Coinductive guide to inductive transformer heads

Adam Nemecek
adam@cofunctional.ai

1 Abstract

We argue that all building blocks of transformer models can be expressed with a single concept: combinatorial Hopf algebra.

Transformer learning emerges as a result of the subtle interplay between the algebraic and coalgebraic operations of the combinatorial Hopf algebra. Viewed through this lens, the transformer model becomes a linear time-invariant system where the attention mechanism computes a generalized convolution transform and the residual stream serves as a unit impulse.

Attention-only transformers then learn by enforcing an invariant between these two paths. We call this invariant **Hopf coherence**. Due to this, with a degree of poetic license, one could call combinatorial Hopf algebras "tensors with a built-in loss function gradient". This loss function gradient occurs within the single layers and no backward pass is needed. This is in stark contrast to automatic differentiation which happens across the whole graph and needs a explicit backward pass. This property is the result of the fact that combinatorial Hopf algebras have the surprising property of calculating eigenvalues by repeated squaring.

2 Introduction

With the rise of popularity of transformer models [Vaswani et al., 2017], there have been concerns about the interpretability, or lack thereof, of said models. Such concerns are warranted, transformer models exhibit behaviors that appear surprising.

We analyze these behaviors in the context of combinatorial Hopf algebra, which is a tensorial bialgebra. This algebra turns out to be uniquely suited for interpretation and understanding of transformer models and, as we show in future work, machine learning models in general.

Despite the fact that this is, to our knowledge, the first paper which interprets machine learning in terms of Hopf algebras, we are confident that this venue of research will prove to be a quintessential semantic bridge between the representation of machine learning models in memory as IEEE 754 and the observable behaviors of said models.

The paper is structured as follows: section 3 discusses the idea of duality in general and briefly discusses how it has been used. Section 4 is an introduction to Hopf algebras, and section 5 extends that to combinatorial Hopf algebra. Section 6 discusses how these concepts apply to transformer models.

3 Duality

3.1 Induction

Induction, the basic idea behind algebra and recursion, is a well-understood concept in both mathematics and computer science. Fundamentally, induction is about building up objects from smaller components via the use of a **product** operation.

$$C \otimes C = C$$

Inductive data structures are defined in terms of their **constructors**

$$nil : 1 \to U$$

$$cons : D \times U \to U$$

where U is the the data type being defined. One can think of the linked list being formalized as:

$$\zeta : 1 + D \times U \to U$$

[Barbosa, 2022]

3.2 Coinduction

> "[...] coalgebras are about observation. We can think of a coalgebra $f : X \to 1 + AX$ as observing about an entity whether it contains something A-detectable or not, and if so which element of A it detects. Having observed something it modifies it. The final coalgebra has as elements all possible outcomes of the behavior you might observe. Do you still have observations to add as list elements? If ever no, we have a finite list. If always yes, we have an infinite list. And there's no other behavior that can be detected." [Corfield, 2011]

Coinduction, a dual of induction, has been relegated to the realm of computer science esoterica. Coinduction, and the related coalgebra, is about analysis, namely specifying how things break down by defining **coproduct**:

$$C = C \otimes C$$

Turning an algebra into a coalgebra is as simple as reversing the arrows. Using this principle, we can create **destructors** from linked list constructors above like so:

$$head : U \to 1$$

$$tail : U \to D \times U$$

The coinductive list formula is then:

$$\alpha : U \to 1 + D \times U$$

.

[Barbosa, 2022]
Coinduction is fundamentally about defining how something changes in response to observation.

As such, it is the underlying principle of recurrent relationships, generating functions, dynamic systems, coinductive data structures, and fundamentally anything that deals with modeling of observed, possibly infinite, behavior.

Here are some examples of coinduction:

3.2.1 Recurrence relations

Recurrence relations is a way of defining terms of a sequence in terms of combinations of previous elements of the sequence.

The connection between Hopf algebras and recurrence relations is discussed in [Peterson and Taft, 1980].

3.2.2 Generating function

> A generating function is a device somewhat similar to a bag. Instead of carrying many little objects detachedly, which could be embarrassing, we put them all in a bag, and then we have only one object to carry, the bag.
>
> George Pólya, Mathematics and plausible reasoning

Generating function is a generalization of formal power series, itself a generalization of power series.

