Keywords Lie algebras · pre-Lie operators · Graph Languages · Production rules · Feynman graphs

Mathematics Subject Classification 68Q45 · 68Q42 · 81T18

1 Introduction

Graph Languages and Graph Grammars were introduced in theoretical computer science as an extension of the theory of formal languages (linear languages), in order to model various types of parallelism in computation, [10,11,19,21]. Instead of replacing nonterminal symbols with combinations of nonterminals and terminals in a linear string of characters, the production rules of graph grammars replace a subgraph of a graph with a new graph. The latter is obtained either by gluing along a common subgraph, or by first performing an excision of a subgraph and then replacing it with a new graph. An analog of the Chomsky hierarchy of grammars exists for Graph Languages, see [19]. In particular, the context-free Graph Grammars are those where the left-hand-side of the production rules is always a single vertex. Namely, no "context" in the graph is taken into consideration in deciding when a production rule can be applied: it just applies to any vertex. In this context-free case the production rules then consist of inserting a new graph at a vertex of another graph. This operation is reminiscent of the insertion operation that defines the Lie algebras of Feynman graphs in the algebraic approach to renormalization in quantum field theory pioneered in [5] (see also [6,7,9]). In this paper we show that, indeed, to certain classes of Graph Grammars (both context-free and context-sensitive) it is possible to associate a Lie algebra, obtained by constructing a pre-Lie insertion operator

M. Marcolli (✉) · A. Port
Mathematics Department, Caltech, 1200 E. California Blvd, Pasadena, CA 91125, USA
e-mail: matilde@caltech.edu

A. Port
e-mail: aport@caltech.edu

using the production rules of the grammar. We also show that the Feynman graphs of a given quantum field theory are a graph language in the sense of the theory of formal languages. This provides a new class of examples of graph languages, in addition to those arising in the context of computer science (such as FFT networks, Petri nets, distributed parallelism), see the articles in [11] for several examples. Relations between the formalism of algebraic renormalization in quantum field theory and aspects of the theory of computation in theoretical computer science have already been investigated in [18], see also the formulation of Dyson–Schwinger equations in the Hopf algebra of flow charts in [8]. it would be interesting to see if a theory of Dyson–Schwinger equations can be formulated for Graph Languages, using the Lie theoretic approach of [12].

1.1 The Insertion Lie Algebra of Quantum Field Theory

In perturbative quantum field theory, one computes expectation values as a formal series of Feynman amplitudes labeled by Feynman graphs. These graphs are finite and the allowed valencies are constrained to match the exponents in the interaction monomial in the Lagrangian of the field theory. Graphs have a number of internal edges (connecting pairs of vertices) and external edges (half edges). The corresponding Feynman amplitude is a finite dimensional integral over a space of momenta flowing through the graph, with assigned external momenta carried by the external edges, and with conservation laws at the vertices. These Feynman integrals are typically divergent, which leads to the crucial problem of renormalization. The goal of a renormalization procedure is a consistent extraction of finite values from all these integrals that takes into account the combinatorics of how divergent subgraphs are nested inside larger graphs. Since the work of Kreimer [15] and Connes–Kreimer [5], it has become clear that the renormalization procedure can be formulated algebraically in terms of a Hopf algebra of Feynman graphs. The algebraic Feynman rules are seen as algebra homomorphisms to a target commutative algebra determined by a choice of regularization procedure, and endowed with a "pole-subtraction" operation (Rota–Baxter algebra). See §1 of [7] for an overview.

The Hopf algebra $\mathcal{H}$ of Feynman graphs is a graded connected commutative Hopf algebra generated by the 1PI Feynman graphs of the given quantum field theory. The 1PI (one-particle irreducible) condition means that the graphs are connected and cannot be disconnected by removal of a single edge. A standard argument in quantum field theory reduces the combinatorics of Feynman graphs to the connected case, and further to the 1PI case, see [13]. The coproduct in the Hopf algebra is not co-commutative. It is given by

$$\Delta(G) = G \otimes 1 + 1 \otimes G + \sum \gamma \otimes G/\gamma, \tag{1.1}$$

where the sum is over all the (not necessarily connected) subgraphs $\gamma \subset G$, such that the quotient graph G/γ (obtained by shrinking each component of γ to a vertex) is a 1PI Feynman graph of the theory. The Hopf algebra is dual to a pro-unipotent affine group scheme that is entirely determined by its Lie algebra. Connes and Kreimer gave a very explicit geometric description of this *insertion Lie algebra* [6] (see also §1 of [7]). On the vector space spanned by all 1PI Feynman graphs of the theory, one can define a Lie bracket by setting

$$[G_1, G_2] = \sum_v G_1 \circ_v G_2 - \sum_{v'} G_2 \circ_{v'} G_1, \tag{1.2}$$

where the sums are over all vertices in G_1 and G_2. The expression $G_1 \circ_v G_2$ denotes the graph resulting from the insertion of G_2 at the vertex v of G_1. One can insert one graph into another by matching external edges to the edges incident at the vertex. In the sum one counts all the possible inequivalent ways in which the graph can be inserted at the given vertex. This bracket indeed satisfies the Jacobi identity and defines a Lie algebra, which can be related to the primitive elements in the dual Hopf algebra of Feynman graphs [6] (see also §1 of [7]). A detailed survey of the use of Lie algebra methods in Quantum Field Theory can be seen in [9]. The language of Lie algebras in Quantum Field Theory provides an elegant formulation of the Dyson–Schwinger equations (the quantum equations of motion of the theory), and a general method for solving them in the Lie algebra of Feynman graphs [12].

2 Graph Grammars and Lie algebras

2.1 Two Descriptions of Graphs

It is convenient to consider two slightly different ways of assigning the data of a finite graph. The first is the one most commonly used in Combinatorics, while the second is more frequently used in Physics.

2.1.1 Version 1

A graph G consists of a set of vertices $V(G)$ and a set of edges $E(G)$ together with a boundary map $\partial : E(G) \to V(G) \times V(G)$ assigning to an edge $e \in E(G)$ its (unordered) pair of boundary vertices $\partial(e) = \{v_1, v_2\}$. The graph can have looping edges if we allow $v_1 = v_2$ and it can have multiple parallel edges if $\partial^{-1}(v_1, v_2)$ can consist of more than one element. If the graph G is oriented (directed) then the boundary map consists of two maps (source and target) $s, t : E(G) \to V(G)$. A system of vertex and edge labeling consists of two sets Σ_V, Σ_E of vertex and edge labels, respectively, and functions $L_{V,G} : V(G) \to \Sigma_V$ and $L_{E,G} : E(G) \to \Sigma_E$.

2.1.2 Version 2

A graph G consists of a set $C(G)$ of *corollas* (vertices v with valence $\mathrm{val}(v)$ with $\mathrm{val}(v)$ half-edges attached to it) and an involution $\mathcal{I} : \mathcal{F}(G) \to \mathcal{F}(G)$ on the set $\mathcal{F}(G)$ of all half-edges (flags) attached to all the corollas. The set $E_{int}(G)$ of internal edges of G corresponds to all the pairs $\{f, f'\}$ with $f \neq f'$ in $\mathcal{F}(G)$ and with $f' = \mathcal{I}(f)$. The set $E_{ext}(G)$ of external (half)edges of G consists of all the $f \in \mathcal{F}(G)$ such that $\mathcal{I}(f) = f$. A labeling system is given by a set $\Sigma_{\mathcal{F}}$ of flag labels and a set Σ_V of vertex labels together with maps $L_{\mathcal{F},G} : \mathcal{F}(G) \to \Sigma_{\mathcal{F}}$ and $L_{V,G} : C(G) \to \Sigma_V$ with $L_{\mathcal{F},G} \circ \mathcal{I} = L_{\mathcal{F},G}$.

Notice that the objects defined by Version 1 and Version 2 are not exactly the same: the graphs defined in Version 2 can have external (unmatched) half-edges, unlike the graphs in Version 1, which only have full edges.

2.2 Insertion Graph Grammars

Using the first description of graphs, we define an Insertion Graph Grammar as follows.

Definition 2.1 An Insertion Graph Grammar consists of data

$$(N_E, N_V, T_E, T_V, P, G_S)$$

where the set of edge labels of graphs is $\Sigma_E = N_E \cup T_E$, with N_E the nonterminal symbols and T_E the terminal symbols, and the set of vertex labels is given by $\Sigma_V = N_V \cup T_V$, with non-terminal and terminal symbols given respectively by N_V and T_V. The start graph is G_S and P is a finite set of production rules of the form $P = (G_L, H, G_R)$, with G_L and G_R labelled graphs (respectively, the left-hand-side and the right-hand-side of the production) and with H a labelled graph with isomorphisms.

$$\phi_L : H \xrightarrow{\simeq} \phi_L(H) \subset G_L, \quad \phi_R : H \xrightarrow{\simeq} \phi_R(H) \subset G_R.$$

The isomorphism ϕ_L should be label preserving. The production rule $P = (G_L, H, G_R)$ searches for a copy of G_L inside a given graph G and glues in a copy of G_R by identifying them along the common subgraph H, with new labels matching those of $\phi_R(H)$.

