GRAPH-BASED LOGIC AND SKETCHES II: FINITE-PRODUCT CATEGORIES AND EQUATIONAL LOGIC (PRELIMINARY REPORT)

ATISH BAGCHI AND CHARLES WELLS

Abstract. It is shown that the proof theory for sketches and forms provided in [Bagchi and Wells, 1996] is strong enough to produce all the theorems of the entailment system for multisorted equational logic provided in [Goguen and Meseguer, 1982].

1. Introduction

In [Wells, 1990] the second author introduced the notion of **form**, a graph-based method of specification of mathematical structures that generalizes Ehresmann's sketches. In [Bagchi and Wells, 1996], the authors produced a structure for forms which provides a uniform proof theory based on finite-limit constructions for many types of forms, including all types of sketches and also forms that can specify higher-order structures in cartesian closed categories and toposes (among many others). The parameter in the proof theory that determines the types of constructions that can be made is the **constructor space**. For example, the constructor space for cartesian closed categories (with specified structure) is the finite-limit theory **CCC** for cartesian closed categories. In particular for the concerns of the present paper, the constructor space for structures that can be specified by finite products is a finite-limit theory **FinProd** for categories with specified finite products. This theory is described explicitly in [Bagchi and Wells, 1996].

Each finite product form F is given by a **syntactic category** denoted by $\mathsf{SynCat}[\mathbf{FinProd}, F]$. The logical structure in [Bagchi and Wells, 1996] identifies a statement as a **potential factorization** in $\mathsf{SynCat}[\mathbf{FinProd}, F]$, which is a diagram of the form

(1)
$$\begin{tikzcd} & \mathit{hyp} \arrow[d, "\mathit{claimcon}"] \\ \mathit{claim} \arrow[r, "\mathit{hypcon}"'] & \mathit{wksp} \end{tikzcd}$$

Research at MSI is supported in part by NSF grant DMS-9022140.

and the theorem that the given statement is true as an **actual factorization**

$$\begin{tikzcd} & \mathsf{hyp} \arrow[d, "\mathsf{claimcon}"] \\ \mathsf{claim} \arrow[ur, "\mathsf{verif}"] \arrow[r, "\mathsf{hypcon}"'] & \mathsf{wksp} \end{tikzcd}$$

of the diagram (1).

In [Goguen and Meseguer, 1982], Goguen and Meseguer produced a sound and complete entailment system for multisorted equational logic. In this paper, we verify that the theorems of that logic for a particular signature and equations all occur as actual factorizations in $\mathsf{SynCat}[\mathbf{FinProd}, F]$, where F is a **FinProd** form induced (in a manner to be described) by the given signature and equations. We also compare the expressive powers of these two systems.

2. Preliminaries

2.1. **Lists.** Given a set A, $\mathsf{List}[A]$ denotes the set of lists of elements of A, including the empty list. The kth entry in a list w of elements of A is denoted by w_k and the length of w is denoted by $\mathsf{Length}\,[w]$. The **range** of w, denoted by $\mathsf{Rng}[w]$, is the set of elements of A occurring as entries in w. If $f : A \to B$ is a function, $\mathsf{List}[f] : \mathsf{List}[A] \to \mathsf{List}[B]$ is by definition f "mapped over" $\mathsf{List}[A]$: If w is a list of elements of A, then the kth entry of $\mathsf{List}[f](w)$ is by definition $f(w_k)$. This makes List a functor from the category of sets to itself.

2.2. **Signatures.**

2.2.1. *Expressions and terms.* In the description that follows of the terms and equations for a signature, we use a notation that specifies the variables of a term or equation explicitly. In particular, one may specify variables that do not actually appear in the expression. For this reason, the formalism we introduce in the definitions below distinguishes an *expression* such as $f(x, g(y, x), z)$ from a *term*, which is an expression together with a specified set of typed variables; in this case that set could be for example $\{x, y, z, w\}$. This formalism is equivalent to that of [Goguen and Meseguer, 1982].

2.2.2. *Definition.* A pair (Σ, Ω) of sets together with two functions $\mathsf{Inp} : \Omega \to \mathsf{List}[\Sigma]$ and $\mathsf{Outp} : \Omega \to \Sigma$ is called a **signature**. Given a signature $\mathcal{S} := (\Sigma, \Omega)$, elements of Σ are called the **types** of $\mathcal{S}$ and the elements of Ω are called the **operations** of $\mathcal{S}$.

2.2.3. *Notation.* Given a signature $\mathcal{S} = (\Sigma, \Omega)$, we will denote the set Σ of types by $\mathsf{Types}[\mathcal{S}]$ and the set Ω of operations by $\mathsf{Oprns}[\mathcal{S}]$. For any $f \in \Omega$, the list $\mathsf{Inp}[f]$ is called the **input type list** of f and the type $\mathsf{Outp}[f]$ is the **output type** of f.

2.2.4. *Remark.* The input type list of f is usually called the **arity** of f, and the output type of f is usually called simply the **type** of f.

2.2.5. *Definition.* An operation f of a signature $\mathcal{S}$ is called a **constant** if and only if $\mathsf{Inp}[f]$ is the empty list.

2.2.6. *Definition.* A type γ of a signature $\mathcal{S}$ is said to be **inhabited** if and only if either

a) there is a constant of output type γ in $\mathcal{S}$, or

b) there is an operation f of output type γ for which every type in $\mathsf{Inp}[f]$ is inhabited.

The type γ is said to be **empty** if and only if it is not inhabited.

2.3. **Terms and equations.** In this section, we define the terms and equations of a given signature.

2.3.1. *Assumptions.* In these definitions, we make the following assumptions, useful for bookkeeping purposes.

A.1 We assume that we are given a signature $\mathcal{S}$ for which $\mathsf{Types}[\mathcal{S}] = \{\sigma^i \mid i \in I\}$ for some ordinal I.

A.2 For each $i \in I$, we assume there is an indexed set $\mathsf{Vbl}[\sigma^i] := \{x^i_j \mid j \in \omega\}$ whose elements are by definition **variables of type** σ^i. In this setting, x^i_j is the jth variable of type σ^i.

A.3 The set of variables is ordered by defining

$$x^i_j < x^k_l :\Leftrightarrow \begin{cases} \text{either} & i < k \\ \text{or} & i = k \text{ and } j < l \end{cases}$$

We also define $\mathsf{Vbl}[\mathcal{S}] := \cup_{i\in\omega} \mathsf{Vbl}[\sigma^i]$.

2.3.2. *Definition.* For any type τ, an **expression of type** τ is defined recursively as follows.

Expr.1 A variable of type τ is an expression of type τ.

Expr.2 If f is an operation with $\mathsf{Inp}[f] = (\gamma^i \mid i \in 1\,..\,n)$ and $\mathsf{Outp}[f] = \tau$, and $(e_i \mid i \in 1\,..\,n)$ is a list of expressions for which each e_i is of type γ^i, then $f(e_i \mid i \in 1\,..\,n)$ is an expression of type τ.

2.3.3. *Notation.* The type of a variable x is denoted by $\mathsf{Type}[x]$, so that in the notation of 2.3.1, $\mathsf{Type}[x^i_j] = \sigma^i$. This notation will be extended to include lists and sets of variables: If $W := \{x^1_2, x^1_3, x^2_1, x^3_2\}$, then $\mathsf{Type}[W] = \sigma^1\times\sigma^1\times\sigma^2\times\sigma^3$. (Note that this depends on the ordering given by A.1.) The type of an expression e will be denoted by $\mathsf{Type}[e]$. Thus the function Type is overloaded: it may be applied to variables, sets of variables, or expressions, and will in the following be applied to terms and equations as well.

2.3.4. *Definition.* For a given expression e, the list of variables in e, in order of appearance in e from left to right, counting repetitions, is called the **variable list** of e, denoted by $\mathsf{VarList}[e]$. $\mathsf{Rng}\big[\mathsf{VarList}[e]\big]$, the set of distinct variables occurring in e, is called the **variable set** of e and denoted by $\mathsf{VarSet}[e]$. The list $(\mathsf{List}[\mathsf{Type}])\big[\mathsf{VarList}[e]\big]$ is called the **type list** of e, denoted by $\mathsf{TypeList}[e]$. Thus if the kth entry of $\mathsf{VarList}[e]$ is x^i_j, then the kth entry of $\mathsf{TypeList}[e]$ is σ^i. The set $\mathsf{Rng}\big[\mathsf{TypeList}[e]\big]$, which is the set of distinct types occurring in e, is called the **type set** of e, denoted by $\mathsf{TypeSet}[e]$.

2.3.5. *Example.* Let e be the expression $f(x, g(y, x), z)$. If x and y are variables of type γ and z is of type τ, then the variable list of e is (x, y, x, z), the variable set is $\{x, y, z\}$, the type list is $(\gamma, \gamma, \gamma, \tau)$, and the type set is $\{\gamma, \tau\}$. Using the notation of A.2 and supposing $\gamma = \sigma^1$, $\tau = \sigma^2$, $x = x^1_1$, $y = x^1_2$ and $z = x^2_1$, we have $e = f(x^1_1, g(x^1_2, x^1_1), x^2_1)$ and the following statements hold:

$$\begin{aligned}
\mathsf{VarList}[e] &= (x^1_1, x^1_2, x^1_1, x^2_1)\\
\mathsf{VarSet}[e] &= \{x^1_1, x^1_2, x^2_1\}\\
\mathsf{TypeList}[e] &= (\sigma^1, \sigma^1, \sigma^1, \sigma^2)\\
\mathsf{TypeSet}[e] &= \{\sigma^1, \sigma^2\}
\end{aligned}$$

2.3.6. *Definition.* A **term** t for a signature $\mathcal{S}$ is determined by the following:

TD.1 A set $\mathsf{Var}[t]$ of typed variables. (It is a set, not a list, but it is ordered by the ordering of A.3 in 2.3.1.)

TD.2 An expression $\mathsf{Expr}[t]$.

TD.3 A type $\mathsf{Type}[t] \in \mathsf{Types}[\mathcal{S}]$.

These data must satisfy the following requirements:

TR.1 $\mathsf{VarSet}\big[\mathsf{Expr}[t]\big] \subseteq \mathsf{Var}[t]$.

TR.2 $\mathsf{Type}[t] = \mathsf{Type}\big[\mathsf{Expr}[t]\big]$.

2.3.7. *Notation.* A given term t will be represented as the list

$$(\mathsf{Expr}[t], \mathsf{Var}[t], \mathsf{Type}[t])$$

2.3.8. *Definition.* Let t be a term. The list $\mathsf{InputTypes}[t]$ is defined to be the list whose ith entry is the type of the ith variable in $\mathsf{Var}[t]$ using the ordering given by A.3 in 2.3.1. Thus if the kth entry of $\mathsf{Var}[t]$ is x^i_j, then the kth entry of $\mathsf{InputTypes}[t]$ is σ^i. Observe that there are no repetitions in $\mathsf{Var}[t]$ but there may well be repetitions in $\mathsf{InputTypes}[t]$.

2.3.9. *Example.* Let $e = f\,(x_1^1, g(x_2^1, x_1^1), x_1^2)$ as in Example 2.3.5, and suppose $\mathsf{Outp}[f] = \sigma^5$. Then there are many terms t with $\mathsf{Expr}[t] = e$, for example

$$t_1 := \left(e, \{x_1^1, x_2^1, x_1^2\}, \sigma^5\right)$$

and

$$t_2 := \left(e, \{x_1^1, x_2^1, x_3^1, x_1^2, x_5^7\}, \sigma^5\right)$$

We have $\mathsf{Type}[t_1] = \mathsf{Type}[t_2] = \sigma^5$ and (for example)

$$\mathsf{InputTypes}[t_1] = (\sigma^1, \sigma^1, \sigma^2)$$

and

$$\mathsf{InputTypes}[t_2] = (\sigma^1, \sigma^1, \sigma^1, \sigma^2, \sigma^7)$$

2.3.10. *Definition.* An **equation** E is determined by a set $\mathsf{Var}[E]$ of typed variables (ordered by our convention) and two expressions $\mathsf{Left}[E]$, $\mathsf{Right}[E]$, for which

ER.1 $\mathsf{Type}\big[\mathsf{Left}[E]\big] = \mathsf{Type}\big[\mathsf{Right}[E]\big]$.

