The IFF Term Language Namespace (IFF-TRM)

The Context of Term Languages

trm

This document axiomatizes the abstract namespace for term languages. It has two subordinate documents for the axiomatization of objects and morphisms. The axiomatization in this document rigorously follows the Categorical Design Principle. Please see the core considerations document for the connections between quartets and natural transformations. Table 1 lists the terminology for term languages consisting of 49 concepts and 51 terms (two concepts are denoted by two equivalent terms). The seven terms in bold **red** are terms needed to express the structure of the Lawvere category. The eight terms in bold **green** are terms in the abstract syntax. The ten terms in bold **magenta** are terms used to express the term tuple coproduct structure. The four terms in bold **teal** are terms used to express the term monadic structure. All other terms are supporting – some are needed for the abstract syntax of tuples and some are needed to express the axiomatization.

Table 1: Category-theoretic Terminology for Term Languages

	Other	Functor	Natural Transformation
trm	**language**	**variable function** **case**	function-arity indication projection
		term	term-arity **element substitution** term-substitution
			embedding embedding-tuple
		indexed-term **function-tuple** term-tuple	indexed-term-projection1 indexed-term-projection2 singleton function-tuple-projection1 function-tuple-projection2 term-tuple-projection1 term-tuple-projection2
		indicia **tuple**	 **index arity** **composition identity**
		tuple-tuple	first second
		lawvere	
			initial counique
		indicia-pair **tuple-pair**	**indicia1 indicia2** cocone-diagram opvertex opfirst opsecond colimiting-cocone colimit = **binary-coproduct injection1 injection2** comediator = **pairing**
	monad	endofunctor	unit multiplication

There is a (large) category Trm-Lang of term *language*s. The object class of Trm-Lang is the class of all term languages. The morphism class of Trm-Lang is the class of all term language morphisms. Composition in Trm-Lang is composition of term language morphisms.

```
(1) (CAT$category language)
    (= (CAT$object language) trm.obj$object)
    (= (CAT$morphism language) trm.mor$morphism)
    (= (CAT$source language) trm.mor$source)
    (= (CAT$target language) trm.mor$target)
    (= (CAT$composable language) trm.mor$composable)
```

```
    (= (CAT$composition language) trm.mor$composition)
    (= (CAT$identity language) trm.mor$identity)
```

There are *variable* and *function* symbol functors from the (large) category of term languages to the (large) category of sets

var : Trm-Lang → Set

ftn : Trm-Lang → Set.

There is an arity natural transformation (Figure 1)

: *ftn* ⇒ *var* ∘ ℘ : Trm-Lang → Set,

whose ***L***$^{\text{th}}$ component is the arity function for a term language ***L***.

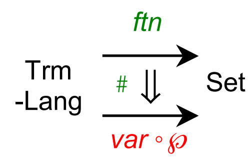

Figure 1: Arity Natural Transformation

```
(2) (FUNC$functor variable)
    (= (FUNC$source variable) language)
    (= (FUNC$target variable) set$set)
    (= (FUNC$object variable) trm.obj$variable)
    (= (FUNC$morphism variable) trm.mor$variable)

(3) (FUNC$functor function)
    (= (FUNC$source function) language)
    (= (FUNC$target function) set$set)
    (= (FUNC$object function) trm.obj$function)
    (= (FUNC$morphism function) trm.mor$function)

(4) (NAT$natural-transformation function-arity)
    (= (NAT$source-category function-arity) language)
    (= (NAT$target-category function-arity) set$set)
    (= (NAT$source-functor function-arity) function)
    (= (NAT$target-functor function-arity) indicia)
    (= (NAT$component function-arity) trm.obj$function-arity)
```

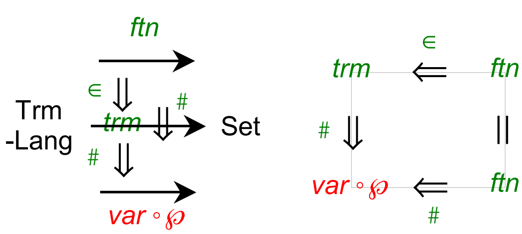

Figure 2: Function-Term Embedding

There is a *term* functor from the (large) category of languages to the (large) category of sets

trm : Trm-Lang → Set.

There is a function-as-term *embedding* natural transformation, whose ***L***$^{\text{th}}$ component is the function-as-term embedding function for a term language ***L***

∈ = *embed* : *ftn* ⇒ *trm* : Trm-Lang → Set.

There is a *term arity* natural transformation (Figure 2), whose ***L***$^{\text{th}}$ component is the term index function for a term language ***L***

= *arity* : *trm* ⇒ *var* ∘ ℘ : Trm-Lang → Set.