2.2.1 Context-Free Graph Grammars

We recall the notion of context-freeness for graph grammars from [19].

Definition 2.2 An Insertion Graph Grammar as in Definition 2.1 is *context-free* if $G_L = \{v\}$ (hence $H = \{v\}$ also). It is *context-sensitive* if $G_L \neq \{v\}$. In the context-sensitive case G_L is called the *context* of the production rule.

A Chomsky hierarchy for graph grammars is described in [19].

2.2.2 Insertion Graph Grammars and Flags

If we consider the second version of the definition of graphs given above, we can formulate a slightly different notion of Insertion Graph Grammars. For a subgraph $G' \subset G$ the set of external edges $E_{ext}(G'; G)$ is defined as the union of the set $\mathcal{F}_{G'} \cap E_{ext}(G)$ and the set of pairs $(f.f') \in E(G)$ such that only one half-edge in the pair belongs to $\mathcal{F}_{G'}$ while the other belongs to $\mathcal{F}_G \smallsetminus \mathcal{F}_{G'}$.

In this setting, we describe an Insertion Graph Grammar as follows.

Definition 2.3 An Insertion Graph Grammar consists of data $(N_{\mathcal{F}}, N_V, T_{\mathcal{F}}, T_V, P, G_S)$, as in Definition 2.1, with $\Sigma_{\mathcal{F}} = N_{\mathcal{F}} \cup T_{\mathcal{F}}$ the non-terminal and terminal labels for flags. The production rules $P = (G_L, H, G_R)$ are as in Definition 2.1, with the additional requirement that $\phi_L(E_{ext}(H, G_R)) \subset E_{ext}(G_L, G)$ and $\phi_R(E_{ext}(H, G_L)) \subset E_{ext}(G_R)$, where G is any graph the production rule is applied to, with $G_L \subset G$.

The reason for this modified definition is that the notion of gluing of two graphs $G_L \cup_H G_R$ along a common subgraph H is formulated by taking as set of corollas

$$C_{G_L \cup_H G_R} = C_{G_L} \cup_{C_H} C_{G_R},$$

identifying the corollas around each vertex of H in G_L and G_R and then matching half-edges by the involution $\mathcal{I}(f) = f'$ with $f' = \mathcal{I}_L(f)$ when both $f, f' \in \mathcal{F}_{G_L}$, with $f \neq f'$, and $f' = \mathcal{I}_R(f)$ when $f, f' \in \mathcal{F}_{G_R}$, with $f \neq f'$. If $\mathcal{I}_1(f) = f$ and $f \in \phi_L(E_{ext}(H, G_R)) \subset E_{ext}(G_L, G)$ with $f = \phi_L(f')$, when $\mathcal{I}(f) = f'$ and similarly for $\phi_R(E_{ext}(H, G_L)) \subset E_{ext}(G_R)$.

In this setting, because vertices are always endowed with a corolla of half-edges, we cannot state the context-free condition by requiring that $G_L = H = \{v\}$. An appropriate replacement of the context free condition is given by the following.

Definition 2.4 An Insertion Graph Grammar as in Definition 2.3 is *context-free* if $G_L = H = C(v)$, the corolla $C(v)$ of a vertex v, and all the vertices of graphs in the graph language have the same valence.

In the case where graphs contain vertices of different valences, these would still be context-sensitive graph grammars, with the context specified by the valence of $C(v)$.

2.3 Insertion-Elimination Graph Grammars

We consider another variant of the definition of graph grammars, where the production rules consist of *replacing* a subgraph by another one, instead of gluing them along a subgraph. While the version discussed above reflects the notion of graph grammars considered for instance in [19], the version we discuss here reflects the use in other references (see for instance [22]).

In order to formulate this version of graph grammars with the first notion of graphs, we need to define the operation of removal of a subgraph from a graph. Let $G' \subset G$ be a subgraph. Let

$$E_G(G') = \{e \in E(G) \smallsetminus E(G') \mid \partial(e) \cap V(G') \neq \emptyset\}. \tag{2.1}$$

We define $G \smallsetminus G'$ as the subgraph of G with $V(G \smallsetminus G') = V(G) \smallsetminus V(G')$ and with edges $E(G) \smallsetminus (E(G') \cup E_G(G'))$. Thus, for example, removing a vertex $G' = \{v\}$ means removing the vertex v along with its star of edges. We then define Insertion-elimination Graph Grammars as follows.

Definition 2.5 An Insertion-elimination Graph Grammar consists of data

$$\mathcal{G} = (N_E, N_V, T_E, T_V, P, G_S)$$

as in Definition 2.1, where the production rule $P = (G_L, H, G_R)$ acts by searching for a copy of G_L in G, removing $G_L \smallsetminus H$ and replacing it with the graph G_R glued along H.

Using the second description of graphs, the removal of a subgraph $G' \subset G$ is defined by cutting all edges in $E_{ext}(G', G)$ into pairs of half-edges, one attached to G' and one to $G \smallsetminus G'$. Thus, the set of corollas $C(G \smallsetminus G')$ is given by the difference $C(G) \smallsetminus C(G')$ and the set of flags is given by $\mathcal{F}(G \smallsetminus G') = \mathcal{F}(G) \smallsetminus \mathcal{F}(G')$, with involution $\mathcal{I}_{G \smallsetminus G'}(f) = \mathcal{I}_G(f)$ if both f and $\mathcal{I}_G(f)$ are in $\mathcal{F}(G) \smallsetminus \mathcal{F}(G')$ and $\mathcal{I}_{G \smallsetminus G'}(f) = f$ for $f \in \mathcal{F}(G) \smallsetminus \mathcal{F}(G')$ with $\mathcal{I}_G(f) \in \mathcal{F}(G')$. Notice that the two notions of removal of subgraphs differ in the way the edges connecting a vertex of the subgraph to a vertex of the complement are treated: in the first case they are removed, while in the second case a half-edge remains as an external edge of the complement graph. We then have the following formulation.

Definition 2.6 An Insertion-elimination Graph Grammar consists of data

$$\mathcal{G} = (N_{\mathcal{F}}, N_V, T_{\mathcal{F}}, T_V, P, G_S)$$

as in Definition 2.3, with the requirement that

$$E_{ext}(\phi_L(H), G) = E_{ext}(\phi_L(H), G_L) = E_{ext}(\phi_R(H), G_R) = E_{ext}(G_R).$$

The production rule $P(G_L, H, G_R)$ acts by searching for a copy of G_L inside G, removing $G_L \smallsetminus \phi_L(H)$ and replacing it with a copy of $G_R \smallsetminus \phi_R(H)$, by matching the half-edges of $E_{ext}(G_R)$ to the half-edges of $E_{ext}(G_L, G)$.

2.4 Pre-Lie Structures

A (right) pre-Lie structure on a vector space V is a bilinear map

$$\triangleleft : V \otimes V \to V$$

satisfying the identity of associators under the exchange $y \leftrightarrow z$,

$$(x \triangleleft y) \triangleleft z - x \triangleleft (y \triangleleft z) = (x \triangleleft z) \triangleleft y - x \triangleleft (z \triangleleft y), \quad \forall x, y, z \in V. \tag{2.2}$$

A Lie algebra is a vector space V endowed with a bilinear bracket $[\cdot, \cdot]$ satisfying antisymmetry $[x, y] = -[y, x]$ and the Jacobi identity

$$[x, [y, z]] + [z, [x, y]] + [y, [z, x]] = 0, \quad \forall x, y, z \in V. \tag{2.3}$$

A pre-Lie structure determines a Lie algebra by setting

$$[x, y] := x \triangleleft y - y \triangleleft x. \tag{2.4}$$

The pre-Lie identity ensures that the Jacobi identity is satisfied. A detailed survey of occurrences of pre-Lie algebras in geometry, physics, and the theory of formal languages can be found in [4].

One can obtain a group structure from a pre-Lie algebra structure (see [2] and [16]) by considering formal series

$$W(x) = x + \frac{1}{2} x \triangleleft x + \frac{1}{6}(x \triangleleft x) \triangleleft x + \cdots$$

with the multiplication operation

$$W(x) \star W(y) = W(C(x, y)),$$

where $C(x, y)$ is the Baker–Campbell–Hausdorff formula

$$C(x, y) = x + y + \frac{1}{2}[x, y] + \frac{1}{12}([x, [x, y]] + [y, [y, x]]) + \cdots$$

2.5 Lie Algebras of Context-Free Grammars of Directed Acyclic Graphs

Consider the case of a context-free Insertion Graph Grammar $\mathcal{G}$ as in Definitions 2.1 and 2.2, where the start graph G_S is a single vertex and all the graphs G are directed acyclic with a marked (root) source vertex. The production rules are of the form $P(v, v_2, G_2)$, where v_2 is the root vertex of G_2 and v is a vertex of the graph G_1, to which the rule is applied. The resulting graph

$$G_1 \lhd_v G_2 = P(v, v_2, G_2)(G_1) = G_1 \cup_{v \equiv v_2} G_2$$

obtained by applying the production rule to G_1 is also a directed acyclic graph with root vertex the root v_1 of G_1.