Ordinary generating function are functions of this format:

$$G(a_n; x) = \sum_{n=0}^{\infty} a_n x^n$$

The nomenclature is not accurate, **generating functions are not functions** per se, they provide a way of expressing something in terms of a sum of lower dimensional elements with additional constraints on what combines with what via the use of "powers". However, these powers are indeterminate, one is not expected to substitute for x and evaluate. Generating functions are for handling infinite sums and establishing recurrences. As such, they have been used extensively in the context of analytic and enumerative combinatorics [Flajolet, 2009].

The powerful idea of generating functions is that they provide a unified interface between linearity and non-linearity due to the fact that generating functions provide a stream of semi-rings which can be further combined to a single generating function producing semi-rings.

As we will see, the this idea is closely related to the idea of coproduct in Hopf algebras.

In the context of machine learning, automatic differentiation can be seen as an an instance of generating functions [Carothers et al., 2012].

3.2.3 Stream algebra

We very briefly mention streams, or infinite lists, as they provide a good mental model for how to think about coalgebraic programming. Stream algebras are closely related to generating functions, and have been studied as a way of modeling recurrences, streaming automata, process algebras etc. One can think of them as "streaming representation-changers" [Gibbons, 2004] as they change their internal state in response to external changes.

One way of working with defining coalgebraic object is in terms of (often self-referential) streams or infinite lists by specifying an initial value and an update function. Due to the requirement for laziness, the Haskell programming language is a natural choice for this programming paradigm.

By self-referential we mean that the stream we are defining appears on both sides of the assignment operator.

Consider this Haskell definition a stream representing the natural numbers:

$$nats = 1 : map(+1)nats$$

Note how *nats* appears on both sides of the assignment operator and how it is defined in terms of an initial state and an update rule. It is for this reason that [Rutten, 2003] calls coinductive streams **differential equations of programming**. [Clenaghan, 2018] provides a complete Haskell implementation of such programming paradigm. Said paper praises the coinductive approach for it s "economy of statement and notation, whilst embracing variety of approach".

The fundamental difference between recursion and corecursion is that recursive Fibonacci function returns only a single value, while corecursive fibonacci returns a **stream** of all Fibonacci numbers.

The stream algebra has a close relationship with Hopf algebra [Bonchi et al., 2014].

3.2.4 Dynamic programming

Dynamic programming can be naturally expressed in terms of recurrences. The Bellman equation

$$V(s) = max_a(R(s,a) + \gamma V(s'))$$

is self-referential. The relationship between coalgebras and dynamic programming is discussed for example in [Hinze et al., 2015].

3.3 Applications

Coalgebras and coinduction have been slowly but surely gaining popularity.

This approach of building up things from their constituent parts enables reasoning about all possible paths and interactions in a black-box dynamic system by analyzing all the possible interdependent interactions among the single atoms starting from the bottom [Barbosa, 2022]. The field of bisimulation defines equivalence in terms of observation equivalences [Sangiorgi, 2012] and thereby verifies the dynamic behaviors of systems.

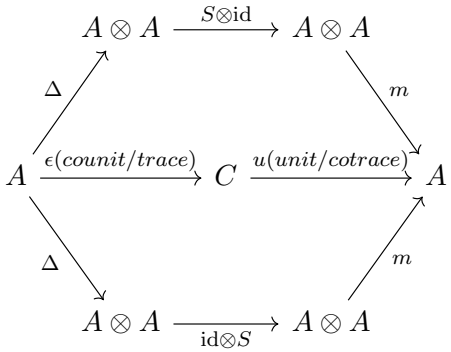
- [Vajjha et al., 2020] has applied coinduction in the context of verified reinforcement learning.
- [Hur et al., 2013] discusses how coinduction allows for building proofs incremental by combining small proofs into larger proofs.
- [Nguyen and Wu, 2022] discusses backpropagation as a coalgebra.
- [Mastorou et al., 2022] extend Haskell to verify coinductive proofs.
- [Pous, 2016] discusses how coinduction enabled validating global properties by checking only local properties.
- [Kozen and Silva, 2017] introduced *CoCaml* which discusses the idea of coinductive programming as programming with a coinductive equation solver.

4 Hopf algebra

Hopf algebra is a tensor bialgebra A over a field C meaning it is both a tensor algebra and a tensor coalgebra at once.

