# A Deep Inference Model of State

Jamie Brine

April 11, 2025

**Abstract**

Made a thing, it did a thing, very happy.

# Contents

# 1 Introduction

## 1.1 Modelling Computation

Proof theorists have developed countless proof systems as mechanisms with which to model computation. Each system has its own grammar and rules, aiming to capture a particular aspect of how a computer programs execute. They may choose to model resource sensitivity in producer/consumer relationships [7], shared-memory parallel code [16], or compiler optimisation techniques [15].

The key idea underpinning these models is the Curry-Howard correspondance. This was first proposed by Haskell Curry in the 1940s, then explicitly formulated by William Alvin Howard in the 1960s. A typeset version of Howard's original hand-written notes have been published [6]; for a more modern and detailed explaination, the reader is referred to a Master's thesis by Husna Farooqui [2]. The correspondence can be distilled to the following phrase:

"Propositions are types, and proofs are programs"

More formally, it provides a framework in which constructing a proof corresponds to the execution of a program. The return type of the program is precisely the type that the conclusion of the proof represents. This connection allows for programs to studied through the lens of logic, particularly with respect to properties like correctness and termination; if a proof is valid in the underlying system, so too is the program it corresponds to.

## 1.2 Deep Inference

This foundational idea has been extended and enriched with the advent of deep inference, a relatively recent development in proof theory. Deep inference is a methodology that contrasts with traditional approaches such as natural deduction and the sequent calculus, in which inference rules are restricted to apply only at the outermost (or top-level) structure of formulae. In contrast, deep inference systems permit inference rules to be applied at any depth within a formula's structure. This increased freedom introduces new possibilities for proof compression, locality, and symmetry. In turn, the types and programs that can be encoded by these systems are more expressive and finely controlled, allowing for richer representations of computational processes and more efficient proof search.

This flexibility is exemplified in deep inference systems such as BV and NEL. BV [1] is a system based on multiplicative linear logic, where a self-dual non-commutative connective is treated using deep inference principles. NEL [5] further extends these ideas by incorporating self-duplicating structures. For a comprehensive explaination of deep inference, and how its relatively subtle differences to classical proof techniques allow for staggering improvements in efficiency, the reader is referred to Guglielmi [4]. His pioneering work throughout the 2000s contributed greatly to the development of the methodology.

## 1.3 How to Model State?

Simply put, the state of a program refers to the set of values or conditions that exist at any given point during its execution, encompassing variables, data structures, and other program-

related information that influence its behavior. Modeling state is essential for understanding how programs evolve over time, tracking the effects of operations, and ensuring correctness in the program's execution.

Uday Reddy proposed the idea of modelling state with an extension of linear logic. In a 1993 paper [12], he proposes a logical system which extends the well studied system of linear logic with two new operators: a connective "before" which is used to impose order on terms in a proposition, and a regenerative storage operator "dagger" which allows these ordered terms to produce "traces". He explains that "a trace denotes the information extracted from a storage object in one particular execution of a program". He calls this system the Linear Logic Model of State.

Reddy also gives semantics to his system using a coherence space model. Semantic models of logical systems are valuable, as their category-theoretic nature allows for a more abstract and structured understanding of the relationships between logical formulas and their proofs, enabling the exploration of proofs as morphisms within a category. This provides powerful tools for reasoning about logical systems, facilitating the comparison of different proof systems and allowing for the transfer of results across various areas of logic and computation.

### 1.4 Motivation

In a preliminary version of this paper [13], Reddy acknowledges that "for the regenerative type system, we need a noncommutative tensor product which has all the properties of tensor except for the commutative property". This motivates a natural continuation of his work: can we add non-commutativity to the Linear Logic Model of State in order to more richly model program state?

In this paper, we present a new logical system, sWIP, with which we attempt to answer this question. It is a deep inference system similar to those mentioned previously, which offers a more flexible framework for reasoning about noncommutative structures within logic. This is particularly useful for modeling complex interactions within program state, where the order of operations may be significant.

We give a full coherence space model for the system, which allows us to study it mathematically and prove a number of useful facts about its nature. Crucially, we aim to show that the categorical model of any proof is a dinatural transformation. The equivalent notion in the logic is that no matter which pair of objects are being related, the structure of the relationship remains invariant, providing a deeper understanding of how proof transformations preserve logical properties. This is an important step in proving the consistency of the logic.

# 2 System sWIP

This system is a modification of NEL, first proposed by Guglielmi and Straßburger in 2002 [5]. The main difference between NEL and sWIP is that the ! and ? structures, which are referred to collectively as the *exponentials*, have been replaced by † and $\vartheta$. These new exponentials can be considered ordered, which is reflected in the modified $\mathsf{b}{\downarrow}$ and $\mathsf{b}{\uparrow}$ rules governing their regeneration.

## 2.1 Structures and Equivalence

**Definition 2.1.** *Structures* in sWIP, denoted $R$,$S$,$T$,$V$.., are defined by the following grammar:

$$R ::= \circ \mid a \mid \overline{R} \mid \dagger R \mid \vartheta R \mid (R, .., R) \mid \langle R; ..; R\rangle \mid [R, .., R]$$

where

- $\circ$ is the *unit*, which is common to all structures and also self dual; this can be thought of as an empty structure
- $a$ is an *atom*, of which there are countably many
- $\overline{R}$ is the *dual* of $R$
- $\dagger R$ and $\vartheta R$ are the *dagger* and *hash* of $R$ respectively
- $(R, .., R)$, $\langle R; ..; R\rangle$, and $[R, .., R]$ are *copar*, *seq*, and *par* structures; they are considered *proper* if they contain at least two elements

**Definition 2.2.** A structure with an empty hole, $S\{\ \}$, is called a *context*. We say $R$ is a *substructure* of $S\{R\}$. If structural parentheses fill the hole exactly, the curly braces will be ommitted, so for example $S\{[R, T]\}$ becomes $S[R, T]$.

**Definition 2.3.** Structures $R$ and $S$ are considered *equivalent* modulo the relation $=$, which is the smallest congruence defined by the equations in Figure 1.

## 2.2 Rewrite Rules, Derivations, and Proofs

We describe how structures can be modified using a set of rules, and how these rules can be chained together to form longer derivations. In particular, we are interested in those derivations which begin with an empty premise, which we will call proofs.

**Definition 2.4.** A *rewrite rule* is a rule $\rho$ with *premise* T and *conclusion* R, denoted $\rho\dfrac{T}{R}$. All of the rewrite rules in sWIP are given in Figure 2.

**Definition 2.5.** A *derivation* $\Delta$ from $R$ to $T$ is a finite chain of rewrites with premise $R$ and conclusion $T$, denoted $\begin{array}{c}R\\ \Delta\|\\ T\end{array}$. A derivation whose premise is $\circ$ is called a *proof*, to which we can add a topmost instance of the *axiom* rule $\circ{\downarrow}\dfrac{}{\circ}$; these are denoted $\begin{array}{c}\Delta\|\\ T\end{array}$.

**Associativity**

$$(R,(T)) = (R,T)$$
$$\langle R;\langle T\rangle;U\rangle = \langle R;T;U\rangle$$
$$[R,[T]] = [R,T]$$

**Commutativity**

$$(R,T) = (T,R)$$
$$[R,T] = [T,R]$$

**Unit**

$$(R,T,\circ) = (R,T)$$
$$\langle R;T;\circ\rangle = \langle R;T\rangle$$
$$\langle \circ;R;T\rangle = \langle R;T\rangle$$
$$[R,T,\circ] = [R,T]$$

**Singleton**

$$(R) = \langle R\rangle = [R] = R$$

**Dual**

$$\overline{\circ} = \circ$$
$$\overline{(R,T)} = [\overline{R},\overline{T}]$$
$$\overline{\langle R;T\rangle} = \langle \overline{R};\overline{T}\rangle$$
$$\overline{[R,T]} = (\overline{R},\overline{T})$$
$$\overline{\dagger R} = \vartheta\overline{R}$$
$$\overline{\vartheta R} = \dagger\overline{R}$$
$$\overline{\overline{R}} = R$$