Term arity is compatible with function arity:

∈ • # = #

```
(5) (FUNC$functor term)
    (= (FUNC$source term) language)
    (= (FUNC$target term) set$set)
    (= (FUNC$object term) trm.obj.trm$term)
    (= (FUNC$morphism term) trm.mor.trm$term)

(6) (NAT$natural-transformation embedding)
    (= (NAT$source-category embedding) language)
    (= (NAT$target-category embedding) set$set)
```

```
    (= (NAT$source-functor embedding) function)
    (= (NAT$target-functor embedding) term)
    (= (NAT$component embedding) trm.obj.trm$embedding)

(7) (NAT$natural-transformation term-arity)
    (= (NAT$source-category term-arity) language)
    (= (NAT$target-category term-arity) set$set)
    (= (NAT$source-functor term-arity) term)
    (= (NAT$target-functor term-arity) indicia)
    (= (NAT$component term-arity) trm.obj.trm$arity)

    (= function-arity (NAT$vertical-composition [embedding term-arity]))
```

There is a function-tuple-as-term-*tuple embedding* natural transformation, whose $\boldsymbol{L}^{\text{th}}$ component is the function-tuple-as-term-tuple embedding function for a term language $\boldsymbol{L}$

$\in \otimes\, id : ftn \otimes tpl \Rightarrow trm \otimes tpl : \text{Trm-Lang} \rightarrow \text{Set}.$

The embedding tuple is effectively defined by two projection commutative diagrams:

$(\in \otimes\, id) \cdot \pi_1 = \pi_1 \cdot \in$ and $(\in \otimes\, id) \cdot \pi_2 = \pi_2$.

```
(8) (NAT$natural-transformation embedding-tuple)
    (= (NAT$source-category embedding-tuple) language)
    (= (NAT$target-category embedding-tuple) set$set)
    (= (NAT$source-functor embedding-tuple) function-tuple)
    (= (NAT$target-functor embedding-tuple) term-tuple)
    (= (NAT$component embedding-tuple) trm.obj.trm$embedding-tuple)

    (= (NAT$vertical-composition [embedding-tuple term-tuple-projection1])
       (NAT$vertical-composition [function-tuple-projection1 embedding]))
    (= (NAT$vertical-composition [embedding-tuple term-tuple-projection2])
       function-tuple-projection2)
```

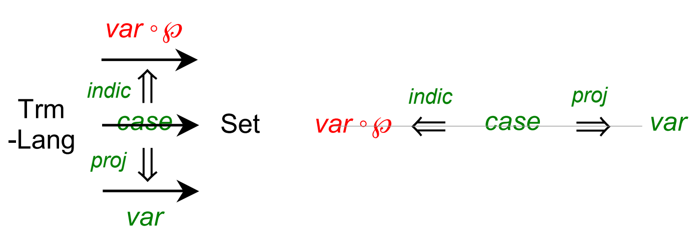

Figure 3: Case Indication and Projection Natural Transformations

There is a *case* functor from the (large) category of term languages to the (large) category of sets

$case : \text{Trm-Lang} \rightarrow \text{Set}.$

There are case *indication* and case *projection* natural transformations (Figure 3), whose $\boldsymbol{L}^{\text{th}}$ components are the case indication and case projection functions for a term language $\boldsymbol{L}$

$indic : case \Rightarrow var \circ \wp : \text{Trm-Lang} \rightarrow \text{Set},$

$proj : case \Rightarrow var : \text{Trm-Lang} \rightarrow \text{Set}.$

```
 (9) (FUNC$functor case)
     (= (FUNC$source case) language)
     (= (FUNC$target case) set$set)
     (= (FUNC$object case) trm.obj$case)
     (= (FUNC$morphism case) trm.mor$case)

(10) (NAT$natural-transformation indication)
     (= (NAT$source-category indication) language)
     (= (NAT$target-category indication) set$set)
     (= (NAT$source-functor indication) case)
     (= (NAT$target-functor indication) indicia)
     (= (NAT$component indication) trm.obj$indication)

(11) (NAT$natural-transformation projection)
```

```
(= (NAT$source-category projection) language)
(= (NAT$target-category projection) set$set)
(= (NAT$source-functor projection) case)
(= (NAT$target-functor projection) variable)
(= (NAT$component projection) trm.obj$projection)
```

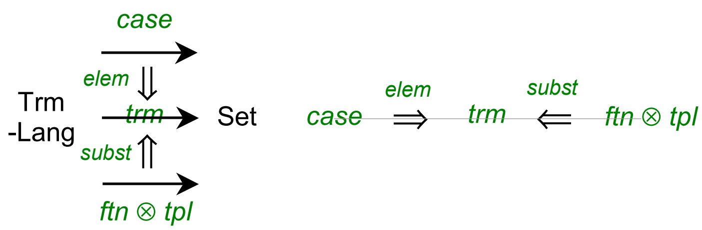

Figure 4: Element and Substitution Natural Transformations

There is an *element* natural transformation (Figure 4), whose $\boldsymbol{L}^{\text{th}}$ component is the element function for a term language $\boldsymbol{L}$

$elem : case \Rightarrow trm : \text{Trm-Lang} \rightarrow \text{Set}.$

There is a *function-tuple* functor from the (large) category of languages to the (large) category of sets

