Graph Grammars

Matilde Marcolli
CS101: Mathematical and Computational Linguistics

Winter 2015

Formal languages adapted to parallelism in computation

- instead of linear languages: strings in an alphabet obtained by production rules of a grammar
- grammars that produce a language consisting of a family of graphs
- production rules that substitute parts of a graph with other parts (gluing)
- an initial graph as starting point
- edge and vertex labels by terminal and non-terminal symbols

Graphs

Two main ways of thinking about graphs:

First description:

- $V(G)$ = set of vertices; $E(G)$ = set of edges; $\partial : E(G) \to V(G) \times V(G)$
- if G is oriented (directed) then source and target $s, t : E(G) \to V(G)$
- Σ_V, Σ_E sets of vertex and edge labels; $L_{V,G} : V(G) \to \Sigma_V$, $L_{E,G} : E(G) \to \Sigma_E$ assignment of labels

Second description:

- $C(G)$ = set of corollas with assigned valences
(a vertex with n half-edges)

- $\mathcal{F}(G)$ = set of all half-edges

- involution: $\mathcal{I} : \mathcal{F}(G) \to \mathcal{F}(G)$

- edges: pairs (f, f') with $f \neq f'$ in $\mathcal{F}(G)$ with $\mathcal{I}(f) = f'$
(an edge is a gluing of two half edges)

- external edges: $f \in \mathcal{F}(G)$ fixed by the involution $\mathcal{I}$
(half-edges not matched to anything else)

- assignment of labels $L_{\mathcal{F},G} : \mathcal{F}(G) \to \Sigma_{\mathcal{F}}$ and $L_{V,G} : C(G) \to \Sigma_V$

$$L_{\mathcal{F},G} \circ \mathcal{I} = L_{\mathcal{F},G}$$

(the involution must match labels)

Graph Grammar

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

- edge labels: $\Sigma_E = N_E \cup T_E$ non-terminal and terminal
- vertex labels: $\Sigma_V = N_V \cup T_V$ non-terminal and terminal
- $G_S =$ start graph
- $P =$ production rules: a finite set

Production rules of a Graph Grammar

$$P = (G_L, G_R, H)$$

- G_L = labelled graph (l.h.s. of production)
- G_R = labelled graph (r.h.s. of production)
- H = labelled graph with label preserving isomorphisms

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

(isomorphic subgraphs in G_L and G_R)

Meaning: the production rule 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

Context-free Graph Grammars

- when all production rules $P = (G_L, G_R, H)$ have G_L (hence H) a single vertex

- Chomsky hierarchy for Graph Grammar (different from the one for linear languages) was identified in
  - M. Nagl, *Graph-Grammatiken: Theorie, Implementirung, Anwendung*, Vieweg, 1979

References on Graph Grammars

- H. Ehrig, K. Ehrig, U. Prange, G. Taentzer, *Fundamentals of algebraic graph transformation*. New York: Springer, 2010.
- H. Ehrig, H.J. Kreowski, G. Rozenberg, *Graph-grammars and their application to computer science*, Lecture Notes in Computer Science, Vol. 532, Springer, 1990.
- G. Rozenberg, *Handbook of Graph Grammars and Computing by Graph Transformation. Volume 1: Foundations*, World Scientific, 1997.

Graph Grammars and Quantum Field Theory

(from a project with Alex Port, SURF 2014)

- perturbative (massless, scalar) field theory: classical action

$$S(\phi) = \int \mathcal{L}(\phi) d^D x = S_0(\phi) + S_{int}(\phi)$$

in D dimensions

- Lagrangian density

$$\mathcal{L}(\phi) = \frac{1}{2}(\partial\phi)^2 - \frac{m^2}{2}\phi^2 - \mathcal{L}_{int}(\phi)$$

- Quantum effects: perturbative expansion in Feynman diagrams

$$S_{eff}(\phi) = S_0(\phi) + \sum_G \frac{G(\phi)}{\#\mathrm{Aut}(G)} \quad \text{(1PI graphs)}$$

contribution of each Feynman graph is a finite dimensional integral in momentum variables flowing through the graph

Renormalization problem in QFT

• most of the integrals $G(\phi)$ in the expansion are divergent

• need a regularization procedure (pole subtraction, cutoff, ...)

• and renormalization is implementation consistently over subgraphs (nested subdivergences)

• Connes-Kreimer ($\sim$ 2000): the renormalization procedure is described algebraically by a Hopf algebra of Feynman graphs

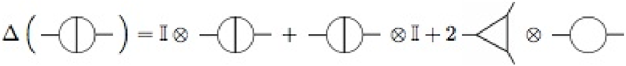
Hopf algebra of Feynman graphs

• commutative algebra generated by all the 1PI graphs G of the QFT (polynomial algebra in the G)

• comultiplication $\Delta : \mathcal{H} \to \mathcal{H} \otimes \mathcal{H}$ (coassociative, non-cocommutative)

$$\Delta(G) = G \otimes 1 + 1 \otimes G + \sum_{\gamma \subset G} \gamma \otimes G/\gamma$$

• Example:



• antipode (related algebra and coalgebra structure) constructed inductively on number of edges (or loops)

Hopf algebra and Lie algebra

• $\mathcal{H} = \oplus_{n\geq 0}\mathcal{H}_n$ with $\mathcal{H}_0 = \mathbb{C}$ connected commutative graded Hopf algebra