ER.2 $\mathsf{VarSet}\big[\mathsf{Left}[E]\big] \cup \mathsf{VarSet}\big[\mathsf{Right}[E]\big] \subseteq \mathsf{Var}[E]$.

2.3.11. *Notation.* We will write $e =_V e'$ to denote an equation E with $V = \mathsf{Var}[E]$, $e = \mathsf{Left}[E]$ and $e' = \mathsf{Right}[E]$. The notation $\mathsf{Type}[E]$ will denote the common type of $\mathsf{Left}[E]$ and $\mathsf{Right}[E]$.

2.3.12. *Example.* Let e be the expression $f\,(x_1^1, g(x_2^1, x_1^1), x_1^2)$ of Example 2.3.5. Let $e' := g(x_2^1, x_3^1)$. Then there are many equations with e and e' as left and right sides, for example:

$$E_1 := f\left(x_1^1, g(x_2^1, x_1^1), x_1^2\right) =_{\{x_1^1, x_2^1, x_3^1, x_1^3\}} g(x_2^1, x_3^1) \tag{2}$$

and

$$E_2 := f\left(x_1^1, g(x_2^1, x_1^1), x_1^2\right) =_{\{x_1^1, x_2^1, x_3^1, x_1^2, x_2^5\}} g(x_2^1, x_3^1) \tag{3}$$

For later use, we need the following definition:

2.3.13. *Definition.* The **most concrete term** associated with an expression e is defined to be the unique term t with the properties that $\mathsf{Expr}[t] = e$ and $\mathsf{Var}[t] = \mathsf{VarSet}[e]$. The **most concrete equation** associated with two expressions e and e' is defined to be the unique equation E such that $\mathsf{Left}[E] = e$, $\mathsf{Right}[E] = e'$, and $\mathsf{Var}[E] = \mathsf{VarSet}[e] \cup \mathsf{VarSet}[e']$.

2.3.14. *Example.* We continue Example 2.3.12. The most concrete equation associated with the expressions $f\,(x_1^1, g(x_2^1, x_1^1), x_1^2)$ and $g(x_2^1, x_3^1)$ is

$$f\left(x_1^1, g(x_2^1, x_1^1), x_1^2\right) =_{\{x_1^1, x_2^1, x_3^1, x_1^2\}} g(x_2^1, x_3^1)$$

The most concrete term associated with $f\,(x_1^1, g(x_2^1, x_1^1), x_1^2)$ is

$$\left(f\left(x_1^1, g(x_2^1, x_1^1), x_1^2\right), \{x_1^1, x_2^1, x_1^2\}, \sigma^5\right)$$

The most concrete term associated with $g(x_2^1, x_3^1)$ is

$$\left(g(x_2^1, x_3^1), \{x_2^1, x_3^1\}, \sigma^5\right)$$

in which we must conclude that $\mathsf{Outp}[g] = \sigma^5$ because it is equated with an expression whose head is f.

3. Equational deduction

Goguen and Meseguer [1982] prove that the following rules for equational deduction in multisorted equational deduction are sound and complete.

reflexivity: $\dfrac{}{e =_V e}$.

symmetry: $\dfrac{e =_V e'}{e' =_V e}$.

transitivity: $\dfrac{e =_V e' \quad e' =_V e''}{e =_V e''}$.

concretion: Given a set V of typed variables, $x \in V$ and an equation $e =_V e'$ such that $x \in V \setminus (\mathsf{VarSet}[e] \cup \mathsf{VarSet}[e'])$, and given that $\mathsf{Type}[x]$ is inhabited,

$$\frac{e =_V e'}{e =_{V\setminus\{x\}} e'}$$

abstraction: Given a set V of typed variables and $x \in \mathsf{Vbl}[\mathcal{S}] \setminus V$,

$$\frac{e =_V e'}{e =_{V\cup\{x\}} e'}$$

substitutivity: Given a set V of typed variables, $x \in V$, and expressions u and u' for which $\mathsf{Type}[x] = \mathsf{Type}[u] = \mathsf{Type}[u']$,

$$\frac{e =_V e' \quad u =_W u'}{e[x \leftarrow u] =_{V\setminus\{x\}\cup W} e'[x \leftarrow u']}$$

4. The sketch associated to a signature

We now show how to construct a finite-product sketch $\mathcal{S}$ corresponding to a given signature in such a way that the categories of models of the signature and of the sketch are naturally equivalent.

Given a signature $\mathcal{S} = (\Sigma, \Omega)$, we now construct a **FinProd** sketch $\mathsf{Sk}[\mathcal{S}]$. This sketch, like any finite-product sketch, determines and is determined (up to isomorphism) by a finite-product form F: Precisely (see [Bagchi and Wells, 1996], Section 6), there is a diagram $\delta : I \to$ **FinProd** and a global element $\ulcorner \mathsf{F} \urcorner : 1 \to \mathsf{v}$ in $\mathsf{SynCat}[\mathbf{FinProd}, \mathsf{F}]$, where v is the limit of δ, with the property that the value of $\ulcorner \mathsf{F} \urcorner$ in the initial model of $\mathsf{SynCat}[\mathbf{FinProd}, \mathsf{F}]$ in **Set** consists (up to isomorphism) of the graph, diagrams and (discrete) cones that make up the sketch $\mathcal{S}$. Moreover, the finite-product theory $\mathsf{FPTh}\big[\mathsf{Sk}[\mathcal{S}]\big]$ (defined in [Barr and Wells, 1995], section 7.5) is equivalent as a category to the finite-product category $\mathsf{CatTh}\big[\mathbf{FinProd}, \mathsf{F}\big]$ as defined in [Bagchi and Wells, 1996].

4.1. The graphs and cones of $\mathsf{Sk}[\mathcal{S}]$.

In what follows, we recursively define arrows and commutative diagrams in $\mathsf{Sk}[\mathcal{S}]$ associated to terms and equations of $\mathcal{S}$ respectively.

4.1.1. *Definition.* The set of nodes of $\mathsf{Sk}[\mathcal{S}]$ consist by definition of the following:

OS.1 Each type of $\mathcal{S}$ is a node.

OS.2 Each list $v = (\gamma_i \mid i \in 1\,.\,.\,n)$ that is the input type list (see Remark 2.2.4) of at least one operation in Ω is a node.

4.1.2. *Definition.* The arrows of $\mathsf{Sk}[\mathcal{S}]$ consist by definition of the following:

AS.1 Each operation f in Ω is an arrow $f : \mathsf{Inp}[f] \to \mathsf{Outp}[f]$.

AS.2 For each list $v = (\gamma_i \mid i \in 1\,.\,.\,n)$ that is the input type list of some operation in Ω, there is an arrow $\mathsf{Proj}[i] : v \to \gamma_i$ for each $i \in 1\,.\,.\,n$. (We will write $\mathsf{Proj}[v, i]$ for $\mathsf{Proj}[i]$ if necessary to to avoid confusion, and on the other hand we will write p_i for $\mathsf{Proj}[i]$ in some diagrams to save space.)

4.1.3. *Definition.* The cones of $\mathsf{Sk}[\mathcal{S}]$ consist by definition of the following: For each list $v = (\gamma_1, \ldots, \gamma_n)$ that is the input type list of some operation in Ω, there is a cone of $\mathsf{Sk}[\mathcal{S}]$ with vertex v and an arrow $\mathsf{Proj}[i] : v \to \gamma_i$ for each $i \in 1\,.\,.\,n$.

It follows that in a model M of the sketch $\mathsf{Sk}[\mathcal{S}]$, $M(v) = \prod_{i \in 1..n} M(\gamma_i)$.

4.1.4. *Constants.* If the signature contains constants, then one of the lists mentioned in OS.2 is the empty list. As a consequence, the sketch will contain an empty cone by Definition 4.1.3, and the vertex will become a terminator in a model.

4.2. **Terms as arrows.** We now describe how to associate each term of a signature $\mathcal{S}$ to an arrow in $\mathsf{CatTh}\big[\textbf{FinProd}, \mathsf{Sk}[\mathcal{S}]\,\big]$ and each equation to a commutative diagram or a pair of equal arrows in $\mathsf{CatTh}\big[\textbf{FinProd}, \mathsf{Sk}[\mathcal{S}]\,\big]$. The constructions given here are an elaboration of those in [Barr and Wells, 1995], pages 185–186.

4.2.1. *The arrow in* $\mathsf{CatTh}\big[\textbf{FinProd}, \mathsf{Sk}[\mathcal{S}]\,\big]$ *corresponding to a term.* We first define recursively two arrows $Q[e]$ and $I[e]$ of $\mathsf{CatTh}\big[\textbf{FinProd}, \mathsf{Sk}[\mathcal{S}]\,\big]$ for each expression e, and an arrow $D[t]$ of $\mathsf{CatTh}\big[\textbf{FinProd}, \mathsf{Sk}[\mathcal{S}]\,\big]$ for each term t. The arrow $\mathsf{Arr}[t] := Q\big[\mathsf{Expr}[t]\,\big] \circ I\big[\mathsf{Expr}[t]\,\big] \circ D[t]$ will then be the arrow of $\mathsf{CatTh}\big[\textbf{FinProd}, \mathsf{Sk}[\mathcal{S}]\,\big]$ associated with t; the meaning of the term t in a model of the signature is up to equivalence the same function as the value of $\mathsf{Arr}[t]$ in the corresponding model of $\mathsf{Sk}[\mathcal{S}]$.

In these definitions, we suppress mention of the universal model of $\mathsf{Sk}[\mathcal{S}]$. For example, if the universal model is $\mathsf{UnivMod}[\mathcal{S}] : \mathsf{Sk}[\mathcal{S}] \to \mathsf{CatTh}\big[\textbf{FinProd}, \mathsf{Sk}[\mathcal{S}]\,\big]$ and Θ is a node of $\mathsf{Sk}[\mathcal{S}]$, then we write Θ instead of $\mathsf{UnivMod}[\Theta]$. We treat arrows of $\mathsf{Sk}[\mathcal{S}]$ similarly.

4.2.2. *Definition.* For an expression e, $Q[e]$ is defined recursively by these requirements:

Q.1 If e is a variable x of type τ, then $Q[e] := \mathsf{Id}[\tau]$. Using the notation introduced in A.2, if $e = x^i_j$, then $Q[e] = \mathsf{Id}[\sigma^i]$,

Q.2 Suppose $e = f(e_i \mid i \in 1\,..\,n)$, where f is an operation with $\mathsf{Inp}[f] = (\gamma_i \mid i \in 1\,..\,n)$ and $\mathsf{Outp}[f] = \tau$. By definition, for $i \in (1\,..\,n)$, $\mathsf{Type}[e_i] = \gamma_i$.

Then $Q[e]$ is defined to be the arrow

$$\prod_{i=1}^n \mathsf{Dom}[Q(e_i)] \xrightarrow{\prod_{i=1}^n Q[e_i]} \prod_{i=1}^n \gamma_i \xrightarrow{f} \tau \tag{4}$$

4.2.3. *Remark.* We note that if $n = 0$ in Q.2, in other words $\mathsf{Inp}[f]$ is empty, the composite in (4) becomes

$$1 \longrightarrow 1 \xrightarrow{f} \tau$$

4.2.4. *Definition.* Let e be the expression described in Definition 4.2.2 Q.2, so that $\mathsf{TypeList}[e]$ is the concatenate $\mathsf{TypeList}[e_1] \cdots \mathsf{TypeList}[e_n]$. Then $I[e]$ is defined to be the canonical isomorphism

$$I[e] : \prod \mathsf{TypeList}[e] \to \prod_{i=1}^n \mathsf{TypeList}[e_i]$$

given by the associative law for categorial products in $\mathsf{CatTh}\big[\textbf{FinProd}, \mathsf{Sk}[\mathcal{S}]\,\big]$.

4.2.5. *Definition.* Let t be an arbitrary term. Then

$$D[t] : \prod \mathsf{InputTypes}[t] \to \prod \mathsf{TypeList}\big[\mathsf{Expr}[t]\,\big]$$

is defined to be the unique arrow induced by requiring that the following diagrams commute for each pair

$$(i, k) \in \big(1\,..\,\mathsf{Length}\,[\mathsf{InputTypes}[t]]\big) \times \big(1\,..\,\mathsf{Length}\,[\mathsf{TypeList}\,[\mathsf{Expr}[t]\,]]\big)$$

with the property that the ith variable from the left in $\mathsf{Expr}[t]$ is $(\mathsf{Var}[t])_k$.