$ftn \otimes tpl : \text{Trm-Lang} \rightarrow \text{Set}.$

There are *function-tuple projection* natural transformations, whose $\boldsymbol{L}^{\text{th}}$ components are the function-tuple projection functions for a term language $\boldsymbol{L}$

$\pi_1 : ftn \otimes tpl \Rightarrow ftn : \text{Trm-Lang} \rightarrow \text{Set},$

$\pi_2 : ftn \otimes tpl \Rightarrow tpl : \text{Trm-Lang} \rightarrow \text{Set}.$

There is a *substitution* natural transformation (Figure 4), whose $\boldsymbol{L}^{\text{th}}$ component is the substitution function for a term language $\boldsymbol{L}$

$subst : ftn \otimes tpl \Rightarrow trm : \text{Trm-Lang} \rightarrow \text{Set}.$

```
(12) (NAT$natural-transformation element)
     (= (NAT$source-category element) language)
     (= (NAT$target-category element) set$set)
     (= (NAT$source-functor element) case)
     (= (NAT$target-functor element) term)
     (= (NAT$component element) trm.obj.trm$element)

(13) (FUNC$functor function-tuple)
     (= (FUNC$source function-tuple) language)
     (= (FUNC$target function-tuple) set$set)
     (= (FUNC$object function-tuple) trm.obj.trm$function-tuple)
     (= (FUNC$morphism function-tuple) trm.mor.trm$function-tuple)

(14) (NAT$natural-transformation function-tuple-projection1)
     (= (NAT$source-category function-tuple-projection1) language)
     (= (NAT$target-category function-tuple-projection1) set$set)
     (= (NAT$source-functor function-tuple-projection1) function-tuple)
     (= (NAT$target-functor function-tuple-projection1) function)
     (= (NAT$component function-tuple-projection1) trm.obj.trm$function-tuple-projection1)

(15) (NAT$natural-transformation function-tuple-projection2)
     (= (NAT$source-category function-tuple-projection2) language)
     (= (NAT$target-category function-tuple-projection2) set$set)
     (= (NAT$source-functor function-tuple-projection2) function-tuple)
     (= (NAT$target-functor function-tuple-projection2) tuple)
     (= (NAT$component function-tuple-projection2) trm.obj.trm$function-tuple-projection2)

(16) (NAT$natural-transformation substitution)
     (= (NAT$source-category substitution) language)
     (= (NAT$target-category substitution) set$set)
     (= (NAT$source-functor substitution) function-tuple)
     (= (NAT$target-functor substitution) term)
```

```
(= (NAT$component substitution) trm.obj.trm$substitution)
```

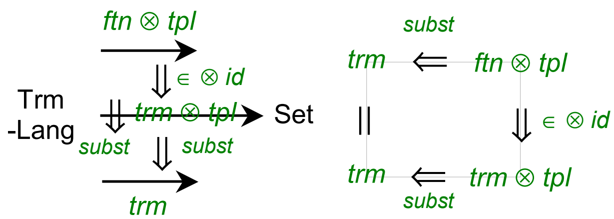

Figure 5: Term Substitution

There is a *term-tuple* functor from the (large) category of languages to the (large) category of sets

$trm \otimes tpl$: Trm-Lang → Set.

There are *term-tuple projection* natural transformations, whose $\boldsymbol{L}^{th}$ components are the term-tuple projection functions for a term language $\boldsymbol{L}$

$\pi_1 : trm \otimes tpl \Rightarrow trm$: Trm-Lang → Set,

$\pi_2 : trm \otimes tpl \Rightarrow tpl$: Trm-Lang → Set.

There is a *term substitution* natural transformation (Figure 5), whose $\boldsymbol{L}^{th}$ component is the term substitution function for a term language $\boldsymbol{L}$

$trm\text{-}subst : trm \otimes tpl \Rightarrow trm$: Trm-Lang → Set.

Term substitution is compatible with function substitution:

$(\in \otimes id) \cdot subst = subst$.

```
(17) (FUNC$functor term-tuple)
     (= (FUNC$source term-tuple) language)
     (= (FUNC$target term-tuple) set$set)
     (= (FUNC$object term-tuple) trm.obj.trm$term-tuple)
     (= (FUNC$morphism term-tuple) trm.mor.trm$term-tuple)

(18) (NAT$natural-transformation term-tuple-projection1)
     (= (NAT$source-category term-tuple-projection1) language)
     (= (NAT$target-category term-tuple-projection1) set$set)
     (= (NAT$source-functor term-tuple-projection1) term-tuple)
     (= (NAT$target-functor term-tuple-projection1) term)
     (= (NAT$component term-tuple-projection1) trm.obj.trm$term-tuple-projection1)

(19) (NAT$natural-transformation term-tuple-projection2)
     (= (NAT$source-category term-tuple-projection2) language)
     (= (NAT$target-category term-tuple-projection2) set$set)
     (= (NAT$source-functor term-tuple-projection2) term-tuple)
     (= (NAT$target-functor term-tuple-projection2) tuple)
     (= (NAT$component term-tuple-projection2) trm.obj.trm$term-tuple-projection2)

(20) (NAT$natural-transformation term-substitution)
     (= (NAT$source-category term-substitution) language)
     (= (NAT$target-category term-substitution) set$set)
     (= (NAT$source-functor term-substitution) term-tuple)
     (= (NAT$target-functor term-substitution) term)
     (= (NAT$component term-substitution) trm.obj.trm$term-substitution)

     (= (FUNC$composition [embedding-tuple term-substitution]) substitution)
```