In the diagram below, we refer to the counit-unit path as "unit impulse path" and the top and bottom paths as the "convolution paths".

Hopf algebra enforces coherence between the two paths by updating internal state in response to input [Ahman and Uustalu, 2014]. This is a very powerful invariant for reasoning about the behavior of a linear time-invariant system [Bonchi et al., 2014].



Considering popularity of Hopf algebra in other scientific fields [Hazewinkel, 2004] it was only a matter of time before they are used in machine learning.

Hopf algebra is defined by the following operations.

- unit: u: $C \to A$
- product: m: $A \otimes A \to A$
- counit: $\epsilon : A \to C$
- coproduct: $\Delta : A \to A \otimes A$
- antipode: $S : A \to A$

4.1 Unit

The definition of unit is straightforward, it takes an element of C and construct and element of A:

$$u : C \to A$$

and obeys the rule

$$u \cdot e = e = e \cdot u$$

for all elements of A.

One should however think of the unit in relationship to the equalizer.

4.2 Product

The product is the standard tensor product

$$(a \otimes c)(b \otimes d) = (ab \otimes cd)$$

4.3 Counit

Counit is similar to the idea of trace from linear algebra and as such provides a feedback loop operator from control theory [Hasegawa and Lemay, 2022].

It is a dual of the unit and as such it works as a coequalizer.

4.4 Coproduct

The coproduct provides a way of generating all possible splittings of a subset into disjoint pieces [Majid, 1995].

$$\Delta(x^n) = \sum_{i=0}^{n} x^i \otimes x^{n-i}$$

where $x^0 = 1$.

One can think of it as converting something into its generating function [Hazewinkel, 2005] or into a sum of elements of lower graded subcoalgebra such as simplices.

In the context of physics, it is a probability density function, a total probability mass that's being shared out among different spaces [Majid, 1995].

$$\begin{aligned}
\Delta(wz) &= \Delta(w)\Delta(z) \\
\Delta(1) &= 1 \otimes 1 \\
\Delta(x) &= 1 \otimes x + x \otimes 1 \\
\Delta(x^2) &= 1 \otimes x + x \otimes x + x \otimes 1
\end{aligned}$$

For a more complicated example:

$$\begin{aligned}
(id \otimes \Delta)(\Delta(x^2)) = & \\
&= (id \otimes \Delta)(1 \otimes x^2 + x \otimes x + x^2 \otimes 1) \\
&= 1 \otimes \Delta(x^2) + x \otimes \Delta(x) + x^2 \otimes \Delta(1) \\
&= 1 \otimes (1 \otimes x^2 + x \otimes x + x^2 \otimes 1) + x \otimes (1 \otimes x + x \otimes 1) + x^2 \otimes (1 \otimes 1) \\
&= 1 \otimes 1 \otimes x^2 + 1 \otimes x \otimes x + 1 \otimes x^2 \otimes 1 + x \otimes 1 \otimes x + x \otimes x \otimes 1 + x^2 \otimes 1 \otimes 1
\end{aligned}$$

Future work will discuss how superposition emerges out of an extension of a product and coproduct, namely the phased biproduct [Tull, 2018]. But fundamentally, this arises out of the fact that when multiplying a sequence of Hopf algebras, one can select which ones to multiply and as a result, the coproduct represents a combinatorial object which includes all possible ways to choose [Diaconis et al., 2012].

4.5 Antipode

Since not all transformations are invertible, a weaker structure, **antipode**, provides a nonlocal "linearized inverse". Antipode doesn't provide an inverse for single elements but for linear combinations. One can think of the antipode as a complex conjugate. However, if the Hopf algebra is finite dimensional, then the antipode is an inverse [Majid, 1995].

The antipode is a defined as:

$$S \cdot S = id$$

$$S(hg) = S(g)S(g)$$

The antipode serves as an intertwining operator, namely a equivariant linear map between two representations which gets updated as the Hopf algebra "learns".

In the context of interacting particle systems, the intertwiner provides a symmetric exclusion process [Redig and Sau, 2018].

The importance of the antipode will become apparent in the context of Hopf convolution.

4.6 Unit impulse, Convolution & Hopf coherence

Given the algebraic operations of the Hopf algebra, one can define the unit impulse δ, also known as Dirac delta, as:

$$\delta = \epsilon * u$$

[Christiansen et al., 2022] discusses how traces of evolution operators can be evaluated as integrals over Dirac delta functions.