(5)
$$\begin{tikzcd}
\prod \mathsf{InputTypes}[t] \arrow[rr, "D{[t]}"] \arrow[dr, "\mathsf{Proj}{[k]}"'] & & \prod \mathsf{TypeList}\big[\mathsf{Expr}[t]\,\big] \arrow[dl, "\mathsf{Proj}{[i]}"] \\
& (\mathsf{InputTypes}[t])_k &
\end{tikzcd}$$

Alternatively, suppose $\mathsf{Var}[t]$ has length L and $\mathsf{VarSet}\big[\mathsf{Expr}[t]\,\big]$ has length M. Let $\phi : 1\,..\,L \to 1\,..\,M$ be defined by $\phi(l) = m$ if $(\mathsf{Var}[t])_l = (\mathsf{VarList}\big[\mathsf{Expr}[t]\,\big])_m$ (there is a unique m that makes this true). Then we may also define

$$D[t] : \prod \mathsf{InputTypes}[t] \to \prod \mathsf{TypeList}\big[\mathsf{Expr}[t]\,\big]$$

to be the arrow $(\mathsf{Proj}[\phi(l)] \mid l \in 1\,..\,L)$.

This works because the $(\phi(l))$th type in $\prod \mathsf{TypeList}\big[\mathsf{Expr}[t]\,\big]$ is indeed the type of the $(\phi(l))$th variable in $\mathsf{VarSet}\big[\mathsf{Expr}[t]\,\big]_m$ (see Definition 2.3.4).

This is equivalent to requiring the diagrams(5) to commute. The two definitions are useful for different sorts of calculations and are therefore included.

4.2.6. *Example.* Consider $e := g(x_1^1, c)$, where c is a constant of type σ^2 and g has type σ^3. Suppose

$$t = (g(x_1^1, c), \{x_1^1, x_1^4\}, \sigma^3)$$

Then e corresponds to the arrow

$$\begin{tikzcd}
\sigma^1 \times \sigma^4 \arrow[d, "\mathsf{Proj}[1]"'] & {} \arrow[d, phantom, "\}D[t]"] \\
\sigma^1 \arrow[d, "\langle \mathsf{Id}[\sigma^1], ! \rangle"'] & {} \arrow[d, phantom, "\}I[t]"] \\
\sigma^1 \times 1 \arrow[d, "\mathsf{Id}[\sigma^1]\times c"'] & {} \arrow[dd, phantom, "\}Q[t]"] \\
\sigma^1 \times \sigma^2 \arrow[d, "g"'] & {} \\
\sigma^3 &
\end{tikzcd}$$

Note that one does not have to consider constants in constructing $D[t]$.

4.2.7. *Example.* Let $e := f\,(x_1^1, g(x_2^1, x_1^1), x_1^2)$ with $\mathsf{Inp}[g] = (\sigma^1, \sigma^1)$, $\mathsf{Outp}[g] = \sigma^2$. $\mathsf{Inp}[f] = (\sigma^1, \sigma^2, \sigma^2)$, and $\mathsf{Outp}[f] = \sigma^5$. Let

$$t := (e, \{x_1^1, x_2^1, x_1^2, x_3^4\}, \sigma^5)$$

Then

$$\mathsf{VarList}\big[\mathsf{Expr}[t]\,\big] = (x_1^1, x_2^1, x_1^1, x_1^2)$$
$$\mathsf{InputTypes}[t] = (\sigma^1, \sigma^1, \sigma^2, \sigma^4)$$
$$\mathsf{Var}[t] = \{x_1^1, x_2^1, x_1^2, x_3^4\}$$

and

$$\mathsf{TypeList}\big[\mathsf{Expr}[t]\,\big] = (\sigma^1, \sigma^1, \sigma^1, \sigma^2)$$

If we use the first definition of $D[t]$ in Definition 4.2.5, then the following four triangles must commute:

$$\begin{tikzcd}
\sigma^1{\times}\sigma^1{\times}\sigma^2{\times}\sigma^4 \arrow[rr, "D[t]"] \arrow[dr, "\mathsf{Proj}[1]"'] & & x\sigma^1{\times}\sigma^1{\times}\sigma^1{\times}\sigma^2 \arrow[dl, "\mathsf{Proj}[1]"] \\
& \sigma^1 &
\end{tikzcd}
\qquad
\begin{tikzcd}
\sigma^1{\times}\sigma^1{\times}\sigma^2{\times}\sigma^4 \arrow[rr, "D[t]"] \arrow[dr, "\mathsf{Proj}[2]"'] & & \sigma^1{\times}\sigma^1{\times}\sigma^1{\times}\sigma^2 \arrow[dl, "\mathsf{Proj}[2]"] \\
& \sigma^1 &
\end{tikzcd}$$

(6)

$$\begin{tikzcd}
\sigma^1\times\sigma^1\times\sigma^2\times\sigma^4 \arrow[rr, "D[t]"] \arrow[dr, "\mathsf{Proj}[1]"'] & & \sigma^1\times\sigma^1\times\sigma^1\times\sigma^2 \arrow[dl, "\mathsf{Proj}[3]"] \\
& \sigma^1 &
\end{tikzcd}
\qquad
\begin{tikzcd}
\sigma^1\times\sigma^1\times\sigma^2\times\sigma^4 \arrow[rr, "D[t]"] \arrow[dr, "\mathsf{Proj}[3]"'] & & \sigma^1\times\sigma^1\times\sigma^1\times\sigma^2 \arrow[dl, "\mathsf{Proj}[4]"] \\
& \sigma^2 &
\end{tikzcd}$$

It follows that $D[t]$ is given by the following diagram, where to save space we write p_k for $\mathsf{Proj}[k]$.

$$\sigma^1\times\sigma^1\times\sigma^2\times\sigma^4 \xrightarrow{\langle p_1, p_2, p_1, p_3\rangle} \sigma^1\times\sigma^1\times\sigma^1\times\sigma^2 \tag{7}$$

and that $\mathsf{Arr}[t]$ is the composite

$$\begin{tikzcd}
\sigma^1\times\sigma^1\times\sigma^2\times\sigma^4 \arrow[d, "{\langle p_1, p_2, p_1, p_3\rangle}"'] \arrow[d, phantom, shift left=8ex, "\big\}D[t]"] \\
\sigma^1\times\sigma^1\times\sigma^1\times\sigma^2 \arrow[d, "{\langle p_1, \langle p_2, p_3\rangle, p_4\rangle}"'] \arrow[d, phantom, shift left=8ex, "\big\}I[t]"] \\
\sigma^1\times(\sigma^1\times\sigma^1)\times\sigma^2 \arrow[d, "{\mathsf{Id}[\sigma^1]\times g\times \mathsf{Id}[\sigma^2]}"'] \\
\sigma^1\times\sigma^2\times\sigma^2 \arrow[d, "f"'] \arrow[phantom, shift left=8ex, "\Big\}Q[t]"] \\
\sigma^5
\end{tikzcd}$$

4.3. **The diagram associated to an equation.** Let the equation $E := e =_V e'$ be given. Define the terms t_1 and t_2 by $t_1 = (e, \mathsf{Var}[E], \mathsf{Type}[E])$ and $t_2 = (e', \mathsf{Var}[E], \mathsf{Type}[E])$ (using the notation of 2.3.6). Recall that $\mathsf{Type}[E] = \mathsf{Type}[e] = \mathsf{Type}[e']$. The notation $\mathsf{InputTypes}[E]$ will denote the list $\mathsf{InputTypes}[t_1]$, which is the same as $\mathsf{InputTypes}[t_2]$. As in 4.2.1, we have arrows $\mathsf{Arr}[t_1]$ and $\mathsf{Arr}[t_2]$ with the same domain and codomain. We will associate the diagram

$$\begin{tikzcd}
\mathsf{InputTypes}[E] \arrow[r, shift left, "{\mathsf{Arr}[t_1]}"] \arrow[r, shift right, "{\mathsf{Arr}[t_2]}"'] & \mathsf{Type}[E]
\end{tikzcd} \tag{8}$$

to the equation E. By 4.2.1, this is the same as

$$\begin{tikzcd}
\mathsf{InputTypes}[E] \arrow[r, "D[t_1]"] \arrow[d, "D[t_2]"'] & \mathsf{TypeList}[e] \arrow[d, "Q[t_1]\circ I[t_1]"] \\
\mathsf{TypeList}[e'] \arrow[r, "Q[t_2]\circ I[t_2]"'] & \mathsf{Type}[E]
\end{tikzcd} \tag{9}$$

This completes the translation.

4.3.1. *Remark.* The commutative diagram as exhibited above can also be viewed as a pair of formally equal arrows as in Diagram (8), and in what follows we will use this description frequently.

4.4. **Examples.** We work out below two examples in detail to facilitate later discussion of substitution.

4.4.1. *Example.*

$$e := f\,(x_1^1, x_3^4, x_2^3, x_1^1, g(x_1^1, x_2^3), x_1^2)$$

$$\begin{cases} \mathsf{Inp}[f] = \sigma^1 \times \sigma^4 \times \sigma^3 \times \sigma^1 \times \sigma^5 \times \sigma^2 \\ \mathsf{Outp}[f] = \sigma^5 \end{cases}$$

$$\begin{cases} \mathsf{Inp}[g] = \sigma^1 \times \sigma^3 \\ \mathsf{Outp}[g] = \sigma^5 \end{cases}$$

$$V = \{x_1^1, x_2^1, \underline{x_3^1}, x_1^2, \underline{x_2^2}, \underline{x_1^3}, x_2^3, x_3^4\}$$

The underlined variables are redundant; that is, they do not appear in the expression e.

$$\begin{cases} u := h(x_1^2, x_2^3) \\ \mathsf{Inp}[h] = \sigma^2 \times \sigma^3 \\ \mathsf{Outp}[h] = \sigma^3 \end{cases}$$

$$W := \{\underline{x_1^1}, x_1^2, \underline{x_2^2}, \underline{x_3^2}, \underline{x_1^3}, x_2^3, \underline{x_3^3}\}$$

x_2^3 is a variable for which we are making a substitution. We wish to calculate

$$(e, V, \sigma^5)\left[x_2^3 \leftarrow (u, W, \sigma^3)\right] = \left(e[x_2^3 \leftarrow u], (V \setminus \{x_2^3\}) \cup W, \sigma^5\right)$$

By direct calculation,

$$(V \setminus \{x_2^3\}) \cup W = \{x_1^1, x_2^1, \underline{x_3^1}, x_1^2, \underline{x_2^2}, \underline{x_3^2}, \underline{x_1^3}, x_2^3, \underline{x_3^3}, x_3^4\}$$

and

$$e(x_2^3 \leftarrow u) = f\,\left(x_1^1, x_3^4, e(x_1^2, x_2^3), x_1^1, g(x_2^1, e(x_1^2, x_2^3)), x_1^2\right)$$

We now exhibit the arrows for e and u over V: $e = f\,(x^1_1, x^4_3, x^3_2, x^1_1, g(x^1_1, x^3_2), x^2_1)$:

$$
\begin{array}{rcl}
 & \sigma^1\times\sigma^1\times\underline{\sigma}^1\times\sigma^2\times\underline{\sigma}^2\times\underline{\sigma}^3\times\sigma^3\times\sigma^4 & \\
\langle p_1, p_8, p_7, p_1, p_2, p_7, p_4\rangle & \Big\downarrow & \Big\}\,D[e] \\
 & \sigma^1\times\sigma^4\times\sigma^3\times\sigma^1\times\sigma^1\times\sigma^3\times\sigma^2 & \\
\langle p_1, p_2, p_3, p_4, \langle p_5, p_6\rangle, p_7\rangle & \Big\downarrow & \Big\}\,I[e] \\
 & \sigma^1\times\sigma^4\times\sigma^3\times\sigma^1\times(\sigma^1\times\sigma^3)\times\sigma^2 & \\
\mathsf{Id}[\sigma^1]\times\mathsf{Id}[\sigma^4]\times\mathsf{Id}[\sigma^3]\times\mathsf{Id}[\sigma^1]\times g\times\mathsf{Id}[\sigma^2] & \Big\downarrow & \\
 & \sigma^1\times\sigma^4\times\sigma^3\times\sigma^1\times\sigma^5\times\sigma^2 & \Bigg\}\,Q[e] \\
f & \Big\downarrow & \\
 & \sigma^5 &
\end{array}
$$