**Contextual Closure**

if $S\{R\} = S\{T\}$,
then $R = T$

Figure 1: Syntactic Equivalence

$$\mathsf{ai}{\downarrow}\,\frac{S\{\circ\}}{S[R,\overline{R}]} \qquad \mathsf{d}{\downarrow}\,\frac{S\{\circ\}}{S\{\dagger\circ\}} \qquad \mathsf{q}{\uparrow}\,\frac{S(\langle R;T\rangle,\langle U;V\rangle)}{S\langle(R,U);(T,V)\rangle}$$

$$\mathsf{s}\,\frac{S([R,T],U)}{S[(R,U),T]} \qquad \mathsf{w}{\downarrow}\,\frac{S\{\circ\}}{S\{\vartheta R\}} \qquad \mathsf{p}{\uparrow}\,\frac{S(\vartheta R,\dagger S)}{S\{\vartheta(R,S)\}}$$

$$\mathsf{q}{\downarrow}\,\frac{S\langle[R,T];[U,V]\rangle}{S[\langle R;U\rangle,\langle T;V\rangle]} \qquad \mathsf{b}{\downarrow}\,\frac{S\langle R;\vartheta R\rangle}{S\{\vartheta R\}} \qquad \mathsf{w}{\uparrow}\,\frac{S\{\dagger R\}}{S\{\circ\}}$$

$$\mathsf{p}{\downarrow}\,\frac{S\{\dagger[R,T]\}}{S[\dagger R,\vartheta T]} \qquad \mathsf{ai}{\uparrow}\,\frac{S(R,\overline{R})}{S\{\circ\}} \qquad \mathsf{b}{\uparrow}\,\frac{S\{\dagger R\}}{S\langle R;\dagger R\rangle}$$

Figure 2: Rewrite Rules for System sWIP

We can turn any derivation into a proof with the following proposition, whose proof uses standard proof theoretical notions common to many systems.

**Proposition 2.6.** *If* $\Delta \begin{Vmatrix} R \\ T \end{Vmatrix}$ *is a derivation, then there exists a proof* $\frac{\Delta \Vdash}{(\overline{R},T)}$.

**Remark.** As sWIP is a deep inference system, a rewrite rule can be applied at arbitrary depth within any structure.

**Example 2.7.** We give a derivation where, when read top down, the substructure on which the rule is being applied is bolded. Note that this is not the simplest derivation from this premise to this conclusion, but has been chosen to demonstrate applying the rules at different levels within the structure.

$$\mathsf{b}\uparrow\ \dfrac{\mathsf{p}\downarrow\ \dfrac{\mathsf{q}\downarrow\ \dfrac{\mathsf{b}\downarrow\ \dfrac{\boldsymbol{\dagger[R,T]}}{[\langle R;\dagger R\rangle,\vartheta T]}}{[\langle R,\dagger R\rangle,\langle \boldsymbol{T;\vartheta T}\rangle]}}{\langle \boldsymbol{[R,T];[\dagger R,\vartheta T]}\rangle}}{\langle [R,T];\boldsymbol{\dagger[R,T]}\rangle}$$

**Example 2.8.** Negating the premise of the derivation from example 2.7, and then use the equivalence relation defined in Figure 1 to push negation to the level of atoms, we get the following:

$$\overline{\dagger[R,T]} = \vartheta\overline{[R,T]} = \vartheta(\overline{R},\overline{T})$$

Using Proposition 2.6, we can hence guarantee that it is possible to contruct the following proof:

$$\dfrac{\Gamma\Vdash}{(\vartheta(\overline{R},\overline{T}),[\langle R;\dagger R\rangle,\vartheta T])}$$

# 3 Coherence Space Semantics

We give denotational semantics to system sWIP using a coherence space model. Broadly speaking, the aim of this model of the system is to give a representation of proofs of some structure $A$ as *cliques* of the corresponding coherence space $[\![A]\!]$.

One of the advantages of doing so is that it allows us to derive properties about the logical system by working with purely categorical constructs. Some desirable results may be difficult to prove using the logic of the system alone, but much simpler when we work with only the model.

**Remark.** We will give the semantics for sWIP in terms of only binary versions of the connectives copar, seq, and par. For example, instead of working with an arbitrary copar structure $(R_1, .., R_n)$, we will only consider the simpler case $(R, T)$, and instead rely on the associative property to inductively construct more complex structures.

## 3.1 Introduction to Coherence Spaces

The idea of coherence spaces was first proposed by Jean-Yves Girard as a model for linear logic [3], and has since been adapted by a range of researchers to model their own systems. Most notably, Uday Reddy extended linear logic with an operator representing one-way communication [12], and gave a coherence space semantics on which much of this work has been based.

We introduce the following definitions which will be used in defining our model. For a more structured, category theoretic introduction to coherence spaces, the reader is referred to Paul-André Melliès' lecture notes [11].

**Definition 3.1.** A *coherence space* is a pair $(|A|, \frown_A)$, where $|A|$ is some underlying set and $\frown_A$ is the *coherence relation* defined on that set. The relation is symmetric, reflexive, and transitive. We also define a strictly irreflexive version $\smile_A$ which we call *strict coherence*, such that $a \smile_A a' \iff a \frown_A a'$ and $a \neq a'$

**Definition 3.2.** We say $a$ and $a'$ are *incoherent*, written $a \asymp_A a'$, if either $a \not\smile_A a'$ or $a = a'$

**Remark.** We will often refer to the coherence space $(|A|, \frown_A)$ as simply $A$ when appropriate. Similarly, we will often write $\frown$ or $\smile$ without the subscript when the coherence space is obvious from the context.

**Definition 3.3.** A *clique* of a coherence space $A$ is some $C \subseteq |A|$, whose elements are all pairwise coherent.

**Remark.** Note that as the clique is a set, the pairwise coherence is necessarily strict.

**Definition 3.4.** A *linear map* between coherence spaces $A$ and $B$ is some relation $f$ on $|A|$ and $|B|$, such that if $f$ relates $a$ to $b$, and also relates $a'$ to $b'$, we have that

$$a \frown_A a' \implies b \frown_B b'$$
$$a \smile_A a' \implies b \smile_B b'$$

We write $f : A \multimap B$, and use the notation $(\!|a, b|\!) \in f$ to mean "$f$ relates $a$ to $b$".

**Definition 3.5.** For linear maps $f : A \multimap B$ and $g : B \multimap C$, we define the *composition* $g \circ f : A \multimap C$ by:

$$g \circ f = \{(\!|a, c|\!) : \exists b \in |B| \text{ such that } (\!|a, b|\!) \in f, (\!|b, c|\!) \in g\}$$

Armed with these definitions, we can begin to construct a semantic model of the system sWIP.

### 3.2 A Model for Structures of sWIP

For each structure $x$ in the grammar of sWIP, we would like to define some semantic model $[\![x]\!]$ to represent it. Thus, we can inductively define the semantics of an arbitrarily complex structure, starting by translating the atoms and then working outwards towards the outermost connective. Here we will state the semantics that we would like for each structure, then, throughout the rest of this section, unpack and justify each definition.

**Remark.** Throughout this section, for any coherence space $A$ we will only define $\supset_A$. Deriving $\frown_A$ and $\asymp_A$ is a simple exercise in applying the definitions of a coherence space, and thus is left to the reader.

- **Unit**: $[\![\circ]\!] = (\{*\}, \{(*,*)\})$, a trivial coherence space whose underlying set is a singleton, with that single element related to itself. Notice that $* \not\frown *$ as the relation is strictly irreflexive, so $\frown_{[\![\circ]\!]} = \emptyset$.
- **Atom**: For each atom $a, b, .., z$ in a structure, we are able to choose any coherence space $A, B, .., Z$ to be the model $[\![a]\!], [\![b]\!], .., [\![z]\!]$. For example, an atom $b$ representing the base type of booleans may have the semantics $[\![b]\!] = (\{\top, \bot\}, \{(\top, \top), (\bot, \bot)\})$.
- **Dual**: $[\![\overline{R}]\!] = (|[\![R]\!]|, \asymp_{[\![R]\!]})$.
- **Copar**: $[\![(R,T)]\!] = ([\![R]\!] \times [\![T]\!], \supset_{[\![(R,T)]\!]})$, such that:

$$(r_1, t_1) \supset (r_2, t_2) \iff r_1 \supset_{[\![R]\!]} r_2, t_1 \supset_{[\![T]\!]} t_2$$

- **Seq**: $[\![\langle R;T\rangle]\!] = ([\![R]\!] \times [\![T]\!], \supset_{[\![\langle R;T\rangle]\!]})$, such that:

$$(r_1, t_1) \supset (r_2, t_2) \iff r_1 \frown_{[\![R]\!]} r_2 \text{ or } r_1 = r_2, t_1 \supset_{[\![T]\!]} t_2$$

- **Par**: $[\![[R,T]]\!] = ([\![R]\!] \times [\![T]\!], \supset_{[\![[R,T]]\!]})$, such that:

$$(r_1, t_1) \supset (r_2, t_2) \iff r_1 \frown_{[\![R]\!]} r_2 \text{ or } t_1 \frown_{[\![T]\!]} t_2 \text{ or } (r_1, t_1) = (r_2, t_2)$$