Let $\mathcal{V}$ be the vector space spanned by the set $\mathcal{W}_{\mathcal{G}}$ all the graphs obtained by repeated application of production rules, starting with G_S. The set $\mathcal{W}_{\mathcal{G}}$ is different from the graph language $\mathcal{L}_{\mathcal{G}}$, as it also contains graphs whose vertices and edges are labelled by non-terminal symbols.

We then define the insertion operator $\lhd : \mathcal{V} \otimes \mathcal{V} \to \mathcal{V}$ as

$$G_1 \lhd G_2 = \sum_{v \in V(G_1)} P(v, v_2, G_2)(G_1) = \sum_{v \in V(G_1)} G_1 \lhd_v G_2. \tag{2.5}$$

Proposition 2.7 *Given a context-free Insertion Graph Grammar $\mathcal{G}$ as above, the insertion operator (2.5) defines a pre-Lie structure on the vector space $\mathcal{V}$.*

Proof We need to check that (2.2) is satisfied. We have

$$(G_1 \lhd G_2) \lhd G_3 = \sum_{v \in V(G_1)} \sum_{v' \in V(G_1 \lhd_v G_2)} (G_1 \lhd_v G_2) \lhd_{v'} G_3$$

where v and v' are glued, respectively, to the root source vertices v_2 and v_3 of G_2 and G_3. The choice of v' can be subdivided into the two cases where v' is a vertex of G_2 or v' is a vertex of G_1, including the case $v' = v$. Thus, we have

$$(G_1 \lhd G_2) \lhd G_3 = \sum_{v \in V(G_1)} \sum_{v' \in V(G_2)} (G_1 \lhd_v G_2) \lhd_{v'} G_3 + \sum_{v, v' \in V(G_1)} (G_1 \lhd_v G_2) \lhd_{v'} G_3.$$

Similarly, we have

$$G_1 \lhd (G_2 \lhd G_3) = \sum_{v \in V(G_1)} \sum_{v' \in V(G_2)} G_1 \lhd_v (G_2 \lhd_{v'} G_3),$$

where v is glued to is the base vertex v_2 of $G_2 \lhd_{v'} G_3$, which is the same as the base vertex of G_2. Thus, we obtain

$$(G_1 \lhd G_2) \lhd G_3 - G_1 \lhd (G_2 \lhd G_3) = \sum_{v, v' \in V(G_1)} G_1 \cup_{v \equiv v_2} G_2 \cup_{v' \equiv v_3} G_3$$

and similarly

$$(G_1 \lhd G_3) \lhd G_2 - G_1 \lhd (G_3 \lhd G_2) = \sum_{v, v' \in V(G_1)} G_1 \cup_{v' \equiv v_3} G_3 \cup_{v \equiv v_2} G_2,$$

which proves (2.2). □

We then obtain the associated Lie algebra.

Corollary 2.8 *Let $\mathcal{G}$ be a context-free Insertion Graph Grammar of rooted directed acyclic graphs, with start graph $G_S = \{v\}$ and production rules $P(v, v_2, G_2)$, with v_2 the source of G_2. Then there is an associated Lie algebra* $\mathrm{Lie}_{\mathcal{G}}$ *given by the vector space $\mathcal{V}$ spanned by the graphs of $\mathcal{W}_{\mathcal{G}}$ with the Lie bracket $[G_1, G_2] = G_1 \lhd G_2 - G_2 \lhd G_1$.*

Remark 2.9 A variant of the above construction that makes it (very mildly) context sensitive is obtained by requiring that the marked source vertex of the graph G_2 in a production rule $P(v, v_2, G_2)$ is glued to a *sink* vertex of the graph G_1, to which the rule is applied. The argument is exactly as before, and one obtains a pre-Lie insertion operator and a Lie algebra. We will generalize this context-sensitive version to more general gluing data in Propositions 2.10 and 2.11 below.

2.6 Some Lie Algebras of Context-Sensitive Grammars of Directed Graphs

We now consider a variant of the case of Proposition 2.7 where we consider an example of context-sensitive graph grammars. We still assume, as above, that $\mathcal{G}$ is an Insertion Graph Grammar $\mathcal{G}$ as in Definition 2.1, with start graph G_S a single vertex, and where all the graphs $G \in \mathcal{W}_{\mathcal{G}}$ are directed. We no longer require that they are acyclic, hence graphs will generally have oriented loops. An oriented loop γ in a graph G is an attractor if all the edges in $E_G(\gamma)$ (defined as in (2.1)) are incoming, that is, $\partial(e) \cap V(\gamma) = t(e)$. It is a repeller if all edges in $E_G(\gamma)$ are outgoing, $\partial(e) \cap V(\gamma) = s(e)$. In general, there will be also oriented loops that are neither attractors not repellers. We modify the previous context-free construction by considering, in addition to the production rules that glue a vertex of one graph to a source vertex of another, also context-sensitive production rules that glue an attractor loop of the first graph to a repeller loop of the second,

$$G_1 \lhd_\gamma G_2 := P(\gamma, \gamma_2, G_2)(G_1) = G_1 \cup_{\gamma \equiv \gamma_2} G_2, \tag{2.6}$$

where the two graphs are glued by identifying the two oriented loops γ and γ_2 (which necessarily have to have the same number of edges). The insertion operator is then defined as

$$G_1 \lhd G_2 = \sum_{\substack{\gamma \subset G_1 \\ \gamma \text{ attractor loop} \\ \gamma \simeq \gamma_2}} G_1 \lhd_\gamma G_2 = \sum_{\substack{\gamma \subset G_1 \\ \gamma \text{ attractor loop} \\ \gamma \simeq \gamma_2}} P(\gamma, \gamma_2, G_2)(G_1). \tag{2.7}$$

Proposition 2.10 *Given a context-sensitive Insertion Graph Grammar $\mathcal{G}$ as above, the insertion operator (2.5) defines a pre-Lie structure on the vector space $\mathcal{V}$ spanned by the graphs in $\mathcal{W}_{\mathcal{G}}$.*

Proof The composition $G_1 \lhd (G_2 \lhd G_3)$ is given by

$$G_1 \lhd (G_2 \lhd G_3) = \sum_{\gamma \subset G_1} \sum_{\gamma' \subset G_2} G_1 \lhd_\gamma (G_2 \lhd_{\gamma'} G_3)$$

while the composition $(G_1 \lhd G_2) \lhd G_3$ is

$$(G_1 \lhd G_2) \lhd G_3 = \sum_{\gamma \subset \Gamma_1} \sum_{\gamma' \subset G_1 \lhd_\gamma G_2} (G_1 \lhd_\gamma G_2) \lhd_{\gamma'} G_3.$$

In the last sum, the choice of $\gamma' \subset G_1 \lhd_\gamma G_2$ can be broken down into the case where $\gamma' \subset G_1$, the case where $\gamma' \subset G_2 \smallsetminus \gamma$, and the case where it intersects both, $\gamma' \cap G_1 \neq \emptyset$ and $\gamma' \cap G_2 \smallsetminus \gamma \neq \emptyset$. In fact, because of our assumptions on the production rules, only the first two possibilities can occur, and the first one can occur only with $\gamma' \cap \gamma = \emptyset$. To see this, suppose γ' intersects both sets. Then it must intersect γ, since γ' is connected and γ is the frontier between G_1 and G_2. Under our assumptions, γ' is an attractor loop for $G_1 \lhd_\gamma G_2$, hence all edges in $E_{G_1 \lhd_\gamma G_2}(\gamma')$ must be incoming to γ'. On the other hand, γ is an attractor loop for G_1 and a repeller loop for G_2, so inside $G_1 \lhd_\gamma G_2$, there are vertices of γ that have both incoming and outgoing edges in $E_{G_1 \lhd_\gamma G_2}(\gamma)$. Consider a vertex v in the intersection $\gamma' \cap \gamma$. Either γ' and γ have an adjacent edge in common, or they cross each other transversely at v. If they are transverse, then the incoming and outgoing edges of γ at v show that γ' cannot be an attractor loop for $G_1 \lhd_\gamma G_2$. If γ and γ' have at least one edge adjacent to v in common, then that edge is either incoming or outgoing at v. If it is incoming, then the next edge of γ is outgoing and that suffices to show γ' is not an attractive loop. If it is outgoing, then one can argue the same way with the next vertex. Thus, we can rewrite the sum above as

$$(G_1 \lhd G_2) \lhd G_3 = \sum_{\gamma \subset \Gamma_1} \sum_{\gamma' \subset G_1 \smallsetminus \gamma} (G_1 \lhd_\gamma G_2) \lhd_{\gamma'} G_3 + \sum_{\gamma \subset \Gamma_1} \sum_{\gamma' \subset G_2 \smallsetminus \gamma} (G_1 \lhd_\gamma G_2) \lhd_{\gamma'} G_3.$$