$u = h(x^2_1, x^3_2)$ (over W):

$$
\begin{array}{rcl}
 & \underline{\sigma}^1\times\sigma^2\times\underline{\sigma}^2\times\underline{\sigma}^2\times\underline{\sigma}^3\times\sigma^3\times\underline{\sigma}^3 & \\
\langle p_2, p_6\rangle & \Big\downarrow & \Big\}\,D[u] \\
 & \sigma^2\times\sigma^3 & \\
\langle p_1, p_2\rangle = \mathsf{Id}[\sigma^2\times\sigma^3] = \mathsf{Id}[\sigma^2]\times\mathsf{Id}[\sigma^3] & \Big\downarrow & \Big\}\,I[u] \\
 & \sigma^2\times\sigma^3 & \\
h & \Big\downarrow & \Big\}\,Q[u] \\
 & \sigma^3 &
\end{array}
$$

Therefore $\mathsf{Arr}[u, W, \sigma^3] = h\langle p_1, p_2\rangle\langle p_2, p_6\rangle = h\langle p_2, p_6\rangle$ and $u = h(x_1^2, x_2^3)$ (over $(V \setminus \{x_2^3\}) \cup W$) is the arrow

$$\begin{tikzcd}
\underline{\sigma}^1\times\underline{\sigma}^1\times\underline{\sigma}^1\times\sigma^2\times\underline{\sigma}^2\times\underline{\sigma}^2\times\underline{\sigma}^3\times\sigma^3\times\underline{\sigma}^3\times\underline{\sigma}^4 \arrow[d, "\langle p_4, p_8\rangle"', "\Big\}D[u]"] \\
\sigma^2\times\sigma^3 \arrow[d, "\langle p_1, p_2\rangle = \mathsf{Id}[\sigma^2\times\sigma^3] = \mathsf{Id}[\sigma^2]\times\mathsf{Id}[\sigma^3]"', "\Big\}I[u]"] \\
\sigma^2\times\sigma^3 \arrow[d, "h"', "\Big\}Q[u]"] \\
\sigma^3
\end{tikzcd}$$

so that $\mathsf{Arr}[u, (V \setminus \{x_2^3\}) \cup W, \sigma^3] = h\langle p_1, p_2\rangle\langle p_4, p_8\rangle = h\langle p_4, p_8\rangle$.

Note that we have the maps

$$\alpha : \mathsf{Type}[W] \to \mathsf{Type}\big[\big(V \setminus \{x_2^3\}\big) \cup W\big]$$
$$\beta : \mathsf{Type}\big[\big(V \setminus \{x_2^3\}\big) \cup W\big] \to \mathsf{Type}[W]$$

as shown below:

$$\begin{tikzcd}
\mathsf{Type}[W] = \underline{\sigma}^1\times\sigma^2\times\underline{\sigma}^2\times\underline{\sigma}^2\times\underline{\sigma}^3\times\sigma^3\times\underline{\sigma}^3 \arrow[d, "\alpha = \langle x_1^1!, x_2^1!, x_3^1!, p_2, x_2^2!, x_2^3!, x_1^3!, p_6, x_3^3!, x_3^4!\rangle"'] \\
\underline{\sigma}^1\times\underline{\sigma}^1\times\underline{\sigma}^1\times\sigma^2\times\underline{\sigma}^2\times\underline{\sigma}^2\times\underline{\sigma}^3\times\sigma^3\times\underline{\sigma}^3\times\underline{\sigma}^4
\end{tikzcd}$$

where the codomain is

$$\mathsf{Type}\big[\big(V \setminus \{x_2^3\}\big) \cup W\big]$$

and where we have identified the variable x_j^i with

$$W \xrightarrow{\;!\;} 1 \xrightarrow{x_j^i} \sigma^i$$

The map β is similarly defined. It follows that

$$\langle p_4, p_8\rangle\alpha = \langle p_2, p_6\rangle$$

and

$$\langle p_2, p_6\rangle\beta = \langle p_4, p_8\rangle$$

and that

$$\mathsf{Arr}[u, W, \sigma^3] = \mathsf{Arr}\big[u, \big(V \setminus \{x_2^3\}\big) \cup W, \sigma^3\big]\circ\alpha$$

and

$$\mathsf{Arr}\big[u, \big(V \setminus \{x_2^3\}\big) \cup W, \sigma^3\big] = \mathsf{Arr}[u, W, \sigma^3]\circ\beta$$

These observations, although made in this special case, capture general features of the system that we shall need later. We record these in passing in the following lemmas. The proofs are quite straightforwarded and are omitted in view of the preceding example.

4.4.2. *Lemma. Let* $t = (\mathsf{Expr}[t], \mathsf{Var}[t], \mathsf{Type}[t])$ *be any term. Then*

$$Q[t] = Q\big[\mathsf{Expr}[t]\big] = Q\big[\mathsf{Expr}[t], \mathsf{Var}[\mathsf{Expr}[t]], \mathsf{Type}[t]\big]$$

and

$$I[t] = I\big[\mathsf{Expr}[t]\big] = I\big[\mathsf{Expr}[t], \mathsf{Var}[\mathsf{Expr}[t]], \mathsf{Type}[t]\big]$$

are determined by the most concrete term associated with t *as defined in 2.3.13. These specifically do not depend on* $\mathsf{Var}[t]$.

Note that by contrast $D[t]$ does depend on $\mathsf{Var}[t]$.

4.4.3. *Lemma. Let* e *be an expression of type* τ*, and let* V_1 *and* V_2 *be lists of variables such that* $\mathsf{VarSet}[e] \subseteq V_1$ *and* $\mathsf{VarSet}[e] \subseteq V_2$*. Let*

$$t_1 := [e, V_1, \tau]$$

and

$$t_2 := [e, V_2, \tau]$$

Then there are arrows

$$\alpha_{12} : \prod \mathsf{TypeList}[t_1] \to \prod \mathsf{TypeList}[t_2]$$

and

$$\alpha_{21} : \prod \mathsf{TypeList}[t_2] \to \prod \mathsf{TypeList}[t_1]$$

for which $\mathsf{Arr}[t_1] = \mathsf{Arr}[t_2] \circ \alpha_{12}$ *and* $\mathsf{Arr}[t_2] = \mathsf{Arr}[t_1] \circ \alpha_{21}$.

4.4.4. *Remark.* Lemmas 4.4.2 and 4.4.3 will be used later in our discussion of the rules “concretion” and “abstraction” that have to do with including extraneous variables in and excluding them from the list of variables of some term.

We now proceed with our example. After substitution,

$$e(x_2^3 \leftarrow u) = f\left(x_1^1, x_3^4, h(x_1^2, x_2^3), x_1^1, g(x_2^1, h(x_1^2, x_2^3)), x_1^2\right)$$

This corresponds to the arrow

$$
\begin{tikzcd}
\sigma^1\times\sigma^1\times\underline{\sigma}^1\times\sigma^2\times\underline{\sigma}^2\times\underline{\sigma}^2\times\underline{\sigma}^3\times\sigma^3\times\underline{\sigma}^3\times\sigma^4 \arrow[d, "\langle p_1,p_{10},p_4,p_8,p_1,p_2,p_4,p_8,p_4\rangle"', "\}\,D[e]"] \\
\sigma^1\times\sigma^4\times\sigma^2\times\sigma^3\times\sigma^1\times\sigma^2\times\sigma^3\times\sigma^2 \arrow[d, "\langle p_1,p_2,\langle p_3,p_4\rangle,\langle p_4,\langle p_7,p_8\rangle\rangle\,,p_9\rangle"', "\}\,I[e]"] \\
\sigma^1\times\sigma^4\times(\sigma^2\times\sigma^3)\times\sigma^1\times\left(\sigma^1\times(\sigma^2\times\sigma^3)\right)\times\sigma^2 \arrow[d, "\mathsf{Id}[\sigma^1]\times\mathsf{Id}[\sigma^4]\times\mathsf{Id}[\sigma^2\times\sigma^3]\times\mathsf{Id}[\sigma^1]\times(\mathsf{Id}[\sigma^1]\times h)\times\mathsf{Id}[\sigma^2]"'] \\
\sigma^1\times\sigma^4\times(\sigma^2\times\sigma^3)\times\sigma^1\times(\sigma^1\times\sigma^3)\times\sigma^2 \arrow[d, "\mathsf{Id}[\sigma^1]\times\mathsf{Id}[\sigma^4]\times h\times\mathsf{Id}[\sigma^1]\times g\times\mathsf{Id}[\sigma^2]"'] \\
\sigma^1\times\sigma^4\times\underline{\sigma}^3\times\sigma^1\times\underline{\sigma}^5\times\sigma^2 \arrow[d, "f"'] \\
\sigma^5
\end{tikzcd}
$$

We now calculate

$$
\begin{aligned}
e\left[x_2^3 \leftarrow u\right] &= f\left(\mathsf{Id}[\sigma^1]\times \mathsf{Id}[\sigma^4]\times e\times \mathsf{Id}[\sigma^1]\times g\times \mathsf{Id}[\sigma^3]\right)\\
&\quad \circ\left(\mathsf{Id}[\sigma^1]\times \mathsf{Id}[\sigma^4]\times \mathsf{Id}[\sigma^2\times\sigma^3]\times \mathsf{Id}[\sigma^1]\times(\mathsf{Id}[\sigma^1]\times e)\times \mathsf{Id}[\sigma^2]\right)\\
&\quad \circ\langle p_1,p_2,\langle p_3,p_4\rangle,p_5,\langle p_6,\langle p_7,p_8\rangle\rangle,p_9\rangle\\
&\quad \circ\langle p_1,p_{10},p_4,p_8,p_1,p_2,p_4,p_8,p_4\rangle\\
&= f\left(\mathsf{Id}[\sigma^1]\times \mathsf{Id}[\sigma^4]\times \mathsf{Id}[\sigma^3]\times \mathsf{Id}[\sigma^1]\times g\times \mathsf{Id}[\sigma^2]\right)\\
&\quad \circ\langle p_1,p_2,e\langle p_3,p_4\rangle,p_8,\langle p_6,e\langle p_7,p_8\rangle\rangle,p_9\rangle\\
&\quad \circ\langle p_1,p_{10},p_4,p_8,p_1,p_2,p_4,p_8,p_4\rangle\\
&= f\left(\mathsf{Id}[\sigma^1]\times \mathsf{Id}[\sigma^4]\times \mathsf{Id}[\sigma^3]\times \mathsf{Id}[\sigma^1]\times g\times \mathsf{Id}[\sigma^2]\right)\\
&\quad \circ\langle p_1,p_2,p_3,p_4,\langle p_5,p_6\rangle,p_7\rangle\\
&\quad \circ\langle p_1,p_2,e\langle p_3,p_4\rangle,p_5,p_6,e\langle p_7,p_8\rangle,p_9\rangle\\
&\quad \circ\langle p_1,p_{10},p_4,p_8,p_1,p_2,p_4,p_8,p_4\rangle\\
&= f\left(\mathsf{Id}[\sigma^1]\times \mathsf{Id}[\sigma^4]\times \mathsf{Id}[\sigma^3]\times \mathsf{Id}[\sigma^1]\times g\times \mathsf{Id}[\sigma^2]\right)\\
&\quad \circ\langle p_1,p_2,p_3,p_4,\langle p_5,p_6\rangle,p_7\rangle\\
&\quad \circ\langle p_1,p_{10},e\langle p_4,p_8\rangle,p_1,p_2,e\langle p_4,p_8\rangle,p_4\rangle\\
&= f\left(\mathsf{Id}[\sigma^1]\times \mathsf{Id}[\sigma^4]\times \mathsf{Id}[\sigma^3]\times \mathsf{Id}[\sigma^1]\times g\times \mathsf{Id}[\sigma^2]\right)\\
&\quad \circ\langle p_1,p_2,p_3,p_4,\langle p_5,p_6\rangle,p_7\rangle\\
&\quad \circ\langle p_1,p_{10},p_8,p_1,p_2,p_8,p_4\rangle\\
&\quad \circ\langle p_1,p_2,p_3,p_4,p_5,p_6,p_7,e\langle p_4,p_8\rangle,p_9,p_{10}\rangle\\
&= \mathsf{Arr}\left[e,V\cup W,\mathsf{Type}[e]\right]\\
&\quad \circ\left\langle p_1,p_2,p_3,p_4,p_5,p_6,p_7,\mathsf{Arr}\left[u,\left(V\setminus\{x_2^3\}\right)\cup W,\mathsf{Type}[u]\right],p_9,p_{10}\right\rangle
\end{aligned}
$$

4.4.5. *Example.*

$$u = m(x_1^2, x_1^2, x_4^4)$$

$$\begin{cases}\mathsf{Inp}[u] = \sigma^2\times\sigma^2\times\sigma^4\\ \mathsf{Outp}[u] = \sigma^3\end{cases}$$

$$W = \{\underline{x}_1^1, x_1^2, \underline{x}_2^2, \underline{x}_3^2, \underline{x}_1^3, \underline{x}_3^3, x_4^4\}$$

This is different from Example 4.4.1 because the variable x_2^3 (in e) for which we are making the substitution does not reappear in u.