- **Dagger**: $[\![\dagger R]\!] = ([\![R]\!]^*, \supset_{[\![\dagger R]\!]})$, such that:

$$r_1 ... r_m \supset r'_1 ... r'_n \iff \text{one of:}$$

  - $\exists 1 \leq l \leq \min(m,n)$ s.t. $r_i = r'_i \ \forall i < l$, and $r_l \frown_{[\![R]\!]} r'_l$
  - $r_i = r'_i \ \forall i \leq \min(m,n)$

- **Hash**: $[\![\vartheta R]\!] = ([\![R]\!]^*, \supset_{[\![\vartheta R]\!]})$, such that:

$$r_1 ... r_m \supset r'_1 ... r'_n \iff \text{one of:}$$

  - $\exists 1 \leq l \leq \min(m,n)$ s.t. $r_i = r'_i \ \forall i < l$, and $r_l \frown_{[\![R]\!]} r'_l$
  - $r_1 ... r_m = r'_1 ... r'_n$

### 3.3 Connectives are Functors

We will now take a step back from modelling the system sWIP, and view the coherence space model from a purely categorical perspective. For the rest of this section, we abstract out the underlying structure of any given coherence space, and instead consider how rewrite rules and connectives behave when applied to arbitrary ones.

We require that connectives in our model behave nicely when working in the category of coherence spaces. We will briefly define this category, and then show that we can model each of the unary and binary connectives as functors.

Modelling connectives as functors allows rewrite rules to be applied within arbitrary contexts, by composing the rule's model with the functor representing the context. This abstraction simplifies deep inference by making rule application uniform and modular, ensuring necessary properties are preserved at all levels of nesting. Contexts can be constructed compositionally, providing a scalable and elegant framework for handling rules in arbitrary settings.

**Definition 3.6.** The *category of coherence spaces* is written $\mathsf{COHS}$, with coherence spaces as objects and linear maps between them as morphisms.

In order to prove that the some functor $F : \mathsf{COHS} \to \mathsf{COHS}$ preserves coherence of some map $f : A \multimap B$, we will show 2 things:

1. For $a \frown_{FA} a'$, and pairs $(\!|a, b|\!), (\!|a', b'|\!) \in Ff$, we have that $b \frown_{FB} b'$
2. For pairs $(\!|a, b|\!), (\!|a, b'|\!) \in Ff$, we have that $b \supset_{FB} b'$

(1) directly shows preservation of strict coherence. As strict coherence is a stronger condition than coherence, it also shows preservation of coherence in all cases aside from that where $a = a'$; (2) verifies this case directly.

#### 3.3.1 Copar

**Definition 3.7.** Let $(\_, \_) : \mathsf{COHS} \times \mathsf{COHS} \to \mathsf{COHS}$ be defined as follows:

- For $R, T \in Ob(\mathsf{COHS})$, $(R, T) \coloneqq (R \times T, \supset_{(R,T)})$ such that:
$$(r, t) \supset_{(R,T)} (r', t') \iff r \supset_R r' \text{ and } t \supset_T t'$$
- For $f : R \multimap T$ and $g : U \multimap V$, $(f, g) : (R, U) \multimap (T, V)$ is defined as follows:
$$(f, g) = \{(\!|(r, u), (t, v)|\!) : (\!|r, t|\!) \in f, (\!|u, v|\!) \in g\}$$

**Lemma 3.8.** $(\_, \_)$ *is a functor.*
*Proof.* $(\_, \_)$ preserves identity:

$$\begin{aligned}(id_R, id_T) &= \{(\!|(r, t), (r, t)|\!) : (\!|r, r|\!) \in id_R, (\!|t, t|\!) \in id_T\} \\ &= \{(\!|(r, t), (r, t)|\!) : r \in |R|, t \in |T|\} \\ &= \{(\!|(r, t), (r, t)|\!) : (r, t) \in |(R, T)|\} \\ &= id_{(R,T)}\end{aligned}$$

$(\_, \_)$ preserves composition:

Define 4 linear maps $f : R \multimap T$, $h : T \multimap U$, $g : V \multimap W$, and $k : W \multimap X$. Then we have $(f, g) : (R, V) \multimap (T, W), (h, k) : (T, W) \multimap (U, X)$, and:

$$\begin{aligned}(h,k) \circ (f,g) &= \{(\!|(r,v)(u,x)|\!) : \exists (t,w) \in |(T,W)| \text{ such that} \\ &\quad (\!|(r,v),(t,w)|\!) \in (f,g), (\!|(t,w),(u,x)|\!) \in (h,k)\} \\ &= \{(\!|(r,v)(u,x)|\!) : \exists t \in |T|, w \in |W| \text{ such that} \\ &\quad (\!|r,t|\!) \in f, (\!|v,w|\!) \in g, (\!|t,u|\!) \in h, (\!|w,x|\!) \in k\} \\ &= \{(\!|(r,v),(u,x)|\!) : (\!|r,u|\!) \in h \circ f, (\!|v,x|\!) \in k \circ g\} \\ &= (h \circ f, k \circ g)\end{aligned}$$

$(\_, \_)$ preserves coherence:

Define $f : R \multimap T$, $g : U \multimap V$.

First assume that $(r, u) \frown_{(R,U)} (r', u')$, and consider pairs $(\!|(r,u),(t,v)|\!)$, $(\!|(r',u'),(t',v')|\!) \in (f, g)$. Without loss of generality, assume that $r \frown_R r'$ and $u \subset\!\!\cdot_U u'$. Linearity of $f$ and $g$ implies that $t \frown_T t'$ and $v \subset\!\!\cdot_V v'$. This gives us $(t, v) \subset\!\!\cdot_{(T,V)} (t', v')$, and as $t \neq t'$ we find that the coherence is strict as required.

Now consider pairs $(\!|(r,u),(t,v)|\!), (\!|(r,u),(t',v')|\!) \in (f, g)$. Linearity of $f$ and $g$ implies that $t \subset\!\!\cdot_T t'$ and $v \subset\!\!\cdot_V v'$. This gives us $(t, v) \subset\!\!\cdot_{(T,V)} (t', v')$ as required.

□

### 3.3.2 Seq

**Definition 3.9.** Let $\langle \_; \_ \rangle : \mathsf{COHS} \times \mathsf{COHS} \to \mathsf{COHS}$ be defined as follows:

- For $R, T \in Ob(\mathsf{COHS})$, $\langle R; T \rangle \coloneqq (|R| \times |T|, \subset\!\!\cdot_{\langle R;T \rangle})$ such that:

$$(r, t) \subset\!\!\cdot_{\langle R;T \rangle} (r', t') \iff r \frown_R r' \text{ or } (r = r' \text{ and } t \subset\!\!\cdot_T t')$$

- For $f : R \multimap T$ and $g : U \multimap V$, $\langle f; g \rangle : \langle R; U \rangle \multimap \langle T; V \rangle$ is defined as follows:

$$\langle f; g \rangle = \{(\!|(r,u),(t,v)|\!) : (\!|r,t|\!) \in f, (\!|u,v|\!) \in g\}$$

**Lemma 3.10.** *$\langle \_; \_ \rangle$ is a functor.*

*Proof.* $\langle \_; \_ \rangle$ preserves identity and composition; the argument is the same as that from the proof of Lemma 3.8, with any coherence space $(R, T)$ replaced by $\langle R; T \rangle$.

$\langle \_; \_ \rangle$ preserves coherence:

Define $f : R \multimap T$, $g : U \multimap V$.

First assume that $(r, u) \frown_{\langle R;U \rangle} (r', u')$, and consider pairs $(\!|(r,u),(t,v)|\!)$, $(\!|(r',u'),(t',v')|\!) \in \langle f; g \rangle$. We have that either $r \frown_R r'$, or that $r = r'$ and $u \frown_U u'$. As $f$ and $g$ are linear maps, the former implies that $t \frown t'$, while the latter implies that $t \subset\!\!\cdot t'$ and $v \frown v'$. In

either case we get that $(t, v) \supset_{\langle T;V \rangle} (t', v')$, and as either $t \neq t'$ or $v \neq v'$ we find that the coherence is strict as required.

Now consider pairs $(\!|(r, u), (t, v)|\!), (\!|(r, u), (t', v')|\!) \in \langle f; g \rangle$. Linearity of $f$ implies that $t \supset_T t'$. We have that either $t \neq t'$, in which case $t \frown_T t'$, or that $t = t'$. Combining the second case with the linearity of $g$ gives that $t = t'$ and $v \supset_V v'$, so either case gives us $(t, v) \supset_{(T,V)} (t', v')$ as required.