Notice that, in the sum describing $G_1 \lhd (G_2 \lhd G_3)$ we also have $\gamma' \cap \gamma = \emptyset$ because $\gamma' \subset G_2 \lhd_{\gamma'} G_3$ is not a repelling loop so it cannot intersect the repelling loop γ_2 that is glued to γ. We then obtain

$$(G_1 \triangleleft G_2) \triangleleft G_3 - G_1 \triangleleft (G_2 \triangleleft G_3) = \sum_{\substack{\gamma,\gamma' \subset G_1 \\ \gamma,\gamma' \text{ attractor loops}}} G_1 \cup_{\gamma \equiv \gamma_2} G_2 \cup_{\gamma' \equiv \gamma_3} G_3$$

and similarly

$$(G_1 \triangleleft G_3) \triangleleft G_2 - G_1 \triangleleft (G_3 \triangleleft G_2) = \sum_{\substack{\gamma,\gamma' \subset G_1 \\ \gamma,\gamma' \text{ attractor loops}}} G_1 \cup_{\gamma' \equiv \gamma_3} G_3 \cup_{\gamma \equiv \gamma_2} G_2,$$

which proves (2.2). □

We obtain an associated Lie algebra $\mathrm{Lie}_{\mathcal{G}}$, as in Corollary 2.8. This construction can be further generalized to other context-sensitive grammars, in the following way. Assume again that $\mathcal{G}$ is an Insertion Graph Grammar $\mathcal{G}$ as in Definition 2.1, with start graph G_S a single vertex, and where all the graphs are oriented. In addition to the production rules that glue a vertex of one graph to a source vertex of another, as in Proposition 2.7, we also allow for context-sensitive production rules of the form $P(G_L, H, G_R)$ where H is a connected, oriented graph with no sources or sinks. Moreover, we require that all the edges in $E_{G_L}(H)$ are incoming to H and all the edges in $E_{G_R}(H)$ are outgoing from H. We define the insertion operator by setting

$$G_1 \triangleleft_{G_L,H} G_2 := P(G_L, H, G_2)(G_1) = G_1 \cup_H G_2 \tag{2.8}$$

to be the gluing of G_1 and G_2 along H, whenever G_1 contains a pair of subgraphs isomorphic to $H \subset G_L$, with the orientation requirements as specified above. We then have

$$G_1 \triangleleft G_2 := \sum_{G_L \subset G_1} P(G_L, H, G_2)(G_1), \tag{2.9}$$

where the sum is over all the production rules and over all the possible ways of identifying G_L with a subgraph of G_1. The result is zero if G_1 does not contain any subgraph isomorphic to G_L. We then have the following straightforward generalization of Proposition 2.10.

Proposition 2.11 *Given a context-sensitive Insertion Graph Grammar $\mathcal{G}$ as above, the insertion operator* (2.9) *defines a pre-Lie structure on the vector space $\mathcal{V}$ spanned by the graphs in $\mathcal{W}_{\mathcal{G}}$.*

Proof The argument follows along the same lines as the previous cases. Observe that, since H is a directed graphs with neither sinks nor sources, at every vertex $v \in V(H)$ there are at least one incoming and one outgoing edge in $E(H)$. We then argue exactly as in Proposition 2.10. In the composition

$$(G_1 \triangleleft G_2) \triangleleft G_3 = \sum_{H \subset G_L \subset \Gamma_1} \sum_{H' \subset G'_L \subset G_1 \triangleleft_{G_L,H} G_2} (G_1 \triangleleft_{G_L,H} G_2) \triangleleft_{G'_L,H'} G_3,$$

if the subgraph $H' \subset G_1 \triangleleft_{G_L,H} G_2$ has nontrivial intersection with both G_1 and $G_2 \smallsetminus H$, then it must intersect H. Then the conditions on the orientations imply that H' cannot have only incoming edges in $E_{G_1 \triangleleft_{G_L,H} G_2}(H')$, which contradicts the orientation requirements for H'. So the only compositions that give non-trivial terms are the ones where H' is fully contained in either G_1 (in fact $G_1 \smallsetminus H$) or in $G_2 \smallsetminus H$. We then write the composition above by separating out the sums for these two cases, and the rest of the argument follows exactly as in the previous proposition. □

2.7 Lie Algebras and Gluing Along Half-Edges

We now consider Insertion Graph Grammars $\mathcal{G}$ as in Definition 2.3 and we consider the case where the left-hand-side of the production rules are corollas $C(v)$, though we do not require that they are all of the same valence, so that we include grammars that are (mildly) context-sensitive. Graphs are not necessarily oriented, but we assume that they

have a base vertex. For a graph G and a vertex $v \in V(G)$ we write $E_{ext}(G, v) \subset E_{ext}(G)$ for the subset of external edges of G that are attached to the vertex v. The condition on external edges in Definition 2.3 corresponds in this case to the requirement that the number of external edges $E_{ext}(G_2, v_2)$ attached to the base vertex v_2 is at least equal to the valence of $C(v)$. These external edges of G_2 are identified by the production rule with the half-edges of the corolla $C(v) \subset G_1$, where G_1 is the graph the production is applied to. The argument is then exactly as in Proposition 2.7. The base vertex of a production $P(C(v), C(v_2), G_2)(G_1)$ is the base vertex v_1 of G_1.

Proposition 2.12 *Let $\mathcal{G}$ be an Insertion Graph Grammars $\mathcal{G}$ as in Definition 2.3, where all the production rules are of the form $P(C(v), C(v_2), G_2)$ with $\mathrm{val}(C(v)) = \mathrm{val}(C(v_2))$ and $E_{ext}(C(v)) \subset E_{ext}(G_2, v_2)$. Assume the start graph G_S of $\mathcal{G}$ is also a corolla $C(v)$. The insertion operator*

$$G_1 \triangleleft G_2 = \sum_{\substack{v \in V(G_1) \\ \mathrm{val}(v)=\mathrm{val}(C(v_2))}} P(C(v), C(v_2), G_2)(G_1) = \sum_{\substack{v \in V(G_1) \\ \mathrm{val}(v)=\mathrm{val}(C(v_2))}} G_1 \triangleleft_{C(v)} G_2 \tag{2.10}$$

defines a pre-Lie structure on the vector space $\mathcal{V}$ generated by the graphs in $\mathcal{W}_{\mathcal{G}}$.

Proof We have

$$G_1 \triangleleft (G_2 \triangleleft G_3) = \sum_{\substack{v \subset V(G_1) \\ \mathrm{val}(v) \leq \# E_{ext}(G_2 \triangleleft_{C(v')} G_3, v_2)}} \sum_{\substack{v' \subset V(G_2) \\ \mathrm{val}(v') \leq \# E_{ext}(G_3, v_3)}} G_1 \triangleleft_{C(v)} (G_2 \triangleleft_{C(v')} G_3)$$

$$(G_1 \triangleleft G_2) \triangleleft G_3 = \sum_{\substack{v \subset V(G_1) \\ \mathrm{val}(v) \leq \# E_{ext}(G_2, v_2)}} \sum_{\substack{v' \subset V(G_1 \triangleleft_\gamma G_2) \\ \mathrm{val}(v') \leq \# E_{ext}(G_3, v_3)}} (G_1 \triangleleft_{C(v)} G_2) \triangleleft_{C(v')} G_3.$$

In the first sum $E_{ext}(G_2 \triangleleft_{C(v')} G_3, v_2)$ is $E_{ext}(G_2, v_2)$ since the base vertex v_2 of $G_2 \triangleleft_{C(v')} G_3$ is the base vertex of G_2. We then separate out the last sum of the second expression into the cases where $v' \in V(G_1)$ or $v' \in V(G_2)$. We obtain, as in the previous cases,

$$(G_1 \triangleleft G_2) \triangleleft G_3 - G_1 \triangleleft (G_2 \triangleleft G_3) = \sum_{v, v' \in V(G_1)} G_1 \cup_{C(v)} G_2 \cup_{C(v')} G_3 = (G_1 \triangleleft G_3) \triangleleft G_2 - G_1 \triangleleft (G_3 \triangleleft G_2).$$

□

Again we obtain an associated Lie algebra $\mathrm{Lie}_{\mathcal{G}}$. One can also similarly extend the case of gluing along oriented loops, by assigning orientations to the attached half-edges, with incoming/outgoing requirements as in Proposition 2.10, or in the case of gluing along more general graphs H with orientation requirements as in Proposition 2.11, reformulated in terms of half-edges. These analogs of Propositions 2.10 and 2.11 are completely straightforward and are proved by essentially the same argument, so we will not state them explicitly here. Within this setting, however, one cannot further extend the construction to more general context-sensitive cases, beyond what we have seen in Proposition 2.11, because the cases where the gluing data graph H' intersects both G_1 and $G_2 \smallsetminus H$ in $G_1 \triangleleft_H G_2$ creates terms that do not cancel in the difference $(G_1 \triangleleft G_2) \triangleleft G_3 - G_1 \triangleleft (G_2 \triangleleft G_3)$ and that are not symmetric with respect to exchanging G_2 and G_3. To see more precisely where the difficulty lies, we can write out the expression above, as before, in the form

$$(G_1 \triangleleft G_2) \triangleleft G_3 - G_1 \triangleleft (G_2 \triangleleft G_3) = \sum_{\substack{H \subset G_1 \\ H' \subset G_1 \triangleleft G_2}} (G_1 \cup_H G_2) \cup_{H'} G_3 - \sum_{\substack{H \subset G_1 \\ H' \subset G_2}} G_1 \cup_H (G_2 \cup_{H'} G_3).$$

We can separate out, in the first sum, the cases where H' is completely contained in G_1 or completely contained in G_2, and where it intersects both graphs. The latter case, in general, cannot be decomposed further, because it is not necessarily true that, if a certain graph H' is the gluing data of a production rule, subgraphs $G_1 \cap H'$ and $G_2 \cap H'$ would also occur in production rules. Thus, the term involving subgraphs H' intersecting both G_1 and G_2 does not cancel in the difference between $(G_1 \triangleleft G_2) \triangleleft G_3$ and $G_1 \triangleleft (G_2 \triangleleft G_3)$ and at the same time is not symmetric with respect to interchanging G_2 and G_3, hence one would not obtain a pre-Lie insertion operator.