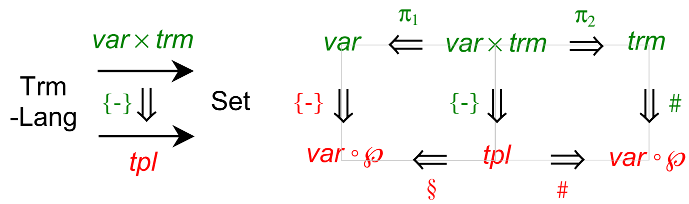

Figure 6: Singleton Natural Transformation

There is an *indexed-term* functor from the (large) category of languages to the (large) category of sets

$var \times trm$: Trm-Lang → Set.

There are *projection* variable and projection term natural transformations (Figure 6), whose L^{th} components are the projection variable and projection term functions for a term language L

$\pi_1 : var \times trm \Rightarrow var$: Trm-Lang → Set,

$\pi_2 : var \times trm \Rightarrow trm$: Trm-Lang → Set.

There is a *singleton* (term-as-tuple embedding) natural transformation (Figure 6), whose L^{th} component is the singleton function for a term language L

$\{\text{-}\} : var \times trm \Rightarrow tpl$: Trm-Lang → Set.

The singleton (term-as-tuple embedding) natural transformation commutes through tuple index and tuple arity:

$\{\text{-}\} \cdot \S = \pi_1 \cdot \{\text{-}\}$ and $\{\text{-}\} \cdot \# = \pi_2 \cdot \#$.

```
(21) (FUNC$functor indexed-term)
     (= (FUNC$source indexed-term) language)
     (= (FUNC$target indexed-term) set$set)
     (= (FUNC$object indexed-term) trm.obj.tpl$indexed-term)
     (= (FUNC$morphism indexed-term) trm.mor.tpl$indexed-term)
     (= indexed-term (FUNC$binary-product [variable term]))

(22) (NAT$natural-transformation indexed-term-projection1)
     (= (NAT$source-category indexed-term-projection1) language)
     (= (NAT$target-category indexed-term-projection1) set$set)
     (= (NAT$source-functor indexed-term-projection1) indexed-term)
     (= (NAT$target-functor indexed-term-projection1) variable)
     (= (NAT$component indexed-term-projection1) trm.obj.tpl$indexed-term-projection1)

(23) (NAT$natural-transformation indexed-term-projection2)
     (= (NAT$source-category indexed-term-projection2) language)
     (= (NAT$target-category indexed-term-projection2) set$set)
     (= (NAT$source-functor indexed-term-projection2) indexed-term)
     (= (NAT$target-functor indexed-term-projection2) variable)
     (= (NAT$component indexed-term-projection2) trm.obj.tpl$indexed-term-projection2)

(24) (NAT$natural-transformation singleton)
     (= (NAT$source-category singleton) language)
     (= (NAT$target-category singleton) set$set)
     (= (NAT$source-functor singleton) indexed-term)
     (= (NAT$target-functor singleton) tuple)
     (= (NAT$component singleton) trm.obj.tpl$singleton)

     (= (NAT$vertical-composition [singleton index])
        (NAT$vertical-composition [indexed-term-projection1 (set$singleton variable)]))
     (= (NAT$vertical-composition [singleton arity])
        (NAT$vertical-composition [indexed-term-projection2 term-arity]))
```

Lawvere

The *indicia* functor is the composition of the variable function and the power functor on sets:

$var \circ \wp$: Trm-Lang → Set → Set.

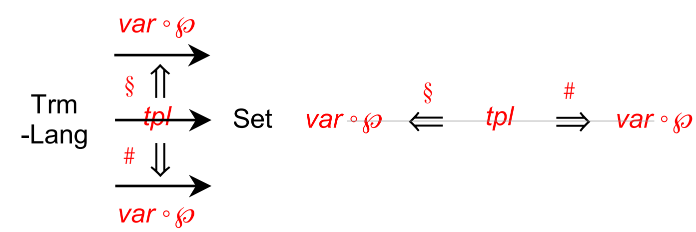

Figure 7: Tuple Index and Arity Natural Transformations

There is a term *tuple* functor from the (large) category of term languages to the (large) category of sets

tpl : Trm-Lang → Set.