□

### 3.3.3 Dual

**Definition 3.11.** Let $\overline{\{\ \ \}} : \mathsf{COHS}^{op} \to \mathsf{COHS}$ be defined as follows:

- For $R \in Ob(\mathsf{COHS})$, $\overline{R} = (|R|, \asymp_R)$.
- For $f : R \multimap T$, $\overline{f} : \overline{T} \multimap \overline{R}$ is defined as follows:

$$\overline{f} = \{(\!|t, r|\!) : (\!|r, t|\!) \in f\}$$

**Lemma 3.12.** $\overline{\{\ \ \}}$ *is a functor*

*Proof.* $\overline{\{\ \ \}}$ preserves identity trivially, as the underlying set of $\overline{R}$ is the same as that of $R$:

$\overline{\{\ \ \}}$ preserves composition:

Define 2 linear maps $f : R \multimap T$, $g : T \multimap U$. Then:

$$\begin{aligned}
\overline{f} \circ \overline{g} &= \{(\!|u, r|\!) : \exists t \text{ such that } (\!|u, t|\!) \in \overline{g}, (\!|t, r|\!) \in \overline{f}\} \\
&= \{(\!|u, r|\!) : \exists t \text{ such that } (\!|r, t|\!) \in f, (\!|t, u|\!) \in g\} \\
&= \{(\!|u, r|\!) : (\!|r, u|\!) \in g \circ f\} \\
&= \overline{g \circ f}
\end{aligned}$$

$\overline{\{\ \ \}}$ preserves coherence:

Define $f : R \multimap T$

Take $t, t' \in |T|$ such that $t \frown_{\overline{T}} t'$, that is, $t \not\frown_T t'$. Consider pairs $(\!|t, r|\!), (\!|t', r'|\!) \in \overline{f}$, so we have pairs $(\!|r, t|\!), (\!|r', t'|\!) \in f$. As $t \not\frown_T t'$, the contrapositive to the linearity of $f$ gives that $r \not\frown_R r'$, and thus $r \frown_{\overline{R}} r'$ as required.

If we instead consider pairs $(\!|t, r|\!), (\!|t, r'|\!) \in \overline{f}$, we have pairs $(\!|r, t|\!), (\!|r', t|\!) \in f$. Then $t = t' \implies t \asymp_T t'$, and again the contrapositive of linearity of $f$ gives $r \asymp_R r'$, thus $r \supset_{\overline{R}} r'$ as required.

□

### 3.3.4 Dagger

**Definition 3.13.** Let $\dagger_{\_} : \mathsf{COHS} \to \mathsf{COHS}$ be defined as follows:

- For $R \in Ob(\mathsf{COHS})$, $\dagger R = (|R|^*, \frown_{\dagger R})$, such that:

$$r_1...r_m \frown r'_1...r'_n \iff \text{one of:}$$

  - $\exists 1 \leq l \leq \min(m,n)$ s.t. $r_i = r'_i \ \forall i < l$, and $r_l \frown_R r'_l$
  - $r_i = r'_i \ \forall i \leq \min(m,n)$

  (Informally, 2 words over $R$ are coherent in $\dagger R$ if one is a prefix of the other, or if the first place they differ they do so strictly coherently)

- For $f : R \multimap T$, $\dagger f : \dagger R \multimap \dagger T$ is defined as follows:

$$\dagger f = \{(\!|r_1...r_n, t_1...t_n|\!) : (\!|r_i, t_i|\!) \in f \ \ \forall 1 \leq i \leq n\}$$

**Lemma 3.14.** $\dagger\_$ *is a functor*

*Proof.* $\dagger\_$ preserves identity:

$$\begin{aligned}\dagger id_R &= \{(\!|r_1...r_n, r_1...r_n|\!) : (\!|r_i, r_i|\!) \in id_R \ \ \forall 1 \leq i \leq n\} \\ &= \{(\!|r_1...r_n, r_1...r_n|\!) : r_i \in |R| \ \ \forall 1 \leq i \leq n\} \\ &= id_{\dagger R}\end{aligned}$$

$\dagger\_$ preserves composition:

Define 2 linear maps $f : R \multimap T$, $g : T \multimap U$. Then:

$$\begin{aligned}\dagger g \circ \dagger f &= \{(\!|r_1...r_n, u_1u_2...u_n|\!) : \exists t_1...t_n \in |T|^* \text{ such that} \\ &\qquad (\!|r_1...r_n, t_1...t_n|\!) \in \dagger f, (\!|t_1...t_n, u_1u_2...u_n|\!) \in \dagger g\} \\ &= \{(\!|r_1...r_n, u_1u_2...u_n|\!) : \exists t_i \in |T| \text{ such that} \\ &\qquad (\!|r_i, t_i|\!) \in f, (\!|t_i, u_i|\!) \in g \ \ \forall 1 \leq i \leq n\} \\ &= \{(\!|r_1...r_n, u_1u_2...u_n|\!) : (\!|r_i, u_i|\!) \in g \circ f \ \ \forall 1 \leq i \leq n\} \\ &= \dagger(g \circ f)\end{aligned}$$

$\dagger\_$ preserves coherence:

Define $f : R \multimap T$.

First assume that $r_1...r_n \frown_{\dagger R} r'_1...r'_n$, and consider pairs $(\!|r_1...r_n, t_1...t_n|\!)$, $(\!|r'_1...r'_n, t'_1...t'_n|\!) \in \dagger f$. $\exists 1 \leq j \leq n$ where $j$ is the smallest index such that $r_j \frown_R r'_j$. Linearity of $f$ then gives that $t_i \frown_T t'_i \ \forall 1 \leq i \leq j$, so in the first position that they differ they must do so coherently (even if all of $t_i = t'_i$, linearity of $f$ ensures that $t_j \frown_T t'_j$). This gives $t_1...t_n \frown_{\dagger T} t'_1...t'_n$ as required.

Now consider pairs $(\!|r_1...r_n, t_1...t_n|\!)$, $(\!|r_1...r_n, t'_1...t'_n|\!) \in \dagger f$. Linearity of $f$ gives that $t_i \frown_T t'_i \ \forall 1 \leq i \leq n$. If $t_1...t_n = t'_1...t'_n$ then the proof is trivial, so assume they are not equal. $\exists 1 \leq j \leq n$ where $j$ is the smallest index such that $t_j \neq t'_j$. However, as $t_j \frown_T t'_j$, we must have that $t_j \frown_T t'_j$, giving $t_1...t_n \frown_{\dagger T} t'_1...t'_n$ as required.

□

### 3.3.5 Par and Hash

The remaining connectives, Par and Hash, can be constructed compositionally, as they are the duals of Copar and Dagger respectively. That is, we can define:

$$[\_,\_] \coloneqq \overline{(\overline{\{\quad\}}, \overline{\{\quad\}})} : \mathsf{COHS} \times \mathsf{COHS} \to \mathsf{COHS}$$

$$\vartheta\_ \coloneqq \overline{\dagger\overline{\{\quad\}}} : \mathsf{COHS} \to \mathsf{COHS}$$

Deriving the actions of these functors is left as an exercise to the reader; the results are as follows:

- For $R, T \in Ob(\mathsf{COHS})$, $[R,T] \coloneqq (R \times T, \frown_{[R,T]})$ such that:

$$(r,t) \frown_{[R,T]} (r',t') \iff r \frown_R r' \text{ or } t \frown_T t' \text{ or } (r,t) = (r',t')$$

- For $f : R \multimap T$ and $g : U \multimap V$, $[f,g] : [R,U] \multimap [T,V]$ is defined as follows:

$$[f,g] = \{(\!|(r,u),(t,v)|\!) : (\!|r,t|\!) \in f, (\!|u,v|\!) \in g\}$$

- For $R \in Ob(\mathsf{COHS})$, $\vartheta R = (|R|^*, \frown_{\vartheta R})$, such that:

$$r_1 ... r_m \frown r'_1 ... r'_n \iff \text{one of:}$$

  - $\exists 1 \leq l \leq \min(m,n)$ s.t. $r_i = r'_i \ \forall i < l$, and $r_l \frown_R r'_l$
  - $r_1 ... r_m = r'_1 ... r'_n$

  (Informally, 2 words over $R$ are coherent in $\vartheta R$ if they are equal, or if the first place they differ they do so strictly coherently)

- For $f : R \multimap T$, $\vartheta f : \vartheta R \multimap \vartheta T$ is defined as follows:

$$\vartheta f = \{(\!|r_1 ... r_n, t_1 ... t_n|\!) : (\!|r_i, t_i|\!) \in f \ \forall 1 \leq i \leq n\}$$

We do not have to prove that these are functors, as composition of functors always results in functors.