We conclude this section by discussing a special example, which we will return to in our application to Feynman graphs. We denote by G_e the graph consisting of a single edge e, identified with the union of two half-edges $e = \{f, f'\}$ glued together by an involution $\mathcal{I}(f) = f'$.

Definition 2.13 Let $\mathcal{G}$ be an Insertion Graph Grammars $\mathcal{G}$ as in Definition 2.3, with start graph G_S and with production rules:

(1) $P(G_S, \{f, f'\} \subset \mathcal{F}_{G_S}, G_e)$, where the single edge graph G_e is glued onto a pair $\{f, f'\}$ of external half-edges of G_S,
(2) $P(G_S, \{f\} \subset \mathcal{F}_{G_S}, G_S \cup_{f'} G_e)$, where the copy of G_S in the right-hand-side of the production is glued to the one on the left-hand-side by matching the remaining external half-edge of G_e to the half-edge f to form a graph $G_S \cup_{e=(f,f')} G_S$ consisting of two copies of G_S glued together along an edge.

Proposition 2.14 *Let $\mathcal{G}$ be an Insertion Graph Grammar as in Definition 2.13 above. Then the insertion operator*

$$G_1 \triangleleft G_2 = \sum P(G_S, \mathcal{F}_{G_S}, G_2)(G_1) \tag{2.11}$$

defines a pre-Lie structure on the vector space $\mathcal{V}$ spanned by the graphs in $\mathcal{W}_{\mathcal{G}}$.

Proof It suffices to notice that, by the form of the production rules, in the composition $(G_1 \triangleleft G_2) \triangleleft G_3$ the gluing of G_3 to $G_1 \triangleleft G_2$ happens along some of the external half-edges of $G_1 \triangleleft G_2$. The previous gluing of G_2 to G_1, in turn, glues some external half-edges of G_2 to some external half-edges of G_1. Thus, the remaining half-edges of $G_1 \triangleleft G_2$ are either in $G_1 \smallsetminus H$ or in $G_2 \smallsetminus H$, where H is the set of half-edges along which the gluing of G_2 to G_1 happened (which are no longer external edges in $G_1 \triangleleft G_2$). This suffices then to get the pre-Lie condition, exactly as in the cases discussed previously. □

Applying (2.11) with P a production rule of the first type listed in Definition 2.13 corresponds to the operation of gluing together a pair of external edges of G_S to form a single edge. This operation can easily be modified to include the possibility of valence two vertices, as is customary in quantum field theory, by redefining the graph G_e in the production rule, so that it consists of a valence two vertex with two half-edges, which are matched to the half edges $\{f, f'\}$. Applying (2.11) with P a production rule of the second type listed in Definition 2.13, on the other hand, results in two copies of the same graph G_S glued along a pair of external edges. We return to discuss more explicitly these two operations in §3 below, where we will see that this provides a different way of constructing of Lie algebras of Feynman graphs. The Lie algebras obtained in this way are not equivalent to the insertion Lie algebra of [6,9]. More notably, we will show that the set of Feynman graphs of a given quantum field theory is a graph language in the sense of the theory of formal languages.

2.8 Lie Algebras of Insertion-Elimination Graph Grammars

We now consider the case of insertion-elimination graph grammars, as in Definitions 2.5 and 2.6.

Let $\mathcal{G}$ be an insertion-elimination graph grammar as in Definition 2.6. We assume the following hypotheses:

(1) In all the production rules $P(G_L, H, G_R)$ the graph G_L is connected and we have $H \subset \mathcal{F}_{G_L}$, a set of flags (half-edges), with $H = E_{ext}(G_R)$.
(2) All the graphs in $\mathcal{W}_{\mathcal{G}}$ are oriented and in all production rules the half-edges in H are incoming to both G_L and G_R.

Proposition 2.15 *Let $\mathcal{G}$ be an insertion-elimination graph grammar satisfying the two conditions above. Then the insertion operator*

$$G_1 \triangleleft G_2 = \sum_{G \subset G_1} P(G, H, G_2)(G_1) \tag{2.12}$$

is a pre-Lie operator.

Proof With the notation $G_1 \lhd_{G,H} G_2 = P(G, H, G_2)(G_1)$, we have

$$G_1 \lhd (G_2 \lhd G_3) = \sum_{G,G'} G_1 \lhd_{G,H} (G_2 \lhd_{G',H'} G_3),$$

with $G \subset G_1$ and $H \subset \mathcal{F}_{G_1}$, $H = E_{ext}(G_2 \lhd_{G',H'} G_3)$, and with $G' \subset G_2$, $H' \subset \mathcal{F}_{G_2}$ and $H' = E_{ext}(G_3)$. Since all the external edges of G_3 are glued to flags of G_2, in the identification along $H' = E_{ext}(G_3)$, we have $H = E_{ext}(G_2 \lhd_{G',H'} G_3) = E_{ext}(G_2)$. When composing in the opposite order, we have

$$(G_1 \lhd G_2) \lhd G_3 = \sum_{G,G'} (G_1 \lhd_{G,H} G_2) \lhd_{G',H'} G_3,$$

with $G \subset G_1$, $H \subset \mathcal{F}_{G_1}$, $H = E_{ext}(G_2)$, and with $G' \subset G_1 \lhd_{G,H} G_2$, $H' \subset \mathcal{F}_{G_1 \lhd_{G,H} G_2}$, $H' = E_{ext}(G_3)$. If the graph G' intersects nontrivially both $G_1 \smallsetminus (G \smallsetminus H)$ and G_2, then by connectedness $H' \cap H \neq \emptyset$, but the orientation conditions on the edges of H and of H' are incompatible, so G' must be contained in either $G_1 \smallsetminus G$ or in G_2. We then obtain

$$(G_1 \lhd G_2) \lhd G_3 - G_1 \lhd (G_2 \lhd G_3) = \sum_{\substack{G,G' \subset G_1 \\ G \cap G' = \emptyset}} (G_1 \smallsetminus ((G \smallsetminus H) \cup (G' \smallsetminus H'))) \cup_H G_2 \cup_{H'} G_3$$

which is symmetric in exchanging G_2 and G_3. □

We obtain an associated Lie algebra $\mathrm{Lie}_{\mathcal{G}}$.

2.8.1 Grading

The Connes–Kreimer Hopf algebra of Feynman graphs and the corresponding insertion Lie algebra are naturally graded, either by loop number (if one does not include valence two vertices) or by number of internal edges. The key property that makes such numbering a good grading is the additivity

$$g(\Gamma) = g(\gamma) + g(\Gamma/\gamma),$$

of the grading function, over decompositions into a subgraph $\gamma \subset \Gamma$ and the quotient graph Γ/γ. In terms of insertions, this corresponds to the additivity

$$g(G_1 \circ_v G_2) = g(G_1) + g(G_2)$$

over insertions of one graph into another at a vertex. These additivity relations can be seen as a special case of inclusion-exclusion relations

$$g(G_1 \circ_v G_2) = g(G_1) + g(G_2) - g(v),$$

where the function satisfies $g(v) = 0$, as is the case for loop number or number of internal edges. In the case of Lie algebras obtained from graph grammars, one can similarly consider a $\mathbb{Z}_+$-valued function on the set of graphs belonging to the graph language, satisfying the inclusion-exclusion property

$$g(G_1 \cup_H G_2) = g(G_1) + g(G_2) - g(H).$$

In the case of an Insertion Graph Grammar, this would assign

$$g(P(G_L, H, G_2)(G_1)) = g(G_1 \cup_H G_2) = g(G_1) + g(G_2) - g(H),$$

and for an Insertion-Elimination Graph Grammar, it would assign

$$g(P(G_L, H, G_2)(G_1)) = g(G_1 \smallsetminus (G_L \smallsetminus H)) \cup_H G_2) = g(G_1) + g(G_2) - g(G_L).$$

For such a function g to define a grading, it suffices that it vanishes on all the graphs H (respectively G_L) that appear in the (finitely many) production rules of the graph language. A possible way to approach the question of the existence of a grading is then to reformulate it as the question of the existence of such functions.

The enveloping algebra of a pre-Lie algebra can be constructed explicitly, using the construction of [20]. In the graded connected case, this also gives a dual commutative Hopf algebra. For example, the Connes–Kreimer Hopf algebra of Feynman graphs and the insertion Lie algebra are related to one another in this way. In the cases of pre-Lie algebras obtained from Graph Grammars, one can similarly construct the enveloping algebra, as in [20]. However, notice that in cases where a grading is obtained as described above, via an inclusion-exclusion function that vanishes on the graphs H (or G_L) of the production rules, the connected property would not be satisfied, since all these graphs would be in the degree zero component, which would then be of dimension greater than one.