There are tuple *index* and *arity* natural transformations (Figure 7),

§ = *index* : *tpl* ⇒ $var \circ \wp$: Trm-Lang → Set and

= *arity* : *tpl* ⇒ $var \circ \wp$: Trm-Lang → Set,

whose $\boldsymbol{L}^{\text{th}}$ components are the tuple index and tuple arity functions for a term language ***L***.

```
(25) (FUNC$functor indicia)
     (= (FUNC$source indicia) language)
     (= (FUNC$target indicia) set$set)
     (= (FUNC$object indicia) trm.obj$indicia)
     (= (FUNC$morphism indicia) trm.mor$indicia)
     (= indicia (FUNC$composition [variable set$power]))

(26) (FUNC$functor tuple)
     (= (FUNC$source tuple) language)
     (= (FUNC$target tuple) set$set)
     (= (FUNC$object tuple) trm.obj.tpl$tuple)
     (= (FUNC$morphism tuple) trm.mor.tpl$tuple)

(27) (NAT$natural-transformation index)
     (= (NAT$source-category index) language)
     (= (NAT$target-category index) set$set)
     (= (NAT$source-functor index) tuple)
     (= (NAT$target-functor index) indicia)
     (= (NAT$component index) trm.obj.tpl$index)

(28) (NAT$natural-transformation arity)
     (= (NAT$source-category arity) language)
     (= (NAT$target-category arity) set$set)
     (= (NAT$source-functor arity) tuple)
     (= (NAT$target-functor arity) indicia)
     (= (NAT$component arity) trm.obj.tpl$arity)
```

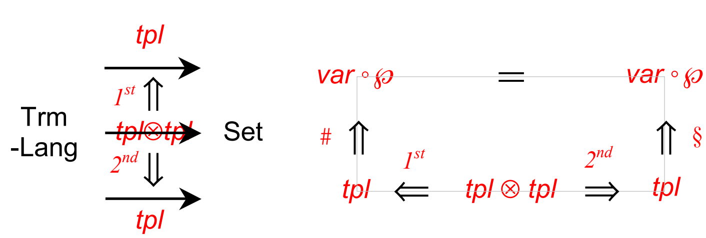

Figure 8: First and Second Term Tuple Natural Transformations

There is a *tuple-tuple* functor from the (large) category of languages to the (large) category of sets

tpl ⊗ *tpl* : Trm-Lang → Set.

There are two projection natural transformations (Figure 8), 1^{st} = *first* and 2^{nd} = *second*, whose $\boldsymbol{L}^{\text{th}}$ components are the first and second composable tuple pair projection functions for a term language ***L***

1^{st} : *tpl* ⊗ *tpl* ⇒ *tpl* : Trm-Lang → Set,

2^{nd} : *tpl* ⊗ *tpl* ⇒ *tpl* : Trm-Lang → Set.

The matching condition for composable tuple pairs is that the arity of the first is the index of the second:

1^{st} • # = 2^{nd} • §.

```
(29) (FUNC$functor tuple-tuple)
     (= (FUNC$source tuple-tuple) language)
     (= (FUNC$target tuple-tuple) set$set)
     (= (FUNC$object tuple-tuple) trm.obj.tpl$tuple-tuple)
     (= (FUNC$morphism tuple-tuple) trm.mor.tpl$tuple-tuple)

(30) (NAT$natural-transformation first)
     (= (NAT$source-category first) language)
     (= (NAT$target-category first) set$set)
     (= (NAT$source-functor first) tuple-tuple)
     (= (NAT$target-functor first) tuple)
     (= (NAT$component first) trm.obj.tpl$first)

(31) (NAT$natural-transformation second)
     (= (NAT$source-category second) language)
     (= (NAT$target-category second) set$set)
     (= (NAT$source-functor second) tuple-tuple)
     (= (NAT$target-functor second) tuple)
     (= (NAT$component second) trm.obj.tpl$second)

     (= (NAT$vertical-composition [first arity])
        (NAT$vertical-composition [second index]))
```

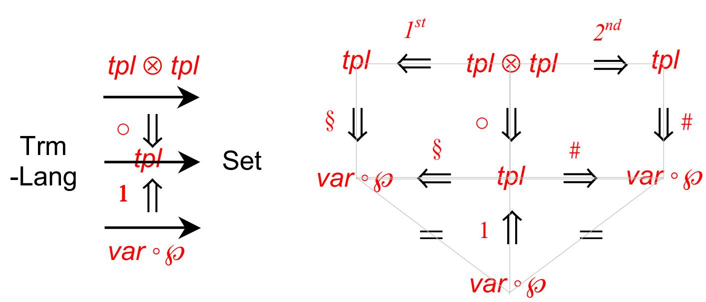

Figure 9: Composition and Identity Natural Transformations

There are *composition* and *identity* natural transformations (Figure 9), which represent composition and identity in the Lawvere categories and functors, and whose $\boldsymbol{L}^{\text{th}}$ components are the Lawverian composition and identity functions for a term language ***L***

○ : *tpl* ⊗ *tpl* ⇒ *tpl* : Trm-Lang → Set,

1 : *var* ∘ ℘ ⇒ *tpl* : Trm-Lang → Set.

The index of the composition of two composable term tuples is the index of the first tuple, and the arity of the composition is the arity of the second tuple:

○ • § = 1^{st} • § and ○ • # = 2^{nd} • #.