With these final 2 functors, we now have a purely categorial model of any structure that could arise from the grammar of sWIP, which preserves all necessary properties no matter the choice of coherence space for atoms.

## 3.4 Rewrite Rules are Linear Maps

For each rewrite rule of sWIP, we define a family of linear maps between coherence spaces to be its model. The map representing some rewrite rule $\rho \dfrac{R}{T}$ will be of the form $\rho_R : R \multimap T$; we will use the notation $\rho\_$ to denote the family of linear maps representing the polymorphic map $\rho$, applicable to arbitrary structures.

We will consider the simplest form of each rule, that is, applying each in an empty context. Extending this to rewrites in arbitrary contexts is automatic, as we can represent the context as a functor.

Suppose we have a rule $\rho$ as above, and would like to apply it inside of a context

$$S = \dagger\langle\{\quad\}; \overline{(U, V)}\rangle$$

to perform the following rewrite:

$$\rho \frac{\dagger\langle R; \overline{(U, V)}\rangle}{\dagger\langle T; \overline{(U, V)}\rangle}$$

We can compose the functors $\dagger_-, \langle_-; _-\rangle, \overline{\{\quad\}}$, and $(_-, _-)$ to create the functor

$$\dagger\langle_-; \overline{(_-, _-)}\rangle : \mathsf{COHS} \times \mathsf{COHS}^{op} \times \mathsf{COHS}^{op} \to \mathsf{COHS}$$

which, when $U$ and $V$ are put into the second and third slots, gives us a representation of a structure with a single hole:

$$\dagger\langle_-; \overline{(U, V)}\rangle : \mathsf{COHS} \to \mathsf{COHS}$$

Finally, we can apply this to $\rho_R$ to get the linear map representing the application of $\rho$ to $R$ in the context $S$.

We will later prove that the same result is obtained whether we first apply the context functor to the linear map, and then apply this new map to the structure representation, or instead apply the rewrite rule to the structure representation and then the context functor to this result. Crucially, we must first show that each of these maps preserve coherence (and thus cliques), which will allow chains of rewrites to model derivations (and thus proofs).

### 3.4.1 $\mathsf{ai}{\downarrow}$

**Definition 3.15.** To model the rewrite

$$\mathsf{ai}{\downarrow} \frac{\circ}{[a, \overline{a}]}$$

we define $\mathsf{ai}{\downarrow}_a : \circ \multimap [a, \overline{a}]$ as follows:

$$\mathsf{ai}{\downarrow}_a = \{(\!|*, (\alpha, \alpha)|\!) : \alpha \in |a|\}$$

**Lemma 3.16.** $\mathsf{ai}{\downarrow}_a$ *is a linear map*

*Proof.* As $* \not\frown_\circ *$, preservation of strict coherence is vacuously true. Now consider pairs $(\!|*, (\alpha, \alpha)|\!), (\!|*, (\beta, \beta)|\!) \in \mathsf{ai}{\downarrow}_\circ$. As $* = *$ and thus $* \supset\!\subset *$, we must show that $(\alpha, \alpha) \supset\!\subset_{[a,\overline{a}]} (\beta, \beta)$. If $\alpha = \beta$ the coherence is trivial, so suppose that $\alpha \neq \beta$. We either have that $\alpha \frown_a \beta$, in which case we get $(\alpha, \alpha) \supset\!\subset_{[a,\overline{a}]} (\beta, \beta)$, or that $\alpha \not\frown_a \beta$. In the latter case, we get that $\alpha \frown_{\overline{a}} \beta$, and so $(\alpha, \alpha) \supset\!\subset_{[a,\overline{a}]} (\beta, \beta)$. □

### 3.4.2 $\mathsf{w}\downarrow$

**Definition 3.17.** To model the rewrite

$$\mathsf{w}\downarrow \frac{\circ}{\vartheta R}$$

we define $\mathsf{w}\downarrow_R : \circ \multimap \vartheta R$ as follows:

$$\mathsf{w}\downarrow_R = \{(\!|*, \epsilon|\!)\}$$

where $\epsilon$ is the empty sequence in $R$

**Lemma 3.18.** $\mathsf{w}\downarrow_R$ *is a linear map*

*Proof.* Trivial, as any two empty sequences in $R$ are coherent by equality. □

### 3.4.3 $\mathsf{d}\downarrow$

**Definition 3.19.** To model the rewrite

$$\mathsf{d}\downarrow \frac{\circ}{\dagger\circ}$$

we define $\mathsf{d}\downarrow_\circ : \circ \multimap \dagger\circ$ as follows:

$$\mathsf{d}\downarrow_\circ = \{(\!|*, *^n|\!) : n \in \mathbb{N}\}$$

**Lemma 3.20.** $\mathsf{d}\downarrow_\circ$ *is a linear map*

*Proof.* Trivial, as $*^n$ is a prefix of $*$ for $n \in \{0, 1\}$, while $*$ is a prefix of $*^n$ for any $n \geq 2$. □

### 3.4.4 $\mathsf{s}$

**Definition 3.21.** To model the rewrite

$$\mathsf{s}\frac{([R, T], U)}{[(R, U), T]}$$

we define $\mathsf{s}_{R,T,U} : ([R, T], U) \multimap [(R, U), T]$ as follows:

$$\mathsf{s}_{R,T,U} = \{(\!|(r, t, u), (r, u, t)|\!) : r \in |R|, t \in |T|, u \in |U|\}$$

**Lemma 3.22.** $\mathsf{s}_{R,T,U}$ *is a linear map*

*Proof.* Consider $(r, t, u), (r', t', u') \in |([R, T], U)|$.

If the two are equal, it is trivial to show that $\mathsf{s}_{R,T,U}$ preserves this equality. Assume instead that $(r, t, u) \frown (r', t', u')$. To get the required coherence $(r, u, t) \frown_{[(R,U),T]} (r', u', t')$, it suffices to show that either $(r, u) \frown_{(R,U)} (r', u')$ or $t \frown t'$. We have 2 possible cases:

Case 1: $(r, t) \frown_{[R,T]} (r', t')$ and $u \supset u'$

> $(r, t) \frown (r', t') \implies r \frown r'$ or $t \frown t'$. The latter alone suffices, and the former combined with $u \supset u'$ gives $(r, u) \frown_{(R,U)} (r', u')$ as required.

Case 2: $(r,t) \supset_{[R,T]} (r',t')$ and $u \frown u'$

The same 2 cases as above can arise here, or alternatively we may have that $(r,t) = (r',t')$. In this case, we have that $r \supset r'$, which combined with $u \frown u'$ gives $(r,u) \frown_{(R,U)} (r',u')$ as required.

□

### 3.4.5 $\mathsf{q}\downarrow$

**Definition 3.23.** To model the rewrite

$$\mathsf{q}\downarrow \frac{\langle [R,T]; [U,V] \rangle}{[\langle R;U \rangle, \langle T;V \rangle]}$$

we define $\mathsf{q}\downarrow_{R,T,U,V} : \langle [R,T]; [U,V] \rangle \multimap [\langle R;U \rangle, \langle T;V \rangle]$ as follows:

$$\mathsf{q}\downarrow_{R,T,U,V} = \{(\!|(r,t,u,v),(r,u,t,v)|\!) : r \in |R|, t \in |T|, u \in |U|, v \in |V|\}$$

**Lemma 3.24.** $\mathsf{q}\downarrow_{R,T,U,V}$ *is a linear map*

*Proof.* Consider $(r,t,u,v),(r',t',u',v') \in |\langle [R,T]; [U,V] \rangle|$.

If the two are equal, it is trivial to show that $\mathsf{q}\downarrow_{R,T,U,V}$ preserves this equality. Assume instead that $(r,t,u,v) \frown (r',t',u',v')$. To get the required coherence $(r,u,t,v) \frown_{[\langle R;U \rangle, \langle T;V \rangle]} (r',u',t',v')$, it suffices to show that either $(r,u) \frown_{\langle R;U \rangle} (r',u')$ or $(t,v) \frown_{\langle T;V \rangle} (t',v')$. We have 2 possible cases:

Case 1: $(r,t) \frown_{[R,T]} (r',t')$

We have that either $r \frown r'$, implying $(r,u) \frown (r',u')$, or $t \frown t'$, in which case $(t,v) \frown (t',v')$. In either case the sufficient condition is met.