A related question is whether other well-known pre-Lie algebras of graphs have a Graph Grammar interpretation, such as the pre-Lie algebra of rooted trees with grafting, or the pre-Lie algebras of [17]. We do not investigate this further in this paper, but we expect that interesting pre-Lie algebras related to rooted trees may arise from the class of Tree Adjoining Grammars, whose production rules include grafting operations, [1].

2.9 The Insertion Lie Algebra of Quantum Field Theory and Primitive Graphs

In the previous sections we have shown that one can associate to certain classes of graph grammars an insertion Lie algebra, that behaves very similarly to the insertion Lie algebra of Quantum Field Theory of [6,9]. It is then natural to ask whether the insertion Lie algebra of Quantum Field Theory is itself obtained from a Graph Grammar via the same procedure discussed above. This is *not* the case, because it would violate the property that graph grammars have a *finite* number of production rules. In fact, the Lie algebra of Feynman graphs is generated by the primitive elements of the Hopf algebra. We would like to obtain all 1PI Feynman graphs of the theory from a graph grammar that has a single start graph G_S and a finite number of production rules $P(G_L, H, G_R)$, in such a way that the Lie bracket $[G_1, G_2]$ of the Lie algebra of quantum field theory would agree with the Lie bracket defined by the graph grammar,

$$[G_1, G_2] = \sum_{G_L \subset G_1} P(G_L, H, G_2)(G_1) - \sum_{G_L \subset G_2} P(G_L, H, G_1)(G_2).$$

For this to be the case, we see that we would need a production rule for each insertion of a primitive graph G of the Hopf algebra of the theory into a vertex with valence equal to the number of external edges of G. Since there are infinitely many primitive graphs, this would violate the requirement that the graph grammar has only finitely many production rules. We will see in §3 that, despite this negative result, the Feynman graphs of a given quantum field theory are a graph language, obtained from a graph grammar with finitely many production rules. These graph grammars in turn define Lie algebras, by the procedure discussed in the previous sections, which are in general not equivalent to the insertion Lie algebra of [6,9].

3 Feynman Diagrams as Graph Languages

Motivated by the insertion Lie algebra of quantum field theory, [6,9], we have shown in the previous section that, under certain conditions on the production rules, one can associate Lie algebras to Graph Grammars. In this section, we return to the motivating example of Feynman graphs and we show that the Feynman graphs of certain quantum field theories are examples of graph languages. Our generative description of Feynman graphs in terms of graph grammars can be seen as "reading in reverse" the procedure described in [3,14], that generates all Feynman graphs starting with the vacuum bubbles (no external edges) and progressively cutting internal edges into pairs of external half-edges.

3.1 The ϕ^4 Graph Language

We first analyze the example of the ϕ^4-theory. This is the scalar quantum field theory with (Euclidean) Lagrangian density

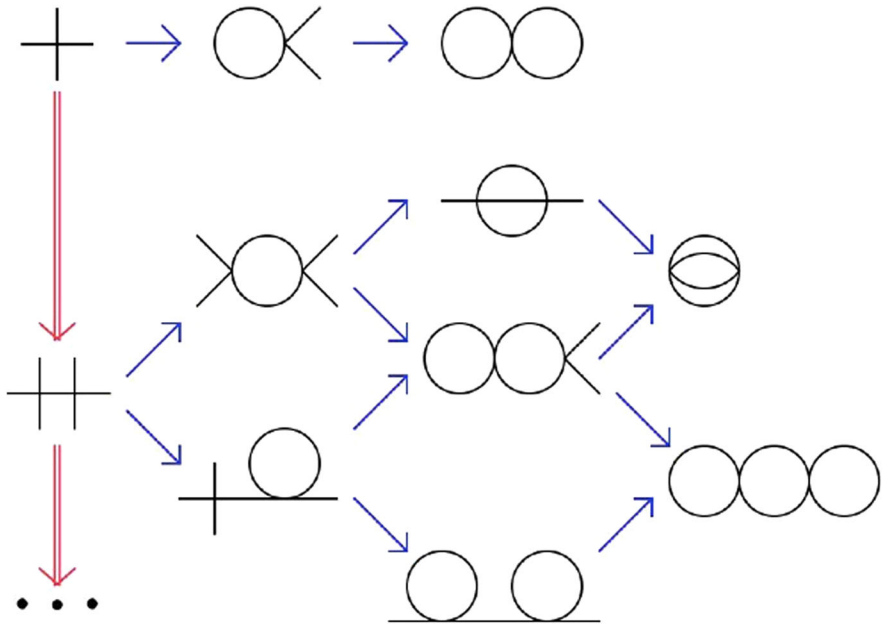


Fig. 1 Generative grammar for Feynman graphs: ϕ^4-theory

$$\mathcal{L}(\phi) = \frac{1}{2}(\partial\phi)^2 + \frac{1}{2}m^2\phi^2 + \frac{1}{4!}\lambda\phi^4. \tag{3.1}$$

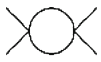
The Feynman graphs of this theory have all vertices of valence four. More precisely, we should also include valence two vertices that correspond to the mass and kinetic terms, but we will not mark them explicitly in the diagrams.

Proposition 3.1 *The Feynman graphs of the ϕ^4-theory are the elements of the graph language $\mathcal{L}_{\mathcal{G}}$ generated by a graph grammar $\mathcal{G}$ as in Definition 2.13 and Proposition 2.14, with start graph G_S given by a 4-valent corolla, and two production rules: one of the form $P(G_S, \{f, f'\} \subset \mathcal{F}_{G_S}, G_e)$, which glues together two external edges of G_S and one of the form $P(G_S, \{f\} \subset \mathcal{F}_{G_S}, G_S \cup_{f'} G_e)$, which glues together two copies of G_S along an edge. At each stage in the application of one of the production rules, the external edges of the resulting graph are marked either with a terminal or with a non-terminal symbol.*

Proof Whenever the external edges of a graph in $\mathcal{W}_{\mathcal{G}}$ are marked by non-terminal symbols one can continue to apply production rules to them, while if all the external edges are marked by terminals the resulting graph is in $\mathcal{L}_{\mathcal{G}}$. Thus, a graph is in $\mathcal{L}_{\mathcal{G}}$ if either it has no more external edges, in which case it is a vacuum bubble of the ϕ^4-theory, or if all the external edges are marked by terminals, in which case, it can be identified with the result of cutting a number of edges of a vacuum bubble into half edges. This produces all Feynman graphs of the ϕ^4 theory, [14]. □

The production procedure is illustrated in Fig. 1. The graph in the upper left-hand corner is G_S for this theory. Single arrows indicate the first type of production gluing external edges to make a loop. Double arrows indicate the second type of production joining base graphs along an edge to make a new graph with no loops. the graphs with no external edges are represented on the right, while the graphs with as many external edges as possible are on the left. Note that the latter type of graphs are trees. It is clear that for this theory, any allowed Feynman graph can be transformed into a tree such as the ones above. Consider the graphs on one internal vertex. They all can be constructed by gluing the edges of the base graph G_S, via repeated application of the production rule represented by the single arrows. A similar statement can be made of the graphs on two internal vertices, except that, after cutting all edges into pairs of half edges, one is now left with a disjoint union of two copies of the base graph G_S. In general, any (connected) ϕ^4 graph with k valence four vertices can be constructed by gluing together pairs of half-edges, starting from k copies of G_S, hence by repeated applications of the two types of production rules.

Example 3.2 To see that the Lie algebra constructed in this way is not the same as the usual insertion Lie algebra of quantum field theory, consider the ϕ^4 graphs

$G_1 =$  $G_2 =$  $G_3 =$ 

In the usual insertion Lie algebra of quantum field theory, the graph G_3 appears in the bracket $[G_1, G_2]$ as the insertion of G_1 at the vertex of G_2, while in the Lie algebra associated to the Graph Grammar of Proposition 1, the graph G_3 does not appear in the bracket $[G_1, G_2]$ since, as one can see from Fig. 1, the graph G_3 is not obtained from G_1 and G_2 by application of a production rule of the grammar.

3.2 Graph Grammar for ϕ^k-Theories

The case discussed above of the ϕ^4-theory can easily be generalized to the case of (Euclidean) Lagrangian densities

$$\mathcal{L}(\phi) = \frac{1}{2}(\partial\phi)^2 + \frac{1}{2}m^2\,\phi^2 + P(\phi) \tag{3.2}$$

where the interaction term is a single monomial $P(\phi) = \frac{1}{k!}\lambda\,\phi^k$.