The index and arity of the identity tuple of an indicia is that indicia:

1 • § = *id* and 1 • # = *id*.

```
(32) (NAT$natural-transformation composition)
     (= (NAT$source-category composition) language)
     (= (NAT$target-category composition) set$set)
     (= (NAT$source-functor composition) tuple-tuple)
     (= (NAT$target-functor composition) tuple)
     (= (NAT$component composition) trm.obj.tpl$composition)
```

```
     (= (NAT$vertical-composition [composition index])
        (NAT$vertical-composition [first index]))
     (= (NAT$vertical-composition [composition arity])
        (NAT$vertical-composition [second arity]))

(33) (NAT$natural-transformation identity)
     (= (NAT$source-category identity) language)
     (= (NAT$target-category identity) set$set)
     (= (NAT$source-functor identity) indicia)
     (= (NAT$target-functor identity) tuple)
     (= (NAT$component identity) trm.obj.tpl$identity)

     (= (NAT$vertical-composition [identity index])
        (NAT$vertical-identity indicia))
     (= (NAT$vertical-composition [identity arity])
        (NAT$vertical-identity indicia))
```

The Lawvere construction is a functor

law : Trm-Lang → Cat

from the (large) category of term languages to the (large) category of small categories: for any term language $\boldsymbol{L}$, *law*($\boldsymbol{L}$) is a small category with coproducts and for any term language morphism $\boldsymbol{f} : \boldsymbol{L}_1 \rightarrow \boldsymbol{L}_2$, *law*($\boldsymbol{f}$) : *law*($\boldsymbol{L}_1$) → *law*($\boldsymbol{L}_2$) is a functor between small categories that preserves coproducts.

```
(34) (FUNC$functor lawvere)
     (= (FUNC$source lawvere) language)
     (= (FUNC$target lawvere) cat$category)
     (= (FUNC$object lawvere) trm.obj.tpl$lawvere)
     (= (FUNC$morphism lawvere) trm.mor.tpl$lawvere)
```

Term Tuple Coproducts

There is an initial and a counique natural transformation, whose L^{th} components are the initial and counique functions for a term language L

$\varnothing : \mathbf{1} \Rightarrow var \circ \wp$: Trm-Lang → Set,

$0 : var \circ \wp \Rightarrow tpl$: Trm-Lang → Set.

The index (source) of the counique tuple is the initial indicia pair, and the arity (target) is the given indicia:

$0 \cdot \S = ! \cdot \varnothing$ and $0 \cdot \# = 1$.

```
(35) (NAT$natural-transformation initial)
     (= (NAT$source-category initial) term-language)
     (= (NAT$target-category initial) set$set)
     (= (NAT$source-functor initial) ((FUNC$constant [term-language set$set]) cat$terminal))
     (= (NAT$target-functor initial) indicia)
     (= (NAT$component initial) trm.obj.col$initial)

(36) (NAT$natural-transformation counique)
     (= (NAT$source-category counique) language)
     (= (NAT$target-category counique) set$set)
     (= (NAT$source-functor counique) indicia)
     (= (NAT$target-functor counique) tuple)
     (= (NAT$component counique) trm.obj.col$counique)

     (= (NAT$vertical-composition [counique index])
        (NAT$vertical-composition [(NAT$vertical-terminal indicia) initial]))
     (= (NAT$vertical-composition [comediator arity])
        (NAT$vertical-identity indicia))
```

There is an indicia-pair functor from the (large) category of languages to the (large) category of sets

$var \circ \wp \times var \circ \wp$: Trm-Lang → Set.

```
(37) (FUNC$functor indicia-pair)
     (= (FUNC$source indicia-pair) language)
     (= (FUNC$target indicia-pair) set$set)
     (= (FUNC$object indicia-pair) trm.obj.col.coprd2$indicia-pair)
     (= (FUNC$morphism indicia-pair) trm.mor.col.coprd2$indicia-pair)
```

There are two projection natural transformations from indicia pairs, whose L^{th} components are the indicia projection functions for a term language L

$ind_1 : var \circ \wp \times var \circ \wp \Rightarrow var \circ \wp$: Trm-Lang → Set,

$ind_2 : var \circ \wp \times var \circ \wp \Rightarrow var \circ \wp$: Trm-Lang → Set.

```
(38) (NAT$natural-transformation indicia1)
     (= (NAT$source-category indicia1) language)
     (= (NAT$target-category indicia1) set$set)
     (= (NAT$source-functor indicia1) indicia-pair)
     (= (NAT$target-functor indicia1) indicia)
     (= (NAT$component indicia1) trm.obj.col.coprd2$indicia1)

(39) (NAT$natural-transformation indicia2)
     (= (NAT$source-category indicia2) language)
     (= (NAT$target-category indicia2) set$set)
     (= (NAT$source-functor indicia2) indicia-pair)
     (= (NAT$target-functor indicia2) indicia)
     (= (NAT$component indicia2) trm.obj.col.coprd2$indicia2)
```

There is a tuple-pair functor from the (large) category of languages to the (large) category of sets