Case 2: $(r,t) = (r',t')$ and $(u,v) \frown_{[U,V]} (u',v')$

We have that either $u \frown u'$ or $v \frown v'$. As $r = r'$, the former implies $(r,u) \frown (r',u')$, and similarly the latter implies $(t,v) \frown (t',v')$. In either case the sufficient condition is met.

□

### 3.4.6 $\mathsf{p}\downarrow$

**Definition 3.25.** To model the rewrite

$$\mathsf{p}\downarrow \frac{\dagger[R,T]}{[\dagger R, \vartheta T]}$$

we define $\mathsf{p}\downarrow_{R,T} : \dagger[R,T] \multimap [\dagger R, \vartheta T]$ as follows:

$$\mathsf{p}\downarrow_{R,T} = \{(\!|(r_1,t_1)...(r_n,t_n),(r_1...r_n,t_1...t_n)|\!) : r_i \in |R|, t_j \in |T|\}$$

**Lemma 3.26.** $\mathsf{p}\downarrow_{R,T}$ *is a linear map*

*Proof.* Consider $(r_1, t_1)...(r_m, t_m), (r'_1, t'_1)...(r'_n, t'_n) \in |\dagger[R, T]|$

If the two are equal, it is trivial to show that $\mathsf{p}{\downarrow}_{R,T}$ preserves this equality. Assume instead that $(r_1, t_1)...(r_m, t_m) \frown_{\dagger[R,T]} (r'_1, t'_1)...(r'_n, t'_n)$, considering pairs $(\!|(r_1, t_1)...(r_m, t_m), (r_1...r_m, t_1...t_m)|\!)$, $(\!|(r'_1, t'_1)...(r'_n, t'_n), (r'_1...r'_n, t'_1...t'_n)|\!)$ . We have 2 possible cases:

Case 1: $\exists 1 \leq l \leq \min(m, n)$ s.t. $(r_i, t_i) = (r'_i, t_i)\ \forall i < l$, and $(r_l, t_l) \frown_{[R,T]} (r'_l, t'_l)$

> Assume WLOG that $r_l \frown_R r'_l$. We then have that $r_i = r'_i\ \forall 1 \leq i \leq l$, and $r_l \frown_R r'_l$. This implies that $r_1...r_m \frown_{\dagger R} r'_1...r'_n$, which then gives $(r_1...r_m, t_1...t_m) \frown_{[\dagger R, \vartheta T]} (r'_1...r'_n, t'_1...t'_n)$ as required.

Case 2: $(r_i, t_i) = (r'_i, t'_i)\ \forall 1 \leq i \leq \min(m, n)$

> We get that $r_i = r'_i\ \forall 1 \leq i \leq \min(m, n)$. As $(r_1, t_1)...(r_m, t_m) \frown_{\dagger[R,T]} (r'_1, t'_1)...(r'_n, t'_n)$, it follows that $m \neq n$. This implies that $r_1...r_m \frown_{\dagger R} r'_1...r'_n$, and so $(r_1...r_m, t_1...t_m) \frown_{[\dagger R, \vartheta T]} (r'_1...r'_n, t'_1...t'_n)$ as required.

□

### 3.4.7 $\mathsf{b}{\downarrow}$

**Definition 3.27.** To model the rewrite

$$\mathsf{b}{\downarrow}\ \frac{\langle R; \vartheta R\rangle}{\vartheta R}$$

we define $\mathsf{b}{\downarrow}_R : \langle R; \vartheta R\rangle \multimap \vartheta R$ as follows:

$$\mathsf{b}{\downarrow}_R = \{(\!|(r_0, r_1...r_n), r_0 r_1...r_n|\!) : r_i \in |R|\}$$

**Lemma 3.28.** $\mathsf{b}{\downarrow}_R$ *is a linear map*

*Proof.* Consider $(r_0, r_1...r_m), (r'_0, r'_1...r'_n) \in |\langle R; \vartheta R\rangle|$.

If the two are equal, it is trivial to show that $\mathsf{b}{\downarrow}_R$ preserves this equality. Assume instead that $(r_0, r_1...r_m) \frown_{\langle R;\vartheta R\rangle} (r'_0, r'_1...r'_n)$. We have that either $r_0 \frown_R r'_0$, or that $r_0 = r'_0$ and $r_1...r_m \frown_{\vartheta R} r'_1...r'_n$. In either case, $r_0 r_1...r_m \neq r'_0 r'_1...r'_n$, and the first place they differ they do so strictly coherently, giving $r_0 r_1...r_m \frown_{\vartheta R} r'_0 r'_1...r'_n$ as required. □

### 3.4.8 Up Rules

We can once again leverage the idea of duality to construct models the remaining rewrite rules. By applying the $\overline{\{\ \}}$ functor to the model for any "down" rule, we get the model for the corresponding "up" rule. Verifying that the resulting linear maps do behave as intended is left as an exercise to the reader.

# 4 Naturality of Rewrites

In order to correctly model derivations, we require a certain freedom regarding the order that we can apply rewrite rules and context functors. As mentioned previously, applying a context to a both sides of a rewrite should yield the same result as applying the context to the rewritten structure. This required property of rewrite rules, when considered as linear maps of coherence spaces, is naturality.

We first give formal definitions of the category theoretic concepts at play, namely natural and dinatural transformations. Then, we show a few key examples of rewrites behaving in the intended way, and classify each rewrite rule's map as one of these types. Finally, we use a key result proved by McCusker to show that we can compose arbitrarily many rewrites into a derivation and still enjoy the same properties.

## 4.1 Natural and Dinatural Transformations

**Definition 4.1.** Given functors $F, G : \mathbb{C} \to \mathbb{D}$, a *natural transformation* $\phi : F \to G$ is a family of morphisms $\phi_A : FA \to GA$, such that for any $f \in \mathrm{Hom}_{\mathbb{C}}(A, B)$ the following diagram commutes:

$$\begin{array}{ccc} FA & \xrightarrow{Ff} & FB \\ \downarrow{\scriptstyle \phi_A} & & \downarrow{\scriptstyle \phi_B} \\ GA & \xrightarrow{Gf} & GB \end{array}$$

It is a standard result that the composition of two natural transformations is itself a natural transformation [8]. We will thus use this fact without proof.

**Definition 4.2.** Given functors $F, G : \mathbb{C} \times \mathbb{C}^{op} \to \mathbb{D}$, a *dinatural transformation* $\psi : F \to G$ is a family of morphisms $\psi_A : F(A, A) \to G(A, A)$, such that for any $f \in \mathrm{Hom}_{\mathbb{C}}(A, B)$ the following diagram commutes:

$$\begin{array}{ccccc} & & F(A,A) & \xrightarrow{\psi_A} & G(A,A) & & \\ & \overset{F(1,f)}{\nearrow} & & & & \overset{G(f,1)}{\searrow} & \\ F(A,B) & & & & & & G(B,A) \\ & \underset{F(f,1)}{\searrow} & & & & \underset{G(1,f)}{\nearrow} & \\ & & F(B,B) & \xrightarrow{\psi_B} & G(B,B) & & \end{array}$$

where 1 is the identity morphism.

Unlike natural transformations, dinatural transformations do not compose in general [10].

## 4.2 Classifying Rewrite Rules

We briefly introduce some new terminology with which to classify rewrite rules of sWIP.

**Definition 4.3.** We call a rewrite rule $\rho \dfrac{R}{T}$ of sWIP a *(di)natural rewrite* if its model as a family of linear maps $\rho_{\_}$ is a (di)natural transformation.

We will classify all rewrite rules of $\mathsf{sWIP}$, but only give proofs for a select few, as they follow the same general structure. The following proposition also limits the amount of rules we need to classify.

**Proposition 4.4.** *An up-down pair of rewrite rules $\rho\uparrow, \rho\downarrow$ must both have the same classification.*

*Proof.* We use a commuting diagram to show the (di)naturality of any rule $\rho$. As an up-down pair are, by definition, dual to eachother, we can simply take duals and reverse the arrows in the proof of $\rho\uparrow$'s classification to get an equivalent one for $\rho\downarrow$. □

We find that all but $\mathsf{ai}\downarrow$ and $\mathsf{ai}\uparrow$ are natural rewrites. The intuition behind this is that they are the only ones which deal explicitly with negative structures, which correspond to the objects of $\mathsf{COHS}^{op}$ present in dinaural transformations. All other rewrites either shuffle substructures and exchange structure brackets, or duplicate/combine instances of the same substructure; in either case, the corresponding transformation is natural. We give proofs for 2 cases, noting that the others follow a similar format and so are left as an exercise to the reader.