The purpose of the unit impulse is to serve as multiplicative identity for convolution $*$ which is defined as:

$$(f * g) = m(f \otimes g)\Delta$$

The fundamental advantage of this convolution over the standard one is, as is the custom with bi-algebras, it provides a way of controlling how splitting-up and recombination works via the coproduct and product.

The unit impulse delta function then provides the multiplicative identity:

$$f * \delta = f$$

Arguably the most important property of Hopf algebra is the fact that this invariant has to hold:

$$m(id \otimes S)\Delta = \epsilon * u$$

We refer to this invariant as the **Hopf coherence**.

Hopf coherence enforces that the algebra updates its internal state in order for the unit impulse path to be equal the convolution path.

The antipode plays an important part in enforcing this. The antipode reconstructs itself in order for this coherence to hold. This reconstruction process is a result of the "Tannaka–Krein duality" or "Tanaka reconstruction theorem" [nca,] [Pareigis, 1994]. The details of this reconstruction process as well as a proof thereof is provided in [Majid, 1995].

The exact antipode reconstruction algorithm has been a subject of much research for example in [Berlin, 2019].

5 Combinatorial Hopf algebra

Combinatorial Hopf algebra is a Hopf algebra where the coproduct is defined as the **shuffle product**. Besides combinatorics [Grinberg and Reiner, 2014], this algebra has also been explored in the context of control theory [Espinosa et al., 2014]. For a classical introduction into the material, consider [Joni and Rota, 1979].

5.1 Shuffle product

Originally, the shuffle product arose in the context of card shuffling.

In the setting of non-commutative algebras setting, the shuffle product provides a way of generating all the ways in which two words can be interwoven. For another example, imagine a situation where cars from two lanes merge into one lane. The shuffle product is a combinatorial object that represents all the possible ways in which cars from the two lanes can be interleaved to merge into one. Since word composition is non-commutative, the shuffle product is a natural match for representing combinatorial objects of words.

For example:

$$ab \sqcup\!\sqcup ab = 4aabb + 2abab$$
$$ab \sqcup\!\sqcup ba = abab + 2abba + 2baab + baba$$

[Lothaire, 1997]

The most surprising property of the combinatorial Hopf algebra is that one can calculate the eigenvectors by repeated squaring (coproduct-product) [Aguiar and Lauve, 2013].

[Pang, 2014] provides some intuition into this coproduct-product operation and how they correspond a repeated splitting and combination of combinatorial objects.

[Diaconis et al., 2012] uses this property in the context of diagonalization of Markov chains in natural bases.

The shuffle product also allows for interpreting differential equations combinatorially [Mishna and Zabrocki, 2008].

6 Transformers

Since first being introduced in [Vaswani et al., 2017], the transformer architecture has become one of the most widely studied architectures and as such has been discussed extensively for example by [Phuong and Hutter, 2022], [Elhage et al., 2021]. Due to this, we only concentrate on selected parts namely the attention mechanism and the residual stream.

In this section, we argue that the attention-only transformer model can be understood in terms of combinatorial Hopf algebras and as such can be analyzed as a **linear time-invariant system** [Espinosa et al., 2018].

The summary of our argument is that since the residual stream has no preferred basis, it should be understood as a trace/counit/unit impulse, and since the attention mechanism is a positive definite matrix, it should be understood as a transfer function of the system. The model learns by enforcing Hopf coherence between the two paths.

The attention heads continuously update the residual stream (trace) and their internal state in order to enforce the invariance between the unit impulse path and Hopf coherence.

Combinatorial Hopf algebra is a good match since it captures the non-commutativity of word composition.

6.1 Residual stream

The residual stream has been described as a channel used by single components of the model to communicate among themselves.

We argue that the **residual stream is the trace/counit/unit impulse** of the model. The major hint is that the residual stream has been observed to have **no privileged basis**

[Elhage et al., 2021]. This is analogous to the behavior of the trace in linear algebra where the trace of a matrix is also independent of the basis. While in linear algebra the trace is a scalar, **categorical trace is more akin to formal power series** [Hines and Scott, 2007].