$tpl \oplus tpl$: Trm-Lang → Set.

```
(40) (FUNC$functor tuple-pair)
     (= (FUNC$source tuple-pair) language)
     (= (FUNC$target tuple-pair) set$set)
     (= (FUNC$object tuple-pair) trm.obj.col.coprd2$tuple-pair)
     (= (FUNC$morphism tuple-pair) trm.mor.col.coprd2$tuple-pair)
```

There are two projection natural transformations from tuple pairs, called opfirst and opsecond, whose $\boldsymbol{L}^{\text{th}}$ components are the tuple pair projection functions for a term language $\boldsymbol{L}$

$$op1^{st} : tpl \oplus tpl \Rightarrow tpl : \text{Trm-Lang} \rightarrow \text{Set},$$

$$op2^{nd} : tpl \oplus tpl \Rightarrow tpl : \text{Trm-Lang} \rightarrow \text{Set}.$$

```
(41) (NAT$natural-transformation opfirst)
     (= (NAT$source-category opfirst) language)
     (= (NAT$target-category opfirst) set$set)
     (= (NAT$source-functor opfirst) tuple-pair)
     (= (NAT$target-functor opfirst) tuple)
     (= (NAT$component opfirst) trm.obj.col.coprd2$opfirst)

(42) (NAT$natural-transformation opsecond)
     (= (NAT$source-category opsecond) language)
     (= (NAT$target-category opsecond) set$set)
     (= (NAT$source-functor opsecond) tuple-pair)
     (= (NAT$target-functor opsecond) tuple)
     (= (NAT$component opsecond) trm.obj.col.coprd2$opsecond)
```

There are also two natural transformations from tuple pairs, called cocone-diagram and opvertex, which give the underling indicia-pair and opvertex components of a tuple pair, and whose $\boldsymbol{L}^{\text{th}}$ components are the cocone-diagram and opvertex functions for a term language $\boldsymbol{L}$

$$dgm : tpl \oplus tpl \Rightarrow var \circ \wp \times var \circ \wp : \text{Trm-Lang} \rightarrow \text{Set},$$

$$opvtx : tpl \oplus tpl \Rightarrow var \circ \wp : \text{Trm-Lang} \rightarrow \text{Set}.$$

The indicia1component of the cocone diagram is the index of the opfirst tuple, and the indicia2 component of the cocone diagram is the index of the opsecond tuple:

$$dgm \cdot ind_1 = op1^{st} \cdot \S \text{ and } dgm \cdot ind_2 = op2^{nd} \cdot \S.$$

The opvertex is the arity of both the opfirst and opsecond tuples:

$$opvtx = op1^{st} \cdot \# \text{ and } opvtx = op2^{nd} \cdot \#.$$

```
(43) (NAT$natural-transformation cocone-diagram)
     (= (NAT$source-category cocone-diagram) language)
     (= (NAT$target-category cocone-diagram) set$set)
     (= (NAT$source-functor cocone-diagram) tuple-pair)
     (= (NAT$target-functor cocone-diagram) indicia-pair)
     (= (NAT$component cocone-diagram) trm.obj.col.coprd2$cocone-diagram)

(44) (NAT$natural-transformation opvertex)
     (= (NAT$source-category opvertex) language)
     (= (NAT$target-category opvertex) set$set)
     (= (NAT$source-functor opvertex) tuple-pair)
     (= (NAT$target-functor opvertex) indicia)
     (= (NAT$component opvertex) trm.obj.col.coprd2$opvertex)

     (= (NAT$vertical-composition [cocone-diagram indicia1])
        (NAT$vertical-composition [opfirst index]))
     (= (NAT$vertical-composition [cocone-diagram indicia2])
        (NAT$vertical-composition [opsecond index]))
     (= opvertex (NAT$vertical-composition [opfirst arity]))
     (= opvertex (NAT$vertical-composition [opsecond arity]))
```

There is a colimiting cocone natural transformation from indicia pairs to tuple pairs, whose $\boldsymbol{L}^{\text{th}}$ component is the colimiting cocone function for a term language $\boldsymbol{L}$

$$\nabla : var \circ \wp \times var \circ \wp \Rightarrow tpl \oplus tpl : \text{Trm-Lang} \rightarrow \text{Set}.$$

The indicia pair of the colimiting cocone is the original pair:

$$\nabla \cdot dgm = 1.$$

```
(45) (NAT$natural-transformation colimiting-cocone)
     (= (NAT$source-category colimiting-cocone) language)
     (= (NAT$target-category colimiting-cocone) set$set)
     (= (NAT$source-functor colimiting-cocone) indicia-pair)
     (= (NAT$target-functor colimiting-cocone) tuple-pair)
     (= (NAT$component colimiting-cocone) trm.obj.col.coprd2$colimiting-cocone)
```

```
    (= (NAT$vertical-composition [colimiting-cocone cocone-diagram])
       (NAT$vertical-identity indicia-pair))
```

There is a colimit natural transformation from indicia pairs to indicia, whose L^{th} component is given by the colimit function for a term language L, and there are two injection natural transformations from indicia pairs to tuples, whose L^{th} component is given by the two injection functions for a term language L

$+ : var \circ \wp \times var \circ \wp \Rightarrow var \circ \wp$: Trm-Lang → Set,
$\iota_1 : var \circ \wp \times var \circ \wp \Rightarrow tpl$: Trm-Lang → Set, and
$\iota_2 : var \circ \wp \times var \circ \wp \Rightarrow tpl$: Trm-Lang → Set.