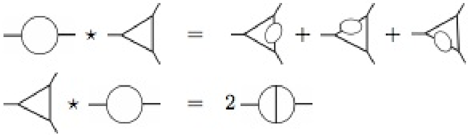
• $A =$ commutative algebra, $\mathrm{Hom}(\mathcal{H}, A) = \mathcal{G}(A)$ is a group

• the Hopf algebra $\mathcal{H}$ is determined by the Lie algebra $\mathcal{L}$ of $\mathcal{G}(\mathbb{C})$

• insertion Lie algebra of Feynman graphs

• given two graphs G_1, G_2: count in how many ways can insert one into the other at a vertex (so that external edges glued to corolla of edges at the vertex)

- Examples of graph insertions:



- pre-Lie structure: a bilinear map $\star : V \otimes V \to V$ on a vector space V

$$(x \star y) \star z - x \star (y \star z) = (x \star z) \star y - x \star (z \star y)$$

identity of associators under $y \leftrightarrow z$

Lie algebras and pre-Lie structures

• Lie algebra: vector space V with bilinear bracket $[\cdot,\cdot]$ operation with $[x,y] = -[y,x]$ and Jacobi identity

$$[x,[y,z]] + [z,[x,y]] + [y,[z,x]] = 0.$$

• tangent space at the identity of a Lie group is a Lie algebra

• Given a pre-Lie structure

$$[x,y] = x \star y - y \star x$$

is a Lie bracket (pre-Lie identity $\Rightarrow$ Jacobi identity)

• insertion of graphs is a pre-Lie operator $\Rightarrow$ Lie algebra

Lie algebra of Feynman graphs

- Lie bracket

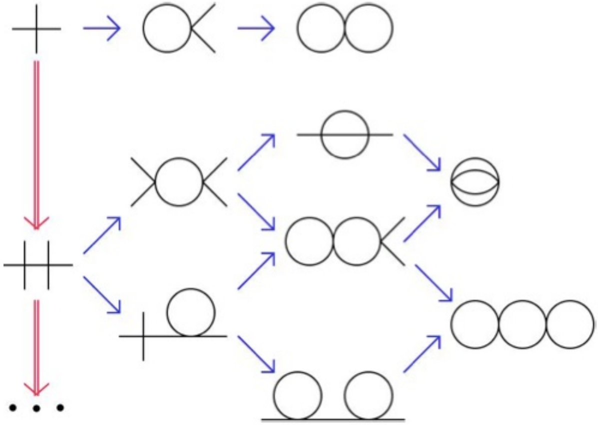
$$[G, G'] = \sum_{v \in V(G)} G \circ_v G' - \sum_{v' \in V(G')} G' \circ_{v'} G,$$

sum over vertices and counting all possible ways of inserting the other graph at that vertex matching external edges

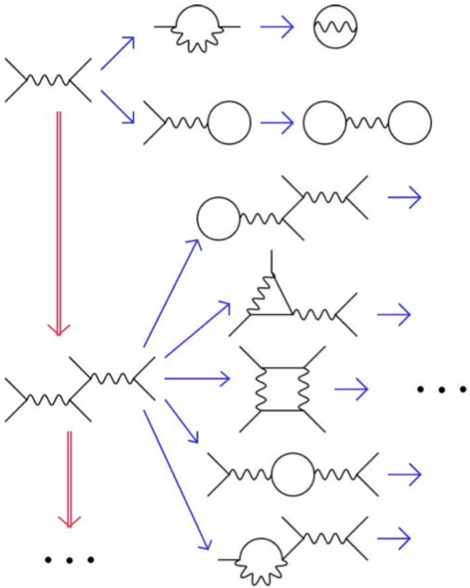
This story can be generalized

1. Any context free graph grammar determines an insertion Lie algebra and a commutative Hopf algebra
2. for context-sensitive it is more delicate: Lie-algebroid?
3. Feynman graphs of a QFT are a graph language

Example: Feynman graph language of ϕ^4-theory



Example: Feynman graph language of $\phi^2 A$-theory



- M. Marcolli, A. Port, *Graph Grammars and the Insertion Lie Algebras of Quantum Field Theory*, in preparation
- A. Connes, D. Kreimer, *Renormalization in quantum field theory and the Riemann-Hilbert problem. I.*, Communications in Mathematical Physics 210 (2000), no. 1, 249–273.
- A. Connes, D. Kreimer, *Insertion and elimination: the doubly infinite Lie algebra of Feynman graphs*. Ann. Henri Poincaré 3 (2002), no. 3, 411–433.
- K. Ebrahimi-Fard, J.M. Gracia-Bondia, F. Patras. *A Lie theoretic approach to renormalization.* Commun. Math. Phys 276, no. 2 (2007): 519-549.
- M. Bachmann, H. Kleinert, A. Pelster, *Recursive graphical construction for Feynman diagrams of quantum electrodynamics*, Phys.Rev. D61 (2000) 085017.
- H. Kleinert, A. Pelster, B. Kastening, M. Bachmann, *Recursive graphical construction of Feynman diagrams and their multiplicities in ϕ^4 and in $\phi^2 A$ theory*, Phys.Rev. E62 (2000) 1537–1559.