As such the residual stream serves a similar purpose to the evaluation trace in automatic differentiation, namely establishing recurrences. This is unsurprising considering the fact that automatic differentiation is a recurrence relationship based on formal power series [Carothers et al., 2012], in particular the Taylor series [Hoffmann, 2014].

The residual stream is a recurrence relationship that attention heads use for both steering (by writing into it) and being steered by it (by reading from it). The values in the stream are continuously updated in response to input in order to enforce Hopf coherence.

The residual stream provides the notion of a feedback analogous to feedback in control theory or linear systems while the attention mechanism is the transfer function.

In the context of linear time-invariant systems, the transfer function is the inverse Laplace transform of the unit impulse response and vice-versa.

This duality can be understood as the trace-transfer function duality [Dold and Puppe, 1985].

It can also be understood as the duality between the trace and the fixed point [Hasegawa, 2004] that's analogous to Tannaka–Krein duality. By interacting with the residual stream, attention heads are steering the direction of the fixed point search algorithm that the transformer performs.

Future research will investigate the exact nature of this interaction.

6.2 Attention

The attention mechanism's role is the same as that of a transfer function in a linear time-invariant system, namely it calculates the frequency response of the transformer model, in the case of transformers, the output token.

Our interpretation is based on the fact that attention matrices have been observed to have real positive eigenvalues. From that we conclude that they are symmetric positive definite matrices. This symmetry enables an input-output symmetry [Majid, 1995].

Attention mechanism operates on three inputs: **query**, **key**, **value** and generates one **output**. Please reference [Vaswani et al., 2017] for explanation of these terms.

[Elhage et al., 2021] formalizes the model as such:

$$T = Id \otimes W_U W_E + \sum_{h \in H} A^h \otimes (W_U W_O^h W_V^h W_E)$$

$$A = softmax(t^T \cdot W_E^T W_Q^T W_K W_E \cdot t)$$

Furthermore, they analyze the attention mechanism by splitting it into two circuits, *QK (query-key)*, and OV *(output-value)* and the associated matrices W_{QK} and W_{OV}.

These two circuits roughly correspond to coproduct-product relationship of the convolutional path where the coproduct splits things and product recombines them as discussed in [Diaconis et al., 2012] or [Redig and Sau, 2018].

6.2.1 QK circuit

The QK circuit assigns an attention score for a given query and key token. This score indicates how much the query wants to attend to the particular key token. We mostly agree with the interpretation of [Chen, 2021] and [Yuan et al., 2021] which understand this circuit as Markov chain transition matrices, as they are row-stochastic matrices, with each row summing to 1.

$$\sum_{j=1}^{\alpha} P_{i,j} = 1$$

This circuit fundamentally distributes the unit of attention among the potential candidates. Since it is a distributive operation, it is to be understood as a part of the coalgebraic circuit.

These weights can be understood as **stochastic recurrence relations**. They are kept around for as long as they are "interesting" and after that, they are gradually pruned.

Since the QK circuit generates the possible candidates, it can be understood as providing induced representation of the Markov chain. Induced representation is a tool in representation theory that enables building representations of large objects from representations of small objects. [Schmitt, 1993] discusses this in more detail.

> "Coinjection is a partially-defined function whose restriction to where it is defined is a bijection; an example is $f : 1, 2, 3, 4 \to 7, 8$ with $f(1) = 8$, $f(3) = 7$, and $f(2)$, $f(4)$ undefined). Intuitively, these are combinatorial structures with a notion of restriction on a subset of their vertex set; one can restrict a graph to a subset of its vertices by considering only the edges connected to this subset (usually known as the *induced subgraph*)".
>
> [Pang, 2014]

6.2.2 OV circuit

The OV circuit operates on the candidate values and combines them to produce a single output. Therefore it belongs to the product circuit.

[Elhage et al., 2021] discusses the behavior of the OV circuit in terms of eigenvalues of the OV matrix. It was observed that there's a strong indication that if the matrix has positive eigenvalues, it is likely copying. Since the eigenvalues are real and positive we can conclude that it is a symmetric positive definite matrix.

A symmetric positive definite matrix, since it is a self-adjoint operator, is equal to its own conjugate transpose, i.e.

$$A = A^T$$

Self-adjointness can be thought of as enforcing conservation of energy laws [Ibragimov, 2011]. If we interpret the QK circuit as branching, we can interpret the the OV circuit as combining in the sense of [Pang, 2014].