Proposition 3.3 *The Feynman graphs of the ϕ^k-theory are the elements of the graph language $\mathcal{L}_{\mathcal{G}}$ generated by a graph grammar $\mathcal{G}$ as in Proposition 2.14, with start graph G_S given by a k-valent corolla, and two production rules: one of the form $P(G_S, \{f, f'\} \subset \mathcal{F}_{G_S}, G_e)$, which glues together two external edges of G_S and one of the form $P(G_S, \{f\} \subset \mathcal{F}_{G_S}, G_S \cup_{f'} G_e)$, which glues together two copies of G_S along an edge. At each stage in the application of one of the production rules, the external edges of the resulting graph are marked either with a terminal or with a non-terminal symbol.*

Proof Let G be a connected graph of the ϕ^k theory. Thus, all vertices $v \in V(G)$ have valence $\mathrm{val}(v) = k$. We neglect for the moment the possible presence of valence 2 vertices associated to the kinetic and mass terms in $\mathcal{L}(\phi)$. Consider all possible ways of cutting internal edges, so that they are replaced by a pair of external half-edges, that leave the graph connected. The number of possible such cuts is the degree of edge-connectedness of the graph. Stop when no further such cuts remain. If we denote the resulting graph by G', then it is clear that G is obtained from G' by repeatedly applying the first production rule. Every connected graph has a decomposition into a tree with insertions at the vertices of 1PI graphs (one particle irreducible, also known as 2-edge-connected) that have a number of external edges equal to the valence of the tree vertex. By repeatedly cutting non-disconnecting edges, and using this decomposition, it is clear that the resulting graph G' is a tree. Since all the vertices of G' have valence $\mathrm{val}(v) = k$, the tree can be constructed by repeated application of the second production rule. □

Remark 3.4 One can consider also the presence of valence two vertices, with each gluing of a copy of G_e in the production rules involving a valence two vertex inserted in the middle of an edge connecting two valence k vertices, and the argument remains essentially the same. The difference between valence two vertices coming from the kinetic and the mass terms can be taken care of by using two different terminal symbols labeling the vertices.

3.3 Graph Grammars for Arbitrary Scalar Field Theories

We then consider the case of a scalar field theory with Lagrangian density (3.2) where the interaction term is a polynomial

$$P(\phi) = \sum_{k\geq 3} \frac{\lambda_k}{k!}\phi^k. \tag{3.3}$$

Proposition 3.5 *The Feynman graphs of a scalar field theory with interaction polynomian $P(\phi)$ as in (3.3) of degree N are the elements of the graph language $\mathcal{L}_{\mathcal{G}}$ generated by a graph grammar $\mathcal{G}$ as in Proposition 2.14, with start graph G_S given by a k-valent corolla, where k is the smallest term in (3.3) with $\lambda_k \neq 0$ and three production rules:*

(1) The first kind of production rules is of the form $P(G_S, \{f, f'\} \subset \mathcal{F}_{G_S}, G_e)$, which glues together two external edges of G_S.

(2) The second kind of production rule $P(G_S, G_e, G_{S,f})$ glues a copy of G_e to the start graph G_S by identifying one half edge of G_e with one of the half-edges of G_S and leaving the other half edge f as a new external edge, thus creating a corolla of valence $k+1$. If $\lambda_{k+1} \neq 0$ the vertex of the resulting graph $G_{S,f}$ can be labeled by either a terminal or a nonterminal symbol, if $\lambda_{k+1} = 0$ it is labeled by a nonterminal symbol. Production rules $P(G_{S,f_1,\dots,f_r}, G_e, G_{S,1,\dots,f_r,f'})$ can be further applied to previously produced corollas $G_{S,f_1,\dots,f_r}$ with vertex labeled by nonterminals, until $\mathrm{val}(G_{S,f_1,\dots,f_r}) = N-1$: in this case the vertex in the resulting $G_{S,1,\dots,f_r,f'}$ can only be labelled by a terminal.

(3) The third kind of production rules $P(G_{S,f_1,\dots,f_r}, \{f_i\} \subset \mathcal{F}_{G_S}, G_{S,f_1,\dots,f_r} \cup_{f_i=f'_j} G_{S,f'_1,\dots,f'_s})$, which glues together along an edge two corollas $G_{S,f_1,\dots,f_r}$ and $G_{S,f'_1,\dots,f'_s}$ produced by the previous type of production rules.

At each stage in the application of one of the production rules, the external edges of the resulting graph are marked either with a terminal or with a non-terminal symbol.

Proof The argument is similar to the previous case: one starts from an arbitrary connected Feynman graph G of the theory and performs the maximal number of cuts of internal edges into pairs of external half-edges that leaves the graph connected. The only difference in the argument is that the resulting graph G' is now a tree with vertices of valences ranging among the values $3 \leq k \leq N$ for which $\lambda_k \neq 0$ in (3.3). These are then obtained by repeated application of the production rules of the second and third type that produce corollas of the right valences and glue them together along edges to form the tree G'. □

Figure 2 illustrates the generative grammar of Proposition 3.5 for the scalar field theory with

$$\mathcal{L}(\phi) = \frac{1}{2}(\partial\phi)^2 + \frac{1}{2}m^2\phi^2 + \frac{1}{6}\lambda_3\phi^3 + \frac{1}{24}\lambda_4\phi^4.$$

3.4 Graph Grammar for the $\phi^2 A$-Theory

This theory has two different propagators for the fields ϕ and A, which one represents by drawing straight edges for the ϕ-propagator and wavy edges for the A-propagator. The cubic interaction terms implies that the Feynman graphs have vertices of valence 3 with two straight and one wavy external half-edges. The Feynman graphs of this theory were analyzed in [14], with ϕ representing fermions and A the photon. Note that in this theory edges representing photons are always internal, unlike what happens in quantum electrodynamics, where photons can be external, [3].

Proposition 3.6 *The Feynman graphs of the $\phi^2 A$-theory are the elements of the graph language $\mathcal{L}_{\mathcal{G}}$ generated by a graph grammar $\mathcal{G}$ as in Proposition 2.14. The start graph G_S has two trivalent vertices, one internal wavy edge connecting them, and four external straight half-edges. There are two kinds of production rules: one of the form $P(G_S, \{f, f'\} \subset \mathcal{F}_{G_S}, G_e)$, which glues together two external edges of G_S and one of the form $P(G_S, \{f\} \subset \mathcal{F}_{G_S}, G_S \cup_{f'} G_e)$, which glues together two copies of G_S along an external edge. At each stage in the application of one of the production rules, the external edges of the resulting graph are marked either with a terminal or with a non-terminal symbol.*

Proof The graph grammar $\mathcal{G}$ is illustrated in Fig. 3. Notice that, if photon edges were allowed to be external, then the argument would be the same as in the ϕ^3-theory, except that the labeling of the edges as bosonic or fermionic

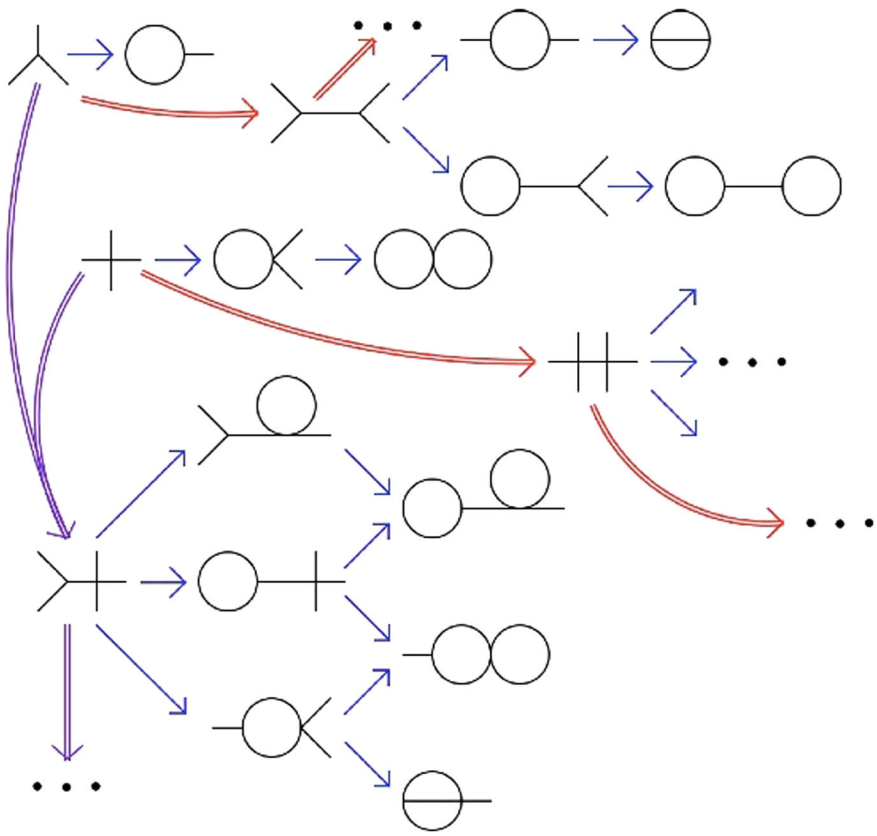


Fig. 2 Generative grammar for $P(\phi) = \frac{1}{6}\lambda_3\,\phi^3 + \frac{1}{24}\lambda_4\,\phi^4$

must be taken into account when inserting the base graph in the tree. The fact that we require bosonic edges to be internal means that these edges cannot be cut in the process that leads from G to G'. SInce all vertices in G have valence 3 with two fermion and one boson line, after all the internal fermion lines are cut, one still obtains a tree G', which we now view as being formed out of repeated application of the second production rule applied to the start graph. □

The properties of external and internal edges of the $\phi^2 A$-theory discussed in [14] are reflected here in the fact that, in the production rules, it is only possible to join base graphs along fermion edges. The fact that photon edges are only internal is taken into account by the choice of the start graph having two vertices instead of one, with one internal bosonic edge.