**Lemma 4.5.** $\mathsf{s}$ *is a natural rewrite.*

*Proof.* Following the definitions, we see that the required $F$ and $G$ are

$$([\_,\_],\_),[(\_,\_),\_] : \mathsf{COHS} \times \mathsf{COHS} \times \mathsf{COHS} \to \mathsf{COHS}$$

We thus need to show that for any $g : R \multimap R'$, $h : T \multimap T'$, and $k : U \multimap U'$, the following diagram commutes:

$$\begin{array}{ccc} ([R,T],U) & \xrightarrow{([g,h],k)} & ([R',T'],U') \\ \downarrow{\scriptstyle \mathsf{s}_{R,T,U}} & & \downarrow{\scriptstyle \mathsf{s}_{R',T',U'}} \\ [(R,U),T] & \xrightarrow{[(g,k),h]} & [(R',U'),T'] \end{array}$$

Indeed,

$$\begin{aligned} \mathsf{s}_{R',T',U'} \circ ([g,h],k) &= \{(\!|(r,t,u),(r',u',t')|\!) : \\ &\quad (\!|(r,t,u),(r',t',u')|\!) \in ([g,h],k), (\!|(r',t',u'),(r',u',t')|\!) \in \mathsf{s}_{R',T',U'}\} \\ &= \{(\!|(r,t,u),(r',u',t')|\!) : (\!|r,r'|\!) \in g, (\!|t,t'|\!) \in h, (\!|u,u'|\!) \in k\} \\ &= \{(\!|(r,t,u),(r',u',t')|\!) : \\ &\quad (\!|(r,t,u),(r,u,t)|\!) \in \mathsf{s}_{R,T,U}, (\!|(r,u,t),(r',u',t')|\!) \in [(g,k),h]\} \\ &= [(g,k),h] \circ \mathsf{s}_{R,T,U} \end{aligned}$$

□

**Lemma 4.6.** $\mathsf{b}\uparrow$ *is a natural rewrite.*

*Proof.* Here, the definitions give us $F = \dagger\_, G = \langle \_ ; \dagger\_ \rangle$, and we are required to show that for all $g : R \multimap R'$ the following diagram must commute:

$$\begin{array}{ccc} \dagger R & \xrightarrow{\dagger g} & \dagger R' \\ \downarrow{\scriptstyle \mathsf{b}\uparrow_R} & & \downarrow{\scriptstyle \mathsf{b}\uparrow_{R'}} \\ \langle R ; \dagger R \rangle & \xrightarrow{\langle g ; \dagger g \rangle} & \langle R' ; \dagger R' \rangle \end{array}$$

Indeed,

$$\begin{aligned}
\mathsf{b}{\uparrow_{R'}} \circ \dagger g &= \{(\!|r_0...r_n, (r'_0, r'_1...r'_n)|\!) : \\
&\quad (\!|r_0...r_n, r'_0...r'_n|\!) \in \dagger g, (\!|r'_0...r'_n, (r'_0, r'_1...r'_n)|\!) \in \mathsf{b}{\uparrow_{R'}}\} \\
&= \{(\!|r_0...r_n, (r'_0, r'_1...r'_n)|\!) : (\!|r_i, r'_i|\!) \in g\} \\
&= \{(\!|r_0...r_n, (r'_0, r'_1...r'_n)|\!) : \\
&\quad (\!|r_0...r_n, (r_0, r_1...r_n)|\!) \in \mathsf{b}{\uparrow_R}, (\!|(r_0, r_1...r_n), (r'_0, r'_1...r'_n)|\!) \in \langle g; \dagger g\rangle\} \\
&= \langle g; \dagger g\rangle \circ \mathsf{b}{\uparrow_R}
\end{aligned}$$

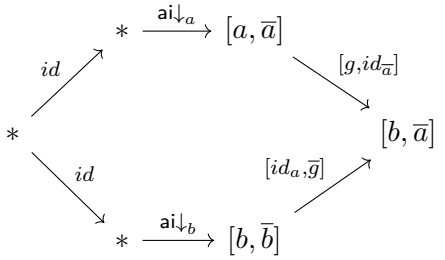
□

We then turn our attention to the dinatural rewrites. A proof is given for the dinaturality of $\mathsf{ai}{\downarrow}$. The corresponding proof for $\mathsf{ai}{\uparrow}$ can be easily derived, as stated previously, by taking duals and reversing arrows.

**Lemma 4.7.** $\mathsf{ai}{\downarrow}$ *is a dinatural rewrite.*
*Proof.* Following the defintions, we find that the required $F$ is the trivial functor $\circ : \mathsf{COHS} \times \mathsf{COHS}^{op} \to \mathsf{COHS}$. This sends any pair of coherence spaces to the trivial coherence space $*$, and all linear maps to the identity. Then $G$ is

$$[\_, \overline{\{\ \}}] : \mathsf{COHS} \times \mathsf{COHS}^{op} \to \mathsf{COHS}$$

For any $g : a \multimap b$, the required commutative diagram is




which can be simplified to the following:

$$\begin{array}{ccc}
* & \xrightarrow{\mathsf{ai}{\downarrow}_a} & [a, \overline{a}] \\
\downarrow{\scriptstyle \mathsf{ai}{\downarrow}_b} & & \downarrow{\scriptstyle [g, id_{\overline{a}}]} \\
[b, \overline{b}] & \xrightarrow{[id_b, \overline{g}]} & [b, \overline{a}]
\end{array}$$

Indeed,

$$\begin{aligned}
[g, id_{\overline{a}}] \circ \mathsf{ai}{\downarrow_a} &= \{(\!|*, (\beta, \alpha)|\!) : (\!|*, (\alpha, \alpha)|\!) \in \mathsf{ai}{\downarrow_a}, (\!|(\alpha, \alpha), (\beta, \alpha)|\!) \in [g, id_{\overline{a}}]\} \\
&= \{(\!|*, (\beta, \alpha)|\!) : (\!|\alpha, \beta|\!) \in g\} \\
&= \{(\!|*, (\beta, \alpha)|\!) : (\!|*, (\beta, \beta)|\!) \in \mathsf{ai}{\downarrow_b}, (\!|(\beta, \beta), (\beta, \alpha)|\!) \in [id_b, \overline{g}]\} \\
&= [id_b, \overline{g}] \circ \mathsf{ai}{\downarrow_b}
\end{aligned}$$

□

Now that all rewrite rules have been classified, we investigate how their naturality is preserved when they are composed to form derivations.

## 4.3 Preserving Dinaturality through Composition

### 4.3.1 Graphical Representation of Derivations

As we know that all derivations in sWIP can be modelled by composing a number of (di)natural transformations, we are able to represent them as simple graphs. Each argument of a functor is a box, and applications of the transformations are lines between them. In our case, white boxes represent objects of COHS and grey ones represent those of $\mathsf{COHS}^{op}$. These graphs are an extension of Kelly-Mac Lane graphs [9], and are introduced formally by McCusker [10].

**Definition 4.8.** We call a graph of this type *well-formed* if it is the representative graph of a well formed derivation in sWIP.

The example below shows the correspondance between a derivation and its representation as a graph:

$$\mathsf{b}\uparrow \dfrac{\mathsf{s}\dfrac{\mathsf{ai}\downarrow \dfrac{\dagger a}{([b,\bar{b}],\dagger a)}}{[(b,\dagger a),\bar{b}]}}{[(b,\langle a;\dagger a\rangle),\bar{b}]}$$

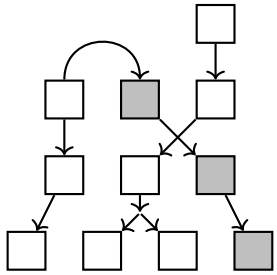


Figure 3: A derivation in sWIP and its graph

This representation abstracts away many of the details of the transformations themselves, and McCusker proved that it also leads to a simple sufficient condition for a composition of dinatural transformations to be dinatural: Any composition of dinatural transformations whose graph is acyclic is itself a dinatural transformation [10]. Note that at the top and bottom of any cycle is necessarily an arrow between a white and a grey box, pointing in opposite directions. THIS WORDING NEEDS TIDYING UP!!!!

Looking specifically at sWIP, in which the only two dinatural rewrites are ai↑ and ai↓, the only possible interactions between a white and a grey box are shown in Figure 4. We can deduce that a cycle could form if the conclusion of an ai↓ rewrite became the premise of an ai↑.