As such it can be thought of as the antipode [Li and Zhang, 2020] or intertwining operator.

In the context of Markov chains, **positive definite matrices can be understood as modeling interactions between Markov chains** [O'Connell, 2019].

6.3 Transformer learning mechanism

The most important property of the residual stream is its basis independence, while the most important property of the attention mechanism is its symmetry and positive definiteness.

In our interpretation of transformers, the transformer model learns by enforcing Hopf coherence between the residual stream (unit response path) and the attention path (convolution path).

[Elhage et al., 2021] describes the attention mechanism as "sum where every term corresponds to an end-to-end path". This is fundamentally convolution as it can be considered a sum of all impulse responses [Cheever, 2022].

In our model learning works as follows:

1. attention head receives a input
2. attention head calculates output from both the unit response path and convolution path
3. if the outputs match, the token gets copied
4. if the outputs differ, the transformer propagates the difference (error) backward along the convolution path and unit impulse path, distributes the error between the QK and OV mechanisms and updates the residual stream (trace)
5. by repeating this coproduct-product process (squaring) with different inputs, the transformer learns a stationary Markov chain distribution in its natural basis [Diaconis et al., 2012]

This mechanism of calculating the gradient of the loss function provides a better alternative to automatic differentiation since the error is calculated within the single layer and there's no explicit backward pass.

The details of this mechanism will be discussed in future work, however the intuition is that it is related to Laplace transform and Tannaka–Krein duality.

Laplace transform has been observed to arise out of the interaction between convolution and shuffle product of combinatorial Hopf algebras [Rutten, 2019].

Tannaka–Krein, as a non-commutative generalization of Pontryagin duality, then obviates the relationship to Laplace transform.

[Pavlovic and Escardo, 2001] also explores the Laplace transform in the context of coinduction.

Another way of looking at it is via conjectured relationship between the induced representation (QK) and the intertwining operator (OV) on one end and the trace (residual stream) on the other [Arthur, 2008].

Dynamic programming provides a simplified, if not simplistic, model of certain elements of transformer models that is nonetheless rather useful. The main difference is that most aspects of dynamic programming algorithms are, rather ironically, quite static. In comparison, the transformer prunes its recurrences when combining a solution from subsolutions.

The interpretation in the context of DP also explains the "phase change" of transformer models. When the DP algorithm first starts filling out the grid, a lot of fields will be updated over a short period of time. Gradually, the frequency of the updates will decrease and the previous subsolutions will just be copied over.

6.4 Attention head composition

Zero layer attention only model learns bigram statistics. Since a bigram is just a Markov chain, these can be thought of as Hopf algebras without induced representation and intertwining operators. As a result, squaring the Hopf algebra similarly to [Diaconis et al., 2012] makes the model learn a bigram. Chapter 5 of [Majid, 1995] discusses how Hopf algebras can be used to model Markov chains.

Ironically, in our model, the one layer and two layer attention models are somewhat similar.

One layer attention only model then learns an ensemble of bigram and skip-gram and the two layer attention head then learns induction heads.

The exact mechanism how this works will be explored in future work, however the intuition is that since the shuffle product allows us to generate possible interleavings and ways of combining them, the chain "A ... B C" can be understood as a sum over all the coproducts which start with "A" and end with "B C".

[Ebrahimi-Fard et al., 2019] discusses these matters in terms of gap-insertion operad of non-crossing partitions.

7 Conclusion and future work

To summarize this paper, combinatorial Hopf algebras provides a rich algebraic infrastructure to interpret transformer models. This algebra has the surprising property of being able to calculate eigenvalues by repeated squaring.

By interpreting transformers as Hopf algebras, we arrive at a new understanding of the transformer learning mechanism as enforcement of Hopf coherence.

This mechanism boils down to enforcing coherence between the unit impulse path and the convolution path via Tannaka–Krein duality. As such, the transformer model can be understood as a linear time-invariant system.

In future work, we will discuss Hopf coherence in more detail.

References

[nca,] Reconstruction theorem in nlab.

[Aguiar and Lauve, 2013] Aguiar, M. and Lauve, A. (2013). Convolution powers of the identity antipode and convolution powers of the identity in graded connected hopf algebras. pages 1053–1064.

[Ahman and Uustalu, 2014] Ahman, D. and Uustalu, T. (2014). Coalgebraic update lenses. *Electronic Notes in Theoretical Computer Science*, 308:25–48.