3.5 A General Procedure

We can summarize all the cases discussed above, for the different theories, in a common general procedure, as follows.

- Fix $n \in \mathbb{Z}^+$. This is the number of distinct conditions.
- Fix $t \in \mathbb{Z}^+$. This is the number of edge types. Let $v \in \{0, 1\}^t$ where $v(i)$ is 0 if type i edges cannot be external and 1 otherwise.
- For each $k \in \{1, \ldots, n\}$:
  - Let $C_k \in M_{t,2}(\mathbb{Z})$ where $C_k(i, 1) = i$ for all $i \in \{1, \ldots, t\}$ (the edge type) and with $C_k(i, 2)$ the number of edges of type i allowed by this condition at each vertex.

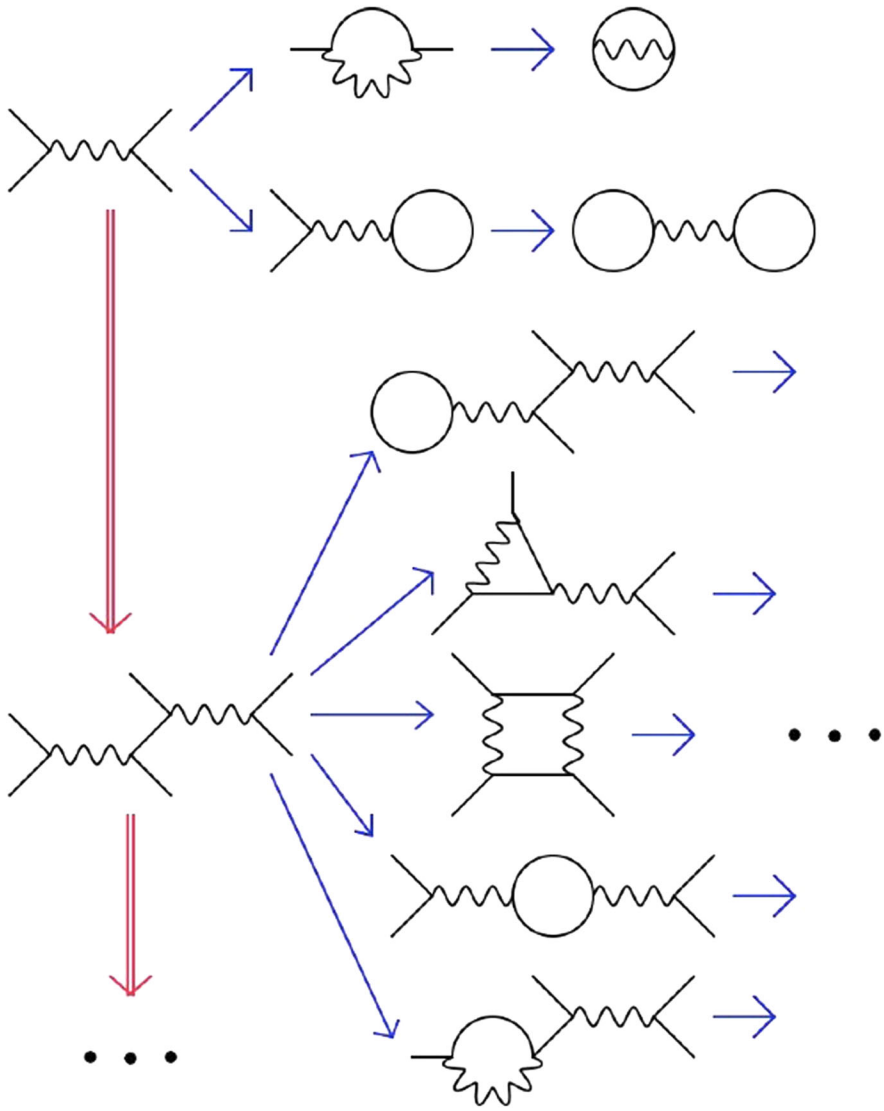


Fig. 3 Generative grammar for the $\phi^2 A$-theory

  - Let G_k be the star graph with edges determined by C_k.

- Let G be a connected graph that satisfies the conditions of v and each of the C_k.
- There is a sequence of production rules that glue edges of finitely many copies of the graphs G_k to make G.
- Any G that satisfies these conditions can be constructed from these initial graphs using the production rules.
- In order to have a single start graph one needs to add further production rules that derive higher valence star graphs G_k from lower valence ones, marking the vertex with a terminal label when the process should stop.

Example 3.7 In a $\phi^2 A$ theory, $n = 1$, $t = 2$, $v = (1, 0)$ and $C_1 = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}$.

Example 3.8 In a theory where $P(\phi) = \frac{\lambda_3}{3!}\phi^3 + \frac{\lambda_4}{4!}\phi^4$, $n = 2$, $t = 1$, $v = (1)$, $C_1 = (1\ 3)$ and $C_2 = (1\ 4)$.

Acknowledgments The first author is supported by NSF Grants DMS-1007207, DMS-1201512, PHY-1205440. The second author was supported by a Summer Undergraduate Research Fellowship at Caltech.

References

1. Abeille, A., Rambow, O.: Tree adjoining grammars, center for the study of language and information (2001)
2. Agrachev, A., Gamkrelidze, R.: Chronological algebras and nonstationary vector fields. J. Sov. Math. **17**(1), 1650–1675 (1981)
3. Bachmann, M., Kleinert, H., Pelster, A.: Recursive graphical construction for Feynman diagrams of quantum electrodynamics. Phys.Rev. D **61**, 085017 (2000)
4. Burde, D.: Left-symmetric algebras, or pre-Lie algebras in geometry and physics. Cent. Eur. J. Math. **4**(3), 323–357 (2006)
5. Connes, A., Kreimer, D.: Renormalization in quantum field theory and the Riemann-Hilbert problem. I. The Hopf algebra structure of graphs and the main theorem. Commun. Math. Phys. **210**(1), 249–273 (2000)
6. Connes, A., Kreimer, D.: Insertion and elimination: the doubly infinite Lie algebra of Feynman graphs. Ann. Henri Poincaré **3**(3), 411–433 (2002)
7. Connes, A., Marcolli, M.: Noncommutative Geometry, Quantum Fields and Motives. Colloquium Publications, Vol. 55. American Mathematical Society, Providence, RI (2008)
8. Delaney, C., Marcolli, M.: Dyson–Schwinger equations in the theory of computation. arXiv:1302.5040, to appear in "Periods and Motives", Clay Math Institute and AMS
9. Ebrahimi-Fard, K., Gracia-Bondia, J.M., Patras, F.: A Lie theoretic approach to renormalization. Commun. Math. Phys **276**(2), 519–549 (2007)
10. Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of Algebraic Graph Transformation. Springer, New York (2010)
11. Ehrig, H., Kreowski, H.J., Rozenberg, G.: Graph-grammars and their application to computer science. Lecture Notes in Computer Science, Vol. 532, Springer, New York (1990)
12. Foissy, L.: Lie algebras associated to systems of Dyson-Schwinger equations. Adv. Math. **226**(6), 4702–4730 (2011)
13. Itzykson, C., Zuber, J.B.: Quantum Field Theory. Dover Publications, New York (2012)
14. Kleinert, H., Pelster, A., Kastening, B., Bachmann, M.: Recursive graphical construction of Feynman diagrams and their multiplicities in ϕ^4 and in $\phi^2 A$ theory. Phys. Rev. E **62**, 1537–1559 (2000)
15. Kreimer, D.: On the Hopf algebra structure of perturbative quantum field theories. Adv. Theor. Math. Phys. **2**(2), 303–334 (1998)
16. Manchon, D.: A short survey on pre-Lie algebras, In: Noncommutative geometry and physics: renormalisation, motives, index theory, pp. 89–102, ESI Lect. Math. Phys., Eur. Math. Soc. (2011)
17. Manchon, D., Saidi, A.: Lois pré-Lie en interaction. Commun. Algebra **39**(10), 3662–3680 (2011)
18. Manin, Y.I.: Infinities in quantum field theory and in classical computing: renormalization program. In: Programs, proofs, processes, pp. 307–316. Lecture Notes in Computer Science, Vol. 6158, Springer, New York (2010)
19. Nagl, M.: Graph-Grammatiken: Theorie, Implementirung, Anwendung. Vieweg, Braunschweig (1979)
20. Oudom, J.M., Guin, D.: On the Lie enveloping algebra of a pre-Lie algebra. J. K Theory **2**(1), 147–167 (2008)
21. Rozenberg, G.: Handbook of Graph Grammars and Computing by Graph Transformation. Volume 1: Foundations. World Scientific, Singapore (1997)
22. Rozenberg, G.: An introduction to the NLC way of rewriting graphs, In: Graph-grammars and their application to computer science, pp. 55–66. Lecture Notes in Computer Science, Vol. 532, Springer, New York (1990)