This is the arrow for $u = m(x_1^2, x_1^2, x_4^4)$ over W:

$$\begin{array}{rcl}
 & \sigma^1 \times \sigma^2 \times \underline{\sigma}^2 \times \underline{\sigma}^2 \times \underline{\sigma}^3 \times \underline{\sigma}^3 \times \sigma^4 & \\
 \langle p_1, p_1, p_7 \rangle & \Big\downarrow & \Big\} D[u] \\
 & \sigma^2 \times \sigma^2 \times \sigma^4 & \\
 \langle p_1, p_2, p_3 \rangle = \mathsf{Id}[\sigma^2 \times \sigma^2 \times \sigma^4] = \mathsf{Id}[\sigma^1] \times \mathsf{Id}[\sigma^2] \times \mathsf{Id}[\sigma^4] & \Big\downarrow & \Big\} I[u] \\
 & \sigma^2 \times \sigma^2 \times \sigma^4 & \\
 m & \Big\downarrow & \Big\} Q[u] \\
 & \sigma^5 &
\end{array}$$

We calculate

$$\left(V \setminus \{x_2^3\}\right) \cup W = \{x_1^1, x_2^1, \underline{x}_3^1, x_1^2, \underline{x}_2^2, \underline{x}_3^2, \underline{x}_1^3, \underline{x}_3^3, x_3^4, x_4^4\}$$

Then $u = m(x_1^2, x_1^2, x_4^4)$ (over $\left(V \setminus \{x_2^3\}\right) \cup W$) gives the arrow

$$\begin{array}{rcl}
 & \sigma^1 \times \sigma^1 \times \underline{\sigma}^1 \times \sigma^2 \times \underline{\sigma}^2 \times \underline{\sigma}^2 \times \underline{\sigma}^3 \times \underline{\sigma}^3 \times \sigma^4 \times \sigma^4 & \\
 \langle p_4, p_4, p_{10} \rangle & \Big\downarrow & \Big\} D[u] \\
 & \sigma^2 \times \sigma^2 \times \sigma^4 & \\
 \langle p_1, p_2, p_3 \rangle = \mathsf{Id}[\sigma^2 \times \sigma^2 \times \sigma^4] = \mathsf{Id}[\sigma^1] \times \mathsf{Id}[\sigma^2] \times \mathsf{Id}[\sigma^4] & \Big\downarrow & \Big\} I[u] \\
 & \sigma^2 \times \sigma^2 \times \sigma^4 & \\
 m & \Big\downarrow & \Big\} Q[u] \\
 & \sigma^5 &
\end{array}$$

After substitution,

$$e[x_2^3 \leftarrow u] = f\left(x_1^1, x_3^4, m\left(x_1^2, x_1^2, x_4^4\right), x_1^1, g\left(x_2^1, m\left(x_1^2, x_1^2, x_4^4\right)\right), x_1^2\right)$$

This corresponds to the arrow shown below.

$$
\begin{array}{cl}
\sigma^1\times\sigma^1\times\underline{\sigma}^1\times\sigma^2\times\underline{\sigma}^2\times\underline{\sigma}^2\times\underline{\sigma}^3\times\underline{\sigma}^3\times\sigma^4\times\sigma^4 & \\
\Big\downarrow \langle p_1,p_9,p_{10},p_4,p_4,p_{10},p_1,p_2,p_4,p_4,p_{10},p_2\rangle & \Big\}D[e] \\
\sigma^1\times\sigma^4\times\sigma^2\times\sigma^2\times\sigma^4\times\sigma^1\times\sigma^1\times\sigma^2\times\sigma^2\times\sigma^4\times\sigma^2 & \\
\Big\downarrow \Big\langle p_1,p_2,\langle p_3,p_4,p_5\rangle, p_6, \langle p_7,\langle p_8,p_9,p_{10}\rangle\rangle, p_{11}\Big\rangle & \Big\}I[e] \\
\sigma^1\times\sigma^4\times\left(\sigma^2\times\sigma^2\times\sigma^2\right)\times\sigma^1\times\Big(\sigma^1\left(\sigma^2\times\sigma^2\times\sigma^4\right)\Big)\times\sigma^2 & \\
\Big\downarrow \mathsf{Id}[\sigma^1]\times\mathsf{Id}[\sigma^4]\times\mathsf{Id}\left[\sigma^2\times\sigma^2\times\sigma^4\right]\times\mathsf{Id}[\sigma^2]\left(\mathsf{Id}[\sigma^1]\times h\right)\times\mathsf{Id}[\sigma^2] & \\
\sigma^1\times\sigma^4\times\left(\sigma^2\times\sigma^2\times\sigma^2\right)\times\sigma^1\times\Big(\sigma^1\times\underline{\sigma}^3\Big)\times\sigma^2 & \\
\Big\downarrow \mathsf{Id}[\sigma^1]\times\mathsf{Id}[\sigma^4]\times h\times\mathsf{Id}[\sigma^1]\times g\times\mathsf{Id}[\sigma^2] & \Bigg\}Q[e] \\
\sigma^1\times\sigma^4\times\underline{\sigma}^3\times\sigma^1\times\underline{\sigma}^5\times\sigma^2 & \\
\Big\downarrow f & \\
\tau &
\end{array}
$$

We next re-express this in a convenient form:

$$
\begin{aligned}
e[x_2^3 \leftarrow u] &= f \circ \big(\mathsf{Id}[\sigma^1]\times \mathsf{Id}[\sigma^4]\times h\times \mathsf{Id}[\sigma^2]\times g\times \mathsf{Id}[\sigma^2]\big)\\
&\quad \circ\big(\sigma^1\times\sigma^4\times \mathsf{Id}\left[\sigma^2\times\sigma^2\times\sigma^4\right] \times \mathsf{Id}[\sigma^1]\times \big(\mathsf{Id}[\sigma^1]\times h\big) \times \mathsf{Id}[\sigma^2]\big)\\
&\quad \circ\big\langle p_1, p_2, \langle p_3, p_4, p_5\rangle, p_6, \langle p_7, \langle p_8, p_9, p_{10}\rangle\rangle, p_{11}\big\rangle\\
&\quad \circ\big\langle p_1, p_9, p_4, p_4, p_{10}, p_1, p_2, p_4, p_4, p_{10}, p_2\big\rangle\\
&= f \circ \big(\mathsf{Id}[\sigma^1]\times \mathsf{Id}[\sigma^4]\times \mathsf{Id}[\sigma^3]\times \mathsf{Id}[\sigma^1]\times g\times \mathsf{Id}[\sigma^2]\big)\\
&\quad \circ\big\langle p_1, p_2, h\langle p_3, p_4, p_5\rangle, p_6, \langle p_7, h\langle p_8, p_9, p_{10}\rangle\rangle, p_{11}\big\rangle\\
&\quad \circ\big\langle p_1, p_9, p_4, p_4, p_{10}, p_1, p_2, p_4, p_4, p_{10}, p_2\big\rangle\\
&= f \circ \big(\mathsf{Id}[\sigma^1]\times \mathsf{Id}[\sigma^4]\times \mathsf{Id}[\sigma^3]\times \mathsf{Id}[\sigma^1]\times g\times \mathsf{Id}[\sigma^2]\big)\\
&\quad \circ\big\langle p_1, p_9, h\langle p_4, p_4, p_{10}\rangle, p_1, \langle p_2, h\langle p_4, p_4, p_{10}\rangle\rangle, p_2\big\rangle\\
&= f \circ \big(\mathsf{Id}[\sigma^1]\times \mathsf{Id}[\sigma^4]\times \mathsf{Id}[\sigma^3]\times \mathsf{Id}[\sigma^1]\times g\times \mathsf{Id}[\sigma^2]\big)\\
&\quad \circ\big\langle p_1, p_2, p_3, p_4, \langle p_5, p_6\rangle, p_7\big\rangle\\
&\quad \circ\big\langle p_1, p_{10}, p_8, p_1, p_2, p_8, p_4\big\rangle\\
&\quad \circ\big\langle p_1, p_2, p_3, p_4, p_5, p_6, p_7, h\langle p_4, p_4, p_{10}\rangle, p_9, p_1, p_{11}\big\rangle\\
&= \mathsf{Arr}\left(e, V\cup W, \sigma^5\right)\\
&\quad \circ\big\langle p_1, p_2, p_3, p_4, p_5, p_6, p_7, \mathsf{Arr}\big(u, \big(V \setminus \{x_2^3\}\big)\cup W\big), p_9, p_{10}, p_{11}\big\rangle
\end{aligned}
$$

4.4.6. *Remark.* In Examples 4.4.1 and 4.4.5 we may define a map

$$A(e,u) : \prod \mathsf{TypeList}\big[\big(V \setminus \{x_2^3\}\big)\cup W\big] \to \prod \mathsf{TypeList}\left[V\cup W\right]$$

as follows. Choose $I \in 1\,..\,\mathsf{Length}\left[V\cup W\right]$ such that

$$\big(\mathsf{TypeList}[V\cup W]\big)_I = x_2^3$$

We next define for all $i \in 1\,..\,\mathsf{Length}\left[V\cup W\right]$

$$(A(e,u))_i = \mathsf{Proj}[i]$$

and

$$(A(e,u))_I = \mathsf{Arr}\big(u, \big(V \setminus \{x_2^3\}\big)\cup W, \sigma^5\big)$$

With this definition in the previous two examples we have

$$\mathsf{Arr}\big[\big(e, V, \sigma^5\big)\left[x_2^3 \leftarrow \big(u, W, \sigma^3\big)\right]\big] = \mathsf{Arr}\left[e, V\cup W, \sigma^5\right] \circ A(e,u)$$

This is again a general feature that the following discussion of substitution is intended to capture.

4.5. **Substitution.** As terms are defined recursively, substitution may be defined either by structural recursion or, in view of Examples 4.4.1 and 4.4.5, using composition. These two ways of defining substitution are convenient for different purposes. Here we establish the equivalence of the two procedures.

4.5.1. *Recursive definition.* Suppose (u, W, τ) is to be substituted for x in $t = (e, V, \sigma)$ where we assume that $x \in V$ and $\mathsf{Type}[x] = \tau$. We may define this recursively as follows: If $t = (x, V, \tau)$, then

$$t\big[x \leftarrow (u, W, \tau)\big] := \big(u, (V \setminus \{x\}) \cup W, \tau\big)$$

If $t = (f(e_1, \ldots, e_n), V, \tau)$, then

$$t\big[x \leftarrow (u, W, \tau)\big] := \big(f\,(e_1[x \leftarrow u], \ldots, e_n[x \leftarrow u])\,, (V \setminus \{x\}) \cup W, \sigma\big)$$

Note that the added complication here is owing to the fact that the list of variables can be independently specified and that we have tacitly assumed the usual recursive definition of the substitution of one expression in the place of a free variable in another. The above serves as a basis for the recursive definition of the arrow corresponding to t once the substitution has been made.