[Arthur, 2008] Arthur, J. (2008). Induced representations, intertwining operators and transfer.

[Barbosa, 2022] Barbosa, L. S. (2022). Coalgebra for the working software engineer. *Journal of Applied Logics-IFCoLog Journal of Logics and their Applications*, 9.

[Berlin, 2019] Berlin, L. (2019). Computational analysis of antipode algorithms for the output computational analysis of antipode algorithms for the output feedback hopf algebra feedback hopf algebra.

[Bonchi et al., 2014] Bonchi, F., Sobocinski, P., and Zanasi, F. (2014). Interacting hopf algebras. *Journal of Pure and Applied Algebra*, 221:144–184.

[Carothers et al., 2012] Carothers, D. C., Lucas, S. K., Parker, G. E., Rudmin, J. D., Sochacki, J. S., Thelwell, R. J., Tongen, A., and Warne, P. G. (2012). Connections between power series methods and automatic differentiation. *Lecture Notes in Computational Science and Engineering*, 87 LNCSE:175–185.

[Cheever, 2022] Cheever, E. (2022). The convolution as a sum of impulse responses.

[Chen, 2021] Chen, Y. (2021). The brownian motion in the transformer model.

[Christiansen et al., 2022] Christiansen, F., Cvitanovic', P., and Putkaradze, V. (2022). *Chaos: Classical and Quantum: Trace formulas*.

[Clenaghan, 2018] Clenaghan, K. (2018). In praise of sequence (co-)algebra and its implementation in haskell.

[Corfield, 2011] Corfield, D. (2011). Understanding the infinite ii: Coalgebra. *STUDIES IN HISTORY AND PHILOSOPHY OF SCIENCE*.

[Diaconis et al., 2012] Diaconis, P., Pang, C. Y. A., and Ram, A. (2012). Hopf algebras and markov chains: Two examples and a theory. *Journal of Algebraic Combinatorics*, 39:527–585.

[Dold and Puppe, 1985] Dold, A. and Puppe, D. (1985). Duality, trace and transfer. *Proceedings of the Steklov Institute of Mathematics*, pages 85–103.

[Ebrahimi-Fard et al., 2019] Ebrahimi-Fard, K., Foissy, L., Kock, J., and Patras, F. (2019). Operads of (noncrossing) partitions, interacting bialgebras, and moment-cumulant relations. *Advances in Mathematics*, 369.

[Elhage et al., 2021] Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., DasSarma, N., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., and Olah, C. (2021). A mathematical framework for transformer circuits. *Transformer Circuits Thread*. https://transformer-circuits.pub/2021/framework/index.html.

[Espinosa et al., 2014] Espinosa, L. A. D., Ebrahimi-Fard, K., and Gray, W. S. (2014). A combinatorial hopf algebra for nonlinear output feedback control systems. *Journal of Algebra*, 453:609–643.

[Espinosa et al., 2018] Espinosa, L. A. D., Ebrahimi-Fard, K., and Gray, W. S. (2018). Combinatorial hopf algebra for interconnected nonlinear systems. 267.

[Flajolet, 2009] Flajolet, P. (2009). Analytic combinatorics (e-version).

[Gibbons, 2004] Gibbons, J. (2004). Streaming representation-changers. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3125:142–168.

[Grinberg and Reiner, 2014] Grinberg, D. and Reiner, V. (2014). Hopf algebras in combinatorics.

[Hasegawa, 2004] Hasegawa, M. (2004). The uniformity principle on traced monoidal categories. 40:991–1014.

[Hasegawa and Lemay, 2022] Hasegawa, M. and Lemay, J.-S. P. (2022). Traced monads and hopf monads.

[Hazewinkel, 2004] Hazewinkel, M. (2004). Pervasiveness of hopf algebras.

[Hazewinkel, 2005] Hazewinkel, M. (2005). Endomorphisms of hopf algebras and a little bit of control. *Lecture Notes in Control and Information Sciences*, 321:107–122.

[Hines and Scott, 2007] Hines, P. and Scott, P. (2007). Categorical traces from single-photon linear optics.

[Hinze et al., 2015] Hinze, R., Wu, N., and Gibbons, J. (2015). Conjugate hylomorphisms or: The mother of all structured recursion schemes. page 15.