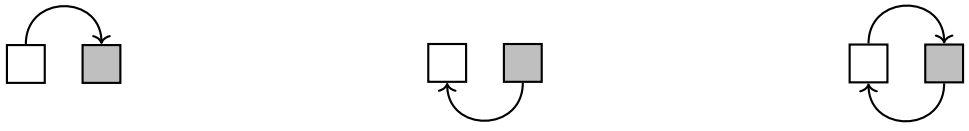


Figure 4: Graphs of ai↓ and ai↑ respectively, and how they could combine to form a simple cycle

This simple cycle can never occur as the corresponding structure brackets do not match up (ai↓ produces a copar conclusion, ai↑ requires a par premise, so the graph is not well-formed). Instead, a more complex cycle may form if we allow additional transformations inbetween the

applications of ai↑ and ai↓. Looking back at the graph in Figure 3, we could concievably add an instance of ai↑ at the bottom. This would introduce an additional arrow (dotted), forming a longer cycle (bolded), as shown in Figure 5.

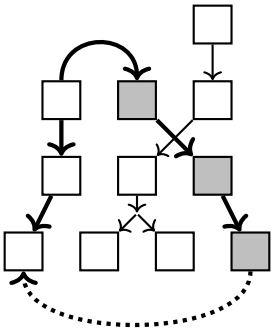


Figure 5: A modification of a graph with a cycle formed

We must verify that this new graph, or any constructed in a similar fashion, is not well-formed.

### 4.3.2 Derivations are Acyclic

We will consider the equivalent notion when working purely with the logic. That is, we must show that for any context $S$ the following derivation cannot be well-formed:

$$\mathsf{ai}{\downarrow}\,\frac{S\{\circ\}}{S[a,\overline{a}]}$$
$$\Gamma\,\|$$
$$\mathsf{ai}{\uparrow}\,\frac{S'(a,\overline{a})}{S'\{\circ\}}$$

Firstly, we show that we are not able to freely rewrite the copar substructure $[a,\overline{a}]$ into the par substructure $(a,\overline{a})$. By analysing the rewrite rules of the system, we can easily verify that this is the case.

**Proposition 4.9.** *For any structures R,T of* sWIP*, the following derivation cannot be formed:*

$$\frac{[R,T]}{(R,T)}\,\Delta\,\|$$

*Proof.* There are 5 rewrites (besides ai↑and ai↓) containing either a copar in the premise or a par in the conclusion. We now analyse each of them individually, deduce which structures can be freely transformed into others, and ultimately show that the copar to par rewrite is impossible in a well-formed structure. The main technique used is to set certain structures in each rule's definition to the unit, to see how the simplified version of the rule can be applied in more general situations.

- $\mathsf{s}\,\dfrac{([R,T],U)}{[(R,U),T]}$:

  By setting $R=\circ$ and applying the singleton laws, we are able to rewrite $(T,U)$ to $[T,U]$. s thus allows us to replace any par structure with a copar. Instead setting $T=\circ$ or $U=\circ$ and removing redundant brackets, the rule simply reduces to the identity.

- $\mathsf{q}{\downarrow}\dfrac{\langle [R,T];[U,V]\rangle}{[\langle R;U\rangle,\langle T;V\rangle]}$:

  Here we can set $T = U = \circ$, which once simplified allows a rewrite from $\langle R;V\rangle$ to $[R,V]$, or indeed any seq structure to a copar. Once again, this is the only application of the rule which allows us to directly replace a structure brackets, with all others either shuffling elements between substructures or reducing to the identity.

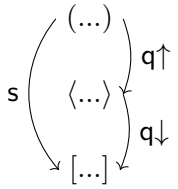
- $\mathsf{q}{\uparrow}\dfrac{(\langle R;T\rangle,\langle U;V\rangle)}{\langle (R,U);(T,V)\rangle}$

  This is the dual of $\mathsf{q}{\downarrow}$; it is unsurprising that by taking the dual of each structure and swapping premise with conclusion, we can rewrite a par structure into a seq. This result is easily verified by once again setting $T = U = \circ$.

- $\mathsf{p}{\uparrow}\dfrac{(\vartheta R,\dagger T)}{\vartheta(R,T)}$, $\mathsf{p}{\downarrow}\dfrac{\dagger[R,T]}{[\dagger R,\vartheta T]}$:

  While these do contain the structures of interest, they only introduce new unary connectives. As they do not exchange any structure brackets, they do not contribute to our search.

Of the 5 rewrites above, only 3 direct exchanges of structure brackets have been shown to be possible: $(...) \to [...]$, $\langle ...\rangle \to [...]$, and $(...) \to \langle ...\rangle$. This is better demonstrated visually as a heirarchy of structure brackets, in which we are free to move down the list but unable to climb back up:




This shows that the proposed rewrite is indeed impossible. □

For a full proof that we can freely chain rewrite rules and represent them as dinatural transformations, we must consider how the context $S\{\}$ may interact with the substructure. We must give particular care to the case that context contains exponentials, and verify that duplication of structures does not lead to cycles. In certain similar logic systems with notions of duplication, this is a point of failure for such a representation.

**Example 4.10.** In SELS [14], a simpler system than sWIP with no notion of sequencing, structures duplicated with the *of course* structure $!R$ are placed inside of a par structure by the $\mathsf{b}{\uparrow}$ rule:

$$\mathsf{b}{\uparrow}\frac{S\{!R\}}{S(R,!R)}$$

This allows us to construct the following derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{!{[a,\overline{a}]}}{([a,\overline{a}],!{[a,\overline{a}]})}\,\mathsf{b}\uparrow}{([a,\overline{a}],[a,\overline{a}],!{[a,\overline{a}]})}\,\mathsf{b}\uparrow}{([a,\overline{a}],[a,\overline{a}])}\,\mathsf{w}\uparrow}{[(a,[a,\overline{a}]),\overline{a}]}\,\mathsf{s}}{[(a,\overline{a}),a,\overline{a}]}\,\mathsf{s}$$

That is, we have a derivation from $S[a,\overline{a}]$ to $S'(a,\overline{a})$, where $S = !\{\quad\}$ and $S' = [\{\quad\},a,\overline{a}]$.

WIP:

**Definition 4.11.** For any application of a rewrite rule in sWIP, atoms in the conclusion are *connected* in any of the following cases:

- They are some pair $[a,\overline{a}]$ introduced by the rule ai$\downarrow$
-

- Conclude that proofs are dinatural as they are compositions of naturals and dinaturals with no cycles - Atomic flows????

# 5 Conclusion

# References


[1] Paola Bruscoli. A purely logical account of sequentiality in proof search. In *International Conference on Logic Programming*, pages 302–316. Springer, 2002.

[2] Husna Farooqui. The curry-howard correspondence. 2021.

[3] Jean-Yves Girard. Linear logic. *Theoretical computer science*, 50(1):1–101, 1987.

[4] Alessio Guglielmi. Deep inference. In *All About Proofs, Proofs for All*. College Publications, 2015.

[5] Alessio Guglielmi and Lutz Straßburger. A non-commutative extension of mell. In *Logic for Programming, Artificial Intelligence, and Reasoning: 9th International Conference, LPAR 2002 Tbilisi, Georgia, October 14–18, 2002 Proceedings 9*, pages 231–246. Springer, 2002.

[6] William A Howard et al. The formulae-as-types notion of construction. *To HB Curry: essays on combinatory logic, lambda calculus and formalism*, 44:479–490, 1980.

[7] Aravind K Joshi and Seth Kulick. Partial proof trees, resource sensitive logics and syntactic constraints. In *International Conference on Logical Aspects of Computational Linguistics*, pages 21–42. Springer, 1996.

[8] Tom Leinster. Basic category theory. *arXiv preprint arXiv:1612.09375*, 2016.

[9] Saunders Mac Lane. *Categories for the working mathematician*, volume 5. Springer Science & Business Media, 2013.

[10] Guy McCusker and Alessio Santamaria. On compositionality of dinatural transformations. In *27th EACSL Annual Conference on Computer Science Logic (CSL 2018)*, pages 33–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2018.

[11] Paul-André Melliès. A survival guide on coherence spaces, 2000.

[12] Uday S. Reddy. A linear logic model of state. 1993.

[13] Uday S Reddy. A linear logic model of state. *Electronic manuscript, University of Illinois (anonymous FTP from cs. uiuc. edu)*, 1993.

[14] Lutz Straßburger. Mell in the calculus of structures. *Theoretical Computer Science*, 309(1-3):213–285, 2003.

[15] Ross Tate, Michael Stepp, and Sorin Lerner. Generating compiler optimizations from proofs. *ACM Sigplan Notices*, 45(1):389–402, 2010.

[16] Jim Woodcock and Arthur Hughes. Unifying theories of parallel programming. In *International Conference on Formal Engineering Methods*, pages 24–37. Springer, 2002.