If $t = (x, V, \tau)$, then

$$\mathsf{Arr}\big[x \leftarrow (u, W, \tau)\big] := \mathsf{Arr}\big[u, (V \setminus \{x\}) \cup W, \tau\big]$$

If $t = (f(e_1, \ldots, e_n), V, \tau)$, then

$$\mathsf{Arr}\big[x \leftarrow (u, W, \tau)\big] := \mathsf{Arr}\big[f\,(e_1[x \leftarrow u], \ldots, e_n[x \leftarrow u])\,, (V \setminus \{x\}) \cup W, \sigma\big]$$

4.5.2. *Direct definition.* The alternative way suggested by Examples 4.4.1 and 4.4.5 is to define $\mathsf{Arr}\,[e[x \leftarrow u], (V \setminus \{x\})\,, \sigma]$ directly, given

$$\mathsf{Arr}\,[e, V, \sigma] = Q[e]I[e]D[(e, V, \sigma)] \tag{10}$$

$$\mathsf{Arr}\,[u, W, \tau] = Q[u]I[u]D[(u, W, \tau)] \tag{11}$$

In view of Lemma 4.4.2, we may suppose $\mathsf{Arr}\,[e, V \cup W, \sigma] = Q[e]I[e]D[(e, V \cup W, \sigma)]$, that is that $\mathsf{Arr}\,[e, V \cup W, \sigma]$ differs from $\mathsf{Arr}\,[e, V, \sigma]$ only in the D-composand of the arrow.

We have

$$\mathsf{Dom}\big[D\,[(e, V \cup W, \sigma)]\big] = \prod (V \cup W)$$

Choose $I \in 1\,..\,\mathsf{Length}[V \cup W]$ such that $(V \cup W)_I = x$. Define an arrow

$$A := A\big((e, V, \sigma)\,, (u, W, \tau)\big) : \prod \big((V \setminus \{x\}) \cup W\big) \to \prod (V \cup W)$$

as follows: For all $i \in \big(1\,..\,\mathsf{Length}\,(V \cup W)\big) \setminus \{I\}$, $A_i = \mathsf{Proj}[i]$ and

$$A_I = \mathsf{Arr}\,[u, (V \setminus \{x\}) \cup W, \tau]$$

Note that $\prod\big((V \setminus \{x\}) \cup W\big)$ and $\prod (V \cup W)$ can differ in at most one factor depending on whether $x \in W$ or not.

Finally, we define

$$\mathsf{Arr}\big[e[x \leftarrow u], (V \setminus \{x\}) \cup W, \tau\big] = \mathsf{Arr}\,[e, V \cup W, \sigma] \circ A\big((e, V, \sigma)\,, (u, W, \tau)\big)$$

We have given two methods of obtaining the arrow corresponding to the term for which substitution has been made. It remains to be seen that these two methods give the same arrow.

Proof by structural induction.

Base case: $t = (x, V, \sigma)$ and $\sigma = \tau$. We note that

$$\begin{aligned}\mathsf{Arr}[t] &= Q[x]\circ I[x]\circ D[t]\\ &= \mathsf{Id}[\sigma]\circ \mathsf{Id}[\sigma]\circ\mathsf{proj}[I] \quad (\text{where } (V\cup W)_I = x)\\ &= \mathsf{proj}[I]\end{aligned}$$

From the direct definition we have

$$\begin{aligned}\mathsf{Arr}\big[x[x\leftarrow u], (V\setminus\{x\})\cup W, \sigma\big] &= \mathsf{Arr}\,[x, V\cup W, \tau]\circ A\big((x,V,\tau),(u,W,\tau)\big)\\ &= \mathsf{proj}[I]\circ A\big((x,V,\tau),(u,W,\tau)\big) = \mathsf{Arr}\big(u, (V\setminus\{x\})\cup W, \tau\big)\end{aligned}$$

by the definition of $A\big((x,V,\tau),(u,W,\tau)\big)$, which agrees with the recursive definition. Induction step: $t = (f(e_1,\dots,e_n), V, \sigma)$, where $\mathsf{Outp}[f] = \sigma$ and for all $i\in 1\,..\,n$, $\mathsf{Outp}[e_i] = \gamma_i$.

We note that, if we define

$$t' := (f(e_1,\dots,e_n), V', \sigma)$$

where $\mathsf{VarSet}\,[\mathsf{Expr}[t]]\subseteq V'$, then

$$\begin{aligned}\mathsf{Arr}[t'] &= f\circ\left(\prod_{1\in 1..n} Q[e_i]\right)\circ\langle I[e_1],\dots,I[e_n]\rangle\circ D[t']\\ &= f\circ\big\langle Q[e_1]I[e_1]D[t'],\dots,Q[e_n]I[e_n]D[t']\big\rangle\\ &= f\circ\big\langle\mathsf{Arr}\,[e_1,V',\gamma_1],\dots,\mathsf{Arr}\,[e_n,V',\gamma_n]\big\rangle\end{aligned}$$

Although this last equality is obvious, a complete proof may require a lemma.

By induction hypothesis, we have, for all $i\in 1\,..\,n$,

$$\mathsf{Arr}\big[e_i[x\leftarrow u], (V\setminus\{x\})\cup W, \gamma_i\big] = \mathsf{Arr}\big[e_i, V\cup W, \gamma_i)\big]\circ A\big((e_i,V,\gamma_i),(u,W,\tau)\big)$$

We next note that the direct definition yields

$$\begin{aligned}\mathsf{Arr}[t] = \mathsf{Arr}&\big[f(e_1,\dots,e_n)[x\leftarrow u], (V\setminus\{x\}), \sigma\big]\\ &= \mathsf{Arr}\,[f(u_1\dots,u_n), V\cup W, \sigma]\circ A\big((e,V,\sigma),(u,W,\tau)\big)\\ &\qquad\qquad\text{where } e := f(e_1,\dots,e_n)\\ &= f\circ\big\langle\mathsf{Arr}\,[e_1,V\cup W,\gamma_1],\dots,\mathsf{Arr}\,[e_n,V\cup W,\gamma_n]\big\rangle\\ &\qquad\circ A\big((e,V,\sigma),(u,W,\tau)\big)\\ &= f\circ\big\langle\mathsf{Arr}\,[e_1,V\cup W,\gamma_1]\circ A\big((e,V,\sigma),(u,W,\tau)\big),\\ &\qquad\dots,\mathsf{Arr}\,[e_n,V\cup W,\gamma_n]\circ A\big((e,V,\sigma),(u,W,\tau)\big)\big\rangle\\ &= f\circ\big\langle\mathsf{Arr}\,[e_1,V\cup W,\gamma_1]\circ A\big((e_1,V,\gamma_1),(u,W,\tau)\big),\\ &\qquad\dots,\mathsf{Arr}\,[e_n,V\cup W,\gamma_n]\circ A\big((e_n,V,\gamma_n),(u,W,\tau)\big)\big\rangle\end{aligned}$$

Again a complete proof of the last equality may require a lemma. Continuing, we have that

$$\begin{aligned}\mathsf{Arr}[t] = f \circ \big\langle \mathsf{Arr}\,[e_1[e \leftarrow u], (V \setminus \{x\}) \cup W, \gamma_1]\,, \ldots\,, \mathsf{Arr}\,[e_n[e \leftarrow u], (V \setminus \{x\}) \cup W, \gamma_n]\big\rangle \\ \text{(by induction hypothesis)} \\ = \mathsf{Arr}\big[f\,(e_1[x_1 \leftarrow u], \ldots\,, e_n[x_n \leftarrow u])\,, (V \setminus \{x\}) \cup W, \sigma\big] \\ = \mathsf{Arr}\big[t\,[x \leftarrow (u, W, \tau)]\big]\end{aligned}$$

which is what we get from the recursive definition. This completes the proof of the equivalence of the two definitions.

Later, we shall use the equivalence of these two methods of obtaining the arrow corresponding to the term in which substitution has been made. To facilitate reference we record this in the form of a lemma.

4.5.3. *Lemma. Let* $t\, := (e, V, \sigma)$ *and* $t'\, := (u, W, \tau)$ *be terms, suppose* $x \in V$ *and suppose* $\mathsf{Outp}[u] = \mathsf{Type}[x]$ *so that* u *may be substituted for* x*. Then there exists an arrow*

$$A\, := A\big((e, V, \sigma)\,, (u, W, \tau)\big) : \prod\big((V \setminus \{x\}) \cup W\big) \to \prod\,(V \cup W)$$

so that

$$\mathsf{Arr}\big[t\,[x \leftarrow u]\,, (V \setminus \{x\}) \cup W, \tau\big] = \mathsf{Arr}\big[(e, V \cup W, \sigma)\big] \circ A$$

5. Rules of inference of MSEL

In this section we show how the rules of inference of multisorted equational logic can be codified into our present system. This is a two-step process. First, we show that for each rule of inference the pair of equal arrows corresponding to the conclusion of the rule of inference can be constructed using the rules of construction of graph-based logic [Bagchi and Wells, 1996] from the single arrow or the product of the equal pairs of arrows that form the hypothesis of that rule of inference. Next, we exhibit the construction as an actual factorization as defined in [Bagchi and Wells, 1996], where the nodes and arrows appearing in the positions corresponding to the various labels on the diagram (12) are the appropriate instances of the hypothesis, claim, workspace and so on for the rule in question.

(12)

$$\begin{tikzcd} & \mathsf{hyp} \arrow[d, "\mathsf{claimcon}"] \\ \mathsf{claim} \arrow[ur, "\mathsf{verif}"] \arrow[r, "\mathsf{hypcon}"'] & \mathsf{wksp} \end{tikzcd}$$

While some of these are done in detail some others are not. For our purposes, it is enough to prove that a codification as an actual factorization in $\mathsf{SynCat}\big[\mathbf{FinProd}, \ulcorner F \urcorner\big]$

(as defined in Section 4) is possible. In general, this may be done in more than one way. Symmetry and reflexivity are treated separately. Transitivity, concretion, abstraction and substitutivity are all treated in Section 5.7, as they all are special instances of a worked-out example in [Bagchi and Wells, 1996].

5.1. **Reflexivity.** The equational rule of inference is

$$\frac{}{h =_V h}$$

5.1.1. *Translation as a construction.* Translated into the present context, as an instance of the rule of construction REF [Bagchi and Wells, 1996], this is represented as

$$\text{REF} \qquad \frac{A \xrightarrow{f} B}{A \underset{f}{\overset{f}{\rightrightarrows}} B}$$

where

$$\begin{aligned} f &:= \mathsf{Arr}\,[h, V, \mathsf{Type}[V]] \\ A &:= \mathsf{Dom}\big[\mathsf{Arr}\,[h, V, \mathsf{Type}[V]]\big] \\ B &:= \mathsf{Cod}\big[\mathsf{Arr}\,[h, V, \mathsf{Type}[V]]\big] \end{aligned}$$

This concludes the first step.

5.1.2. *Expression as actual factorization.* The corresponding actual factorization:

$$\begin{tikzcd} & (\mathsf{ar} \times \mathsf{ar})^{\langle f,f\rangle} \arrow[d, "\mathsf{proj}_1"] \\ \mathsf{ar}^f \arrow[ur, "\Delta"] \arrow[r, "\mathsf{id}"'] & \mathsf{ar}^f \end{tikzcd}$$

Note that one can also use proj_2 as *claimcon*. This factorization actually occurs in $\mathsf{SynCat}[\mathbf{Cat}, F]$ and is inherited by $\mathsf{SynCat}[\mathbf{FinProd}, F]$. A similar remark is true of the constructions for symmetry and transitivity.