[Hoffmann, 2014] Hoffmann, P. H. (2014). A hitchhiker's guide to automatic differentiation. *Numerical Algorithms*, 72:775–811.

[Hur et al., 2013] Hur, C. K., Neis, G., Dreyer, D., and Vafeiadis, V. (2013). The power of parameterization in coinductive proof.

[Ibragimov, 2011] Ibragimov, N. H. (2011). Nonlinear self-adjointness and conservation laws. *Journal of Physics A: Mathematical and Theoretical*, 44.

[Joni and Rota, 1979] Joni, S. A. and Rota, G.-C. (1979). Coalgebras and bialgebras in combinatorics. *Studies in Applied Mathematics*, 61:93–139.

[Kozen and Silva, 2017] Kozen, D. and Silva, A. (2017). Practical coinduction. *Mathematical Structures in Computer Science*, 27.

[Li and Zhang, 2020] Li, L. and Zhang, P. (2020). Twisted hopf algebras, ringel - hall algebras, and green's categories. *Journal of Algebra*, 231.

[Lothaire, 1997] Lothaire, M. (1997). *Combinatorics on Words*. Cambridge University Press, 2 edition.

[Majid, 1995] Majid, S. (1995). *Foundations of Quantum Group Theory*. Cambridge University Press.

[Mastorou et al., 2022] Mastorou, L., Papaspyrou, N., and Vazou, N. (2022). Coinduction inductively: mechanizing coinductive proofs in liquid haskell. pages 1–12.

[Mishna and Zabrocki, 2008] Mishna, M. and Zabrocki, M. (2008). Analytic aspects of the shuffle product. pages 561–572.

[Nguyen and Wu, 2022] Nguyen, M. and Wu, N. (2022). Folding over neural networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 13544 LNCS:129–150.

[O'Connell, 2019] O'Connell, N. (2019). Interacting diffusions on positive definite matrices. *Probability Theory and Related Fields*, 180:679–726.

[Pang, 2014] Pang, C. Y. A. (2014). Hopf algebras and markov chains. *arXiv preprint arXiv:1412.8221*.

[Pareigis, 1994] Pareigis, B. (1994). Reconstruction of hidden symmetries.

[Pavlovic and Escardo, 2001] Pavlovic, D. and Escardo, M. (2001). Calculus in coinductive form. *Esca*.

[Peterson and Taft, 1980] Peterson, B. and Taft, E. J. (1980). The hopf algebra of linearly recursive sequences. *Aequationes Mathematicae*, 20:1–17.

[Phuong and Hutter, 2022] Phuong, M. and Hutter, M. (2022). Formal algorithms for transformers.

[Pous, 2016] Pous, D. (2016). Coinduction all the way up. volume 05-08-July-2016.

[Redig and Sau, 2018] Redig, F. and Sau, F. (2018). Stochastic duality and eigenfunctions. *Springer Proceedings in Mathematics and Statistics*, 282:621–649.

[Rutten, 2019] Rutten, J. (2019). *The Method of Coalgebra: exercises in coinduction*.

[Rutten, 2003] Rutten, J. J. (2003). Behavioural differential equations: A coinductive calculus of streams, automata, and power series. *Theoretical Computer Science*, 308.

[Sangiorgi, 2012] Sangiorgi, D. (2012). An introduction to bisimulation and coinduction.

[Schmitt, 1993] Schmitt, W. R. (1993). Hopf algebras of combinatorial structures. *Canadian Journal of Mathematics*, 45.

[Tull, 2018] Tull, S. (2018). A categorical reconstruction of quantum theory. *Logical Methods in Computer Science*, 16:4:1–4:39.

[Vajjha et al., 2020] Vajjha, K., Shinnar, A., Trager, B., Pestun, V., and Fulton, N. (2020). Certrl: Formalizing convergence proofs for value and policy iteration in coq. *CPP 2021 - Proceedings of the 10th ACM SIGPLAN International Conference on Certified Programs and Proofs, co-located with POPL 2021*, pages 18–31.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Łukasz Kaiser, and Polosukhin, I. (2017). Attention is all you need. volume 2017-December.

[Yuan et al., 2021] Yuan, T., Li, X., Xiong, H., Cao, H., and Dou, D. (2021). Explaining information flow inside vision transformers using markov chain.