5.2. **Symmetry.** Although in [Bagchi and Wells, 1996], we did not use a rule of construction corresponding to symmetry, we shall record an actual factorization for this to facilitate later discussion (in this section) on proofs as actual factorizations. The rule in equational deduction is

$$\frac{e =_V e'}{e' =_V e}$$

We define

$$\begin{aligned} f \ &:= \mathsf{Arr}\,[e, V, \mathsf{Type}[V]] \\ f' \ &:= \mathsf{Arr}\,[e', V, \mathsf{Type}[V]] \end{aligned}$$

then the actual factorization is as exhibited below:

$$\begin{tikzcd} & (\mathsf{ar}\times\mathsf{ar})^{\langle g,f\rangle} \arrow[d, "\langle \mathsf{Proj}[2],\mathsf{Proj}[1]\rangle"] \\ (\mathsf{ar}\times\mathsf{ar})^{\langle f,g\rangle} \arrow[ur, "\langle \mathsf{Proj}[2],\mathsf{Proj}[1]\rangle"] \arrow[r, "\mathsf{id}\times\mathsf{id}"'] & (\mathsf{ar}\times\mathsf{ar})^{\langle f,g\rangle} \end{tikzcd}$$

5.3. **Transitivity.** The equational rule of inference is

$$\frac{e =_V e' \quad e' =_V e''}{e =_V e''}$$

For the first step we define

$$\begin{aligned} f : D \to C \ &:= \mathsf{Arr}\,[e, V, \mathsf{Type}[V]] \\ g : D \to C \ &:= \mathsf{Arr}\,[e', V, \mathsf{Type}[V]] \\ h : D \to C \ &:= \mathsf{Arr}\,[e'', V, \mathsf{Type}[V]] \end{aligned}$$

Note that f, g, and h have the same domain and the same codomain as e, e', and e'' have the same type and as V is the same in each of the terms exhibited below:

$$\text{TRANS} \quad \frac{D \underset{g}{\overset{f}{\rightrightarrows}} C \qquad D \underset{h}{\overset{g}{\rightrightarrows}} C}{D \underset{h}{\overset{f}{\rightrightarrows}} C}$$

for all objects D and C and all arrows $f, g, h : D \to C$ of $\mathsf{CatTh}\big[\mathbf{FinLim}, \mathsf{Sk}[\mathcal{S}]\big]$

The corresponding actual factorization is provided in Section 5.7.

5.4. **Concretion.** In this case the equational inference rule reads

Given a set V of typed variables, $x \in V$ and an equation $e =_V e'$ such that $x \in V \setminus (\mathsf{VarSet}[e] \cup \mathsf{VarSet}[e'])$, and given that $\mathsf{Type}[x]$ is inhabited,

$$\frac{e =_V e'}{e =_{V\setminus\{x\}} e'}$$

We define $\tau := \mathsf{Type}[e] = \mathsf{Type}[e']$ and $\sigma := \mathsf{Type}[x]$, and

$$\begin{aligned} f : P \to \tau \; &:= \mathsf{Arr}\,[e, V, \tau] \\ f' : P \to \tau \; &:= \mathsf{Arr}\,[e', V, \tau] \\ g : Q \to \tau \; &:= \mathsf{Arr}\,[e, V\setminus\{x\}, \tau] \\ g' : Q \to \tau \; &:= \mathsf{Arr}\,[e', V\setminus\{x\}, \tau] \end{aligned}$$

Using Lemmas 4.4.2 and 4.4.3, we may choose a map

$$h : \prod (V \setminus \{x\}) \to \prod V$$

such that

$$\begin{aligned} g &= f \circ h \\ g' &= f' \circ h \end{aligned}$$

Thus coded as arrows, the rule reads

$$\frac{f = f'}{f \circ h = f' \circ h}$$

5.5. **Abstraction.** In this case the equational rule of inference reads

Given a set of typed variables and $x \in \mathsf{Vbl}[S] \setminus V$,

$$\frac{e =_V e'}{e =_{V\cup\{x\}} e'}$$

We define $\tau := \mathsf{Type}[e] = \mathsf{Type}[e']$ and $\sigma := \mathsf{Type}[x]$, and

$$\begin{aligned} f : P \to \tau \; &:= \mathsf{Arr}\,[e, V, \tau] \\ f' : P \to \tau \; &:= \mathsf{Arr}\,[e', V, \tau] \\ g : Q \to \tau \; &:= \mathsf{Arr}\,[e, V \cup \{x\}, \tau] \\ g' : Q \to \tau \; &:= \mathsf{Arr}\,[e', V \cup \{x\}, \tau] \end{aligned}$$

Using Lemmas 4.4.2 and 4.4.3, we may choose a map

$$h : \prod (V \cup \{x\}) \to V$$

such that $g = f \circ h$ and $g' = f' \circ h$. Thus coded as arrows the rule reads

$$\frac{f = f'}{f \circ h = f' \circ h}$$

5.6. **Substitutivity.** Given a set V of typed variables, $x \in V$, and expressions u and u' for which $\mathsf{Type}[x] = \mathsf{Type}[u] = \mathsf{Type}[u']$ and $\mathsf{Type}[e] = \mathsf{Type}[e'] = \tau$,

$$\frac{e =_V e' \quad u =_W u'}{e[x \leftarrow u] =_{V\setminus\{x\}\cup W} e'[x \leftarrow u']}$$

We already have the representations

$$\begin{aligned}
f \ &:= \mathsf{Arr}\,[e, V, \mathsf{Type}[e]] = Q[e]I[e]D[e] \\
f' \ &:= \mathsf{Arr}\,[e', V, \mathsf{Type}[e]] = Q[e']I[e']D[e'] \\
g \ &:= \mathsf{Arr}\,[u, W, \mathsf{Type}[u]] = Q[u]I[u]D[u] \\
g' \ &:= \mathsf{Arr}\,[u', W, \mathsf{Type}[u]] = Q[u']I[u']D[u'] \\
h \ &:= \mathsf{Arr}\big[e[x \leftarrow u], (V \setminus \{x\}) \cup W, \tau\big] \\
h' \ &:= \mathsf{Arr}\big[e'[x \leftarrow u'], (V \setminus \{x\}) \cup W, \tau\big]
\end{aligned}$$

In view of Lemma 4.5.3, we may choose an arrow

$$A : \prod\big((V \setminus \{x\}) \cup W\big) \to \prod (V \cup W)$$

and

$$A' : \prod\big((V \setminus \{x\}) \cup W\big) \to \prod (V \cup W)$$

for which $h = f \circ A$ and $h' = f' \circ A'$. Note that A and A' are equal, as f and f' and g and g' are (refer to the definition of A in Lemma 4.5.3).

Thus coded in terms of arrows, the rule reads

$$\frac{f = f' \qquad A = A'}{f \circ A = f' \circ A'}$$

5.7. **Transitivity, concretion, abstraction and substitutivity as actual factorizations.** We recall the following proposition in [Bagchi and Wells, 1996]:

5.7.1. *Proposition. In any category, given the diagram*

(13)
$$\begin{tikzcd}
A \arrow[r, "h"] \arrow[d, "f"'] & B \arrow[d, "k"] \\
C \arrow[r, "g"'] \arrow[ur, "x"] & D
\end{tikzcd}$$

if the two triangles commute, then so does the outside square.

It is shown there how the proof may be viewed as an actual factorization.

Transitivity may be viewed as a special case of this once equations are interpreted as commutative diagrams as shown in Diagram (14):

(14)
$$\begin{tikzcd} D \arrow[r, "h"] \arrow[d, "\mathsf{Id}[D]"'] & C \arrow[d, "\mathsf{Id}[C]"] \\ D \arrow[ur, "g"] \arrow[r, "f"'] & C \end{tikzcd}$$

The fact that the two triangles commute means that $h = g$ and $g = f$. That the outside square commutes means that $h = f$.

In view of Lemmas 4.4.2 and 4.4.3, concretion and abstraction can be seen to be special cases of the following: For every pair of formally equal arrows $f, f' : D \to C$ and for every $h : E \to D$, $f \circ h$ and $f' \circ h$ are formally equal. This can also be realized as a special case of the commutativity of Diagram (13), with choices as shown: :

(15)
$$\begin{tikzcd} E \arrow[r, "h"] \arrow[d, "h"'] & D \arrow[d, "f"] \\ D \arrow[ur, "\mathsf{Id}[D]"] \arrow[r, "g"'] & C \end{tikzcd}$$

Particular choices for h yield concretion and abstraction.

In substitutivity, in view of Lemma 4.4.3, we have the following in terms of arrows: For every pair of formally equal arrows $f, f' : D \to C$, and for every pair of formally equal arrows $A, A' : E \to D$, $f \circ A$ and $f' \circ A'$ are formally equal. This is also a special case of Diagram (13) as shown below:

(16)
$$\begin{tikzcd} E \arrow[r, "A"] \arrow[d, "A'"'] & D \arrow[d, "f"] \\ D \arrow[ur, "\mathsf{Id}[D]"] \arrow[r, "f'"'] & C \end{tikzcd}$$

On the basis of the preceding discussion we conclude that we may make choices for all nodes and arrows in the diagram

$$\begin{tikzcd} & \mathit{hyp} \arrow[d, "\mathit{claimcon}"] \\ \mathit{claim} \arrow[ur, "\mathit{verif}"] \arrow[r, "\mathit{hypcon}"'] & \mathit{wksp} \end{tikzcd}$$

so that the actual factorization in $\mathsf{SynCat}[\mathbf{FinProd}, F]$ codes transitivity, concretion, abstraction and substitutivity respectively.

5.7.2. *Remark.* Our goal in this section is to produce for every equational deduction for every equation a corresponding actual factorization. However, in order to do this we need to put equational deductions into some normal form to allow easy translation. We shall also need to use certain operations on actual factorizations. We record these below as various lemmas.

5.7.3. *Lemma. Every two actual factorizations in any syntactic category*

$$\begin{tikzcd} & C_i \arrow[d, "c_1"] \\ H \arrow[ur, "u_1"] \arrow[r, "h_1"'] & W_1 \end{tikzcd} \qquad \begin{tikzcd} & C \arrow[d, "c_2"] \\ C_i \arrow[ur, "u_2"] \arrow[r, "h_2"'] & W_2 \end{tikzcd}$$

with the above labels can be pasted together to yield a single actual factorization with labels as shown:

$$\begin{tikzcd} & C \arrow[d, "c"] \\ H \arrow[ur, "u"] \arrow[r, "h"'] & W \end{tikzcd}$$

Proof. As every node in a syntactic category (or $\mathsf{SynCat}[E, F]$) is the vertex of a limit cone over some diagram in E, we may choose

$$\Delta_1 = \mathsf{BsDiag}[W_1]$$
$$\Delta_2 = \mathsf{BsDiag}[W_2]$$
$$\Delta_i = \mathsf{BsDiag}[C_i]$$

to get the following in the category of diagrams of E:

$$\begin{tikzcd} & \Delta_1 \arrow[d, "\alpha_1"] \\ \Delta_2 \arrow[r, "\alpha_2"'] & \Delta_i \end{tikzcd}$$

where α_1 and α_2 are the morphisms of diagrams that give rise to c_1 and h_2. As the category of diagrams in a category is small complete, we may form the pullback as shown:

$$\begin{tikzcd} \Delta \arrow[r, "\beta_1"] \arrow[d, "\beta_2"'] & \Delta_1 \arrow[d, "\alpha_1"] \\ \Delta_2 \arrow[r, "\alpha_2"'] & \Delta_i \end{tikzcd} \tag{17}$$

Taking the limit over the diagrams corresponding to the vertices in (17) and using the lemmas in Section 3 of [Bagchi and Wells, 1996], we get the following diagram in $\mathsf{SynCat}[E, F]$:

$$\begin{tikzcd} C_i \arrow[r, "h_2"] \arrow[d, "c_1"'] & W_2 \arrow[d, "d_2"] \\ W_1 \arrow[r, "d_1"'] & W \end{tikzcd}$$

This gives the following diagram in $\mathsf{SynCat}[E, F]$:

$$\begin{tikzcd} & & C \arrow[d, "c_2"] \\ & C_i \arrow[ur, "u_2"] \arrow[r, "c_2"'] \arrow[d, "c_1"] & W_2 \arrow[d, "d_2"] \\ H \arrow[ur, "u_1"] \arrow[r, "h_1"'] & W_1 \arrow[r, "d_1"'] & W \end{tikzcd}$$

The lemma follows by setting

$$\begin{aligned} u &:= u_2 \circ u_1 \\ c &:= d_2 \circ c_2 \\ h &:= d_1 \circ h_1 \end{aligned}$$

□

5.7.4. *Remark.* The above is an analogue of getting the deduction $\frac{E_1}{E}$ given the deduction $\frac{E_1}{E_2}$ and $\frac{E_2}{E}$.

5.7.5. *Lemma. Given two actual factorizations in any syntactic category*

$$\begin{tikzcd} & C_1 \arrow[d, "c_1"] \\ H_1 \arrow[ur, "u_1"] \arrow[r, "h_1"'] & W_1 \end{tikzcd} \qquad \begin{tikzcd} & C_2 \arrow[d, "c_2"] \\ H_2 \arrow[ur, "u_2"] \arrow[r, "h_2"'] & W_2 \end{tikzcd}$$

we have the factorization

$$\begin{tikzcd} & C_1 \times C_2 \arrow[d, "c_1 \times c_2"] \\ H_1 \times H_2 \arrow[ur, "u_1 \times u_2"] \arrow[r, "h_1 \times h_2"'] & W_1 \times W_2 \end{tikzcd}$$

Proof. Omitted. □

5.8. **Normal forms for equational deductions.** We recall the usual recursive definition of a deduction of the equation E from the family of equations $(E_i \mid i \in 1\,..\,n)$. A deduction is a tree with E at the root and n nodes $(E_i \mid i \in 1\,..\,n)$ at level 0. For all $m \in 1\,..\,(n-1)$, the nodes are all obtained by the rules of inference (listed in Section 5) from either (a) any one or two nodes at strictly smaller levels (as the rules of inference that have a nonempty set for a premise have either one or two premises), or (b) the empty premise.

We assume given a deduction

$$D = \big((E_{lw} \mid w \in 0\,..\,W(l)) \mid i \in 0\,..\,L\big)$$

where l refers to the level in the deduction tree and $W(l)$ refers to the width of the deduction tree at level l. In addition: (a) $W(0) = n$, that is, the list of premises has n entries $(E_i \mid i \in 1\,..\,n)$, and (b) $W(L) = 1$ and has exactly one entry, namely E.

In order to construct an actual factorization corresponding to D, we first transform D into a normal form to facilitate coding. The normal form described below will involve considerable redundancy. We list below the relevant features of the normal form.

NF.1 For all $l \in 1\,..\,L$ and for all $w \in 0\,..\,W(l)$, the premises from which E_{lw} are deduced by one of the rules of inference in Section 5 appear at the immediately preceding level, that is level $l-1$. (Note that $l \in 1\,..\,L$, in other words we exclude the hypotheses at level 0.)

NF.2 For all $l \in 0\,..\,L$ and for all $w \in 0\,..\,W(l)$, E_{lw} is used exactly once for a deduction of some $E_{l+1,w}$ at the immediately following level.

NF.1 is achieved by carrying over every single E_{lw} at level l to the root level, using the rule $\frac{E}{E}$. NF.2 is achieved by repeating every hypothesis starting at level 0 with every equation in the list $(E_i \mid i \in 1\,..\,n)$ as many times as it is used in the body of the given deduction

$$\big((E_{lw} \mid w \in 0\,..\,W(l)) \mid i \in 0\,..\,L\big)$$

We shall give each such deduction the name

$$\mathsf{Ddcn}\big[(E_i \mid i \in 1\,..\,n)\,, E\big] = \big((E_{iw} \mid w \in 0\,..\,W(l)) \mid l \in 0\,..\,L\big)$$

We record the preceding discussion in the following Lemma.

5.8.1. *Lemma. Given any deduction of* E *from the family of equations* $(E_i' \mid i \in 1\,..\,n')$, *we may find an equivalent one (in normal form)*

$$\mathsf{Ddcn}\big[(E_i \mid i \in 1\,..\,n)\,, E\big] = \big((E_{ln} \mid w \in 0\,..\,W(l)) \mid l \in 0\,..\,L\big)$$

that has the following properties:

1. *For all* $l \in 1\,..\,L$ *and for all* $w \in 0\,..\,W(l)$, E_{lw} *follows from some members of the list* $(E_{l-1,w} \mid w \in 1\,..\,W(l-1))$ *or the empty premise using some rule of inference.*
2. *For all* $l \in 0\,..\,L$ *and for all* $w \in 0\,..\,W(L)$, E_{lw} *is used exactly once for deducing some* $E_{l+1,w}$ *in the next level using some rule of inference.*

5.8.2. *Theorem. Given any deduction* $\mathsf{Ddcn}\big[(E_i \mid e \in 1\,..\,n)\,, E\big]$ *of equational logic, we may construct an actual factorization in* $\mathsf{SynCat}[\textbf{FinProd}, \mathsf{F}]$ *that corresponds to it.*

Proof. In view of Lemma 5.8.1, we may, without loss of generality, assume that the given deduction is in normal form, that is,

$$\mathsf{Ddcn}\big[(E_i \mid i \in 1\,..\,n, E)\,, E\big] = \big((E_{lw} \mid w \in 0\,..\,W(l)) \mid l \in 0\,..\,L\big)$$

Because the deduction is in normal form for every pair of consecutive levels $(l, l+1) \in (0\,..\,l) \times (1\,..\,l)$ we may choose partitions of the index sets $0\,..\,W(l)$ and $0\,..\,W(l+1)$ with the following properties:

1. Both index sets have the same number $P(l, l+1)$ of parts. For all $p \in 0\,..\,P(l, l+1)$, we define p_l to be the family of equations in the pth part at level l and similarly p_{l+1} to be the family of equations in the pth part at level $l+1$.
2. For all $p \in 0\,..\,P(l, l+1)$, $\frac{p_l}{p_{l+1}}$ is an instance of some rule of inference; that is, the pair of levels $(l, l+1)$ may be rewritten as
$$\Big(\frac{p_l}{p_{l+1}} \mid p \in 0\,..\,P\,(l, l+1)\Big)$$

As we have already shown how each rule of inference can be coded as an actual factorization in Section 5.7 it follows that we may select, for each pair of levels $(l, l+1) \in (0\,..\,L) \times (1\,..\,L)$, a family (indexed by $p \in 0\,..\,P(l, l+1)$) of actual factorizations of the form

$$\begin{tikzcd} & C_{p,l} \arrow[d, "c_{p,l}"] \\ H_{p,l} \arrow[ur, "u_{p,l}"] \arrow[r, "h_{p,l}"'] & W_{p,l} \end{tikzcd} \tag{18}$$

in SynCat[**FinProd**, F], where each instance of (18) corresponds to
$$\frac{p_l}{p_{l+1}}$$
via the coding of inference rules.

Using Lemma 5.7.5, we may combine for all $l \in 0\,..\,(L-1)$ this family of actual factorizations into a single actual factorization

$$\begin{tikzcd} & C_l \arrow[d, "c_{p,l}"] \\ H_l \arrow[ur, "u_l"] \arrow[r, "h_{p,l}"'] & W_l \end{tikzcd}$$

where for all $x \in \{H, C, W, u, c, h\}$,
$$x_l = \prod_{p \in P(l,l+1)} x_{p,l}$$
Note that $P(l-1, l)$ and $P(l, l+1)$ may be different. However, for all $l \in 0\,..\,(l-1)$, C_l and H_{l+1} will be isomorphic because of associativity. For every l we define $\alpha_{l,l+1}$

to be the associativity isomorphism. We now have the following diagram.

$$\begin{tikzcd}
 & & & & & C_{L-1} \arrow[d, "c_{L-1}"] \\
 & & & & H_{L-1} \arrow[ur, "u^{l_1}"] \arrow[r, "h_{l_1}"'] & W_{L-1} \\
 & & & C_2 \arrow[ur, dotted, no head] \arrow[d, "c_2"] & & \\
 & & H_2 \arrow[ur, "u_2"] \arrow[r, "h_2"'] & W_2 & & \\
 & C_1 \arrow[ur, "\alpha_{1,2}"] \arrow[d, "c_1"] & & & & \\
H_1 \arrow[ur, "u_1"] \arrow[r, "h_1"'] & W_1 & & & &
\end{tikzcd}$$

Using Lemma 5.7.3, we can combine all of this into a single actual factorization

(19)
$$\begin{tikzcd}
 & C_{L-1} \arrow[d, "c'"] \\
H_1 \arrow[ur, "u'"] \arrow[r, "h'"'] & W_{L-1}
\end{tikzcd}$$

Finally we note that we have the commutative diagram (in $\mathsf{CatTh}[\mathbf{FinProd}, S]$)

(20)
$$\begin{tikzcd}
C_{L-1} \arrow[r, "\mathsf{proj}"] \arrow[d, "c_{L-1}"'] & C \arrow[d, "c_{C,W}"] \\
W_{L-1} \arrow[r, "\mathsf{proj}"'] & W
\end{tikzcd}$$

where

$$\begin{tikzcd} & C \arrow[d, "c_{C,W}"'] \\ H \arrow[r, "h_{C,W}"] & W \end{tikzcd}$$

is the actual factorization corresponding to the deduction of E from the corresponding premises in the partition in level $L-1$. We note that E is the only member of one of the parts of level L. Putting (19) and (20) together, we have

(21)
$$\begin{tikzcd} & C \arrow[d, "c"] \\ H_1 \arrow[ur, "u"] \arrow[r, "L"'] & W \end{tikzcd}$$

where

(22) $$u = \mathsf{proj} \circ u'$$

(23) $$h = \mathsf{proj} \circ h'$$

(24) $$c = c_{C,W}$$

Diagram (21) is the actual factorization corresponding to the given deduction. □

5.8.3. *Remark.* The above shows that every deduction in MSEL occurs as an actual factorization in $\mathsf{SynCat}[\mathbf{FinProd}, F]$.

5.8.4. *Remark.* Although we worked out the details for multisorted equational logic and $\mathsf{CatTh}[\mathbf{FinProd}, F]$, the method will work for any logical system that can be described as a constructor-space sketch. Thus, in general, we shall have some logical system L and a category $\mathsf{CatTh}[E_L, F]$ in which E_L is the kind of category in which the models of L are. For instance, if L is the typed λ-calculus, E_L would be $\mathbf{CCC}$, and if L is intuitionistic type theory, then E_L would be a constructor space for toposes.

Given any sound and complete deductive system for L, if we interpret terms as arrows and encode them in $\mathsf{CatTh}[E_L, F]$ as we have done here, then we conjecture that the method will show that all theorems of L can be realized as actual factorizations in $\mathsf{CatTh}[E_L, F]$. (Indeed it appears nearly obvious that this will happen if we know that L and E_L have equivalent models; a detailed proof, is of course necessary to clinch the matter.) In the examples of the preceding paragraphs, we might use the deductive systems formulated in [Lambek and Scott, 1986]. The method used here is quite general.

6. Acknowledgments

The diagrams were prepared using K. Rose's `xypic`.

References

[Bagchi and Wells, 1996] Atish Bagchi and Charles Wells. *Graph-based logic and sketches I: The general framework.* Available by web browser from `http://www.cwru.edu/1/class/mans/math/pub/wells/`, 1996.

[Barr and Wells, 1995] Michael Barr and Charles Wells. **Category Theory for Computing Science, second edition**. Prentice-Hall International Series in Computer Science. Prentice-Hall International, New York, 1995.

[Goguen and Meseguer, 1982] Joseph A. Goguen and José Meseguer. *Completeness of many-sorted equational logic.* Technical Report CSL-135, SRI International Computer Science Laboratory, 333 Ravenswood Ave., Menlo Park, CA 94025, USA, 1982.

[Lambek and Scott, 1986] Joachim Lambek and P. Scott. **Introduction to Higher Order Categorical Logic**, volume 7 of *Cambridge Studies in Advanced Mathematics.* Cambridge University Press, 1986.

[Power and Wells, 1992] A.J. Power and Charles Wells. *A formalism for the specification of essentially algebraic structures in 2-categories.* Mathematical Structures in Computer Science, **2**:1–28, 1992.

[Wells, 1990] Charles Wells. *A generalization of the concept of sketch.* Theoretical Computer Science, **70**:159–178, 1990.

Atish Bagchi, 226 West Rittenhouse Square, #702, Philadelphia, PA 19103
E-mail address: `atish@math.upenn.edu`

Charles Wells, Department of Mathematics, Case Western Reserve University, University Circle, Cleveland, OH 44106-7058, USA
E-mail address: `cfw2@po.cwru.edu`