The colimiting cocone natural transformation factors, according to the components of a cocone, into the colimit (opvertex) and injection (opfirst and opsecond) natural transformations:

$+ = \nabla \bullet opvtx$, $\iota_1 = \nabla \bullet op1^{st}$ and $\iota_2 = \nabla \bullet op2^{nd}$.

```
(46) (NAT$natural-transformation colimit)
     (NAT$natural-transformation binary-coproduct)
     (= binary-coproduct colimit)
     (= (NAT$source-category colimit) language)
     (= (NAT$target-category colimit) set$set)
     (= (NAT$source-functor colimit) indicia-pair)
     (= (NAT$target-functor colimit) indicia)
     (= (NAT$component colimit) trm.obj.col.coprd2$colimit)
     (= colimit (NAT$vertical-composition [colimiting-cocone opvertex]))

(47) (NAT$natural-transformation injection1)
     (= (NAT$source-category injection1) language)
     (= (NAT$target-category injection1) set$set)
     (= (NAT$source-functor injection1) indicia-pair)
     (= (NAT$target-functor injection1) tuple)
     (= (NAT$component injection1) trm.obj.col.coprd2$injection1)
     (= injection1 (NAT$vertical-composition [colimiting-cocone opfirst]))

(48) (NAT$natural-transformation injection2)
     (= (NAT$source-category injection2) language)
     (= (NAT$target-category injection2) set$set)
     (= (NAT$source-functor injection2) indicia-pair)
     (= (NAT$target-functor injection2) tuple)
     (= (NAT$component injection2) trm.obj.col.coprd2$injection2)
     (= injection2 (NAT$vertical-composition [colimiting-cocone opsecond]))
```

There is a comediator (pairing) natural transformation from tuple pairs to tuples, whose L^{th} component is the comediator (pairing) function for a term language L

$[\text{-}] : tpl \oplus tpl \Rightarrow tpl$: Trm-Lang → Set.

The index (source) of the pairing tuple is the colimit of the underlying indicia pair, and the arity (target) of the pairing tuple is the opvertex:

$[\text{-}] \bullet \S = dgm \bullet +$ and $[\text{-}] \bullet \# = opvtx$.

```
(49) (NAT$natural-transformation comediator)
     (NAT$natural-transformation pairing)
     (= pairing comediator)
     (= (NAT$source-category comediator) language)
     (= (NAT$target-category comediator) set$set)
     (= (NAT$source-functor comediator) tuple-pair)
     (= (NAT$target-functor comediator) tuple)
     (= (NAT$component comediator) trm.obj.col.coprd2$comediator)

     (= (NAT$vertical-composition [pairing index])
        (NAT$vertical-composition [cocone-diagram colimit]))
     (= (NAT$vertical-composition [pairing arity]) opvertex)
```

Monad

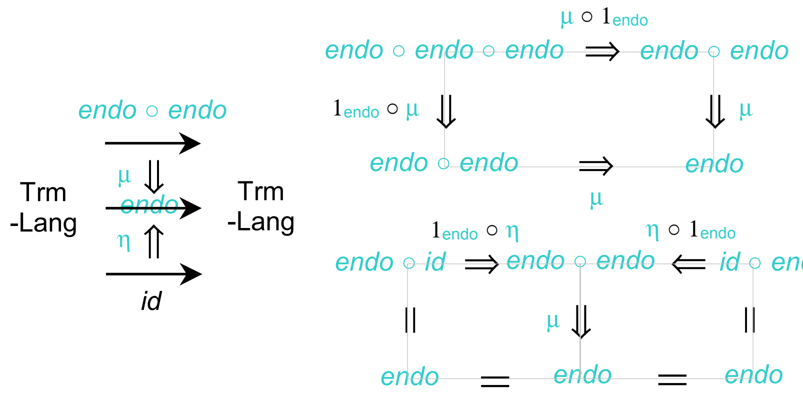

Figure 10: Multiplication and Unit Natural Transformations

There is a term language *endofunctor* on the (large) category of term languages

endo : Trm-Lang → Trm-Lang.

whose composition with the variable and function functors give the variable and term functors, respectively

endo ∘ *var* = *var* and *endo* ∘ *ftn* = *trm*.

There are term *unit* and term *multiplication* natural transformations (Figure 10),

η = *unit* : *id* ⇒ *endo* : Trm-Lang → Trm-Lang and

μ = *mult* : *endo* ∘ *endo* ⇒ *endo* : Trm-Lang → Trm-Lang,

whose $\boldsymbol{L}^{\text{th}}$ components are the embedding and collapsing term language morphisms for a term language ***L***. Term unit and term multiplication satisfy the following associative and left/right unit laws:

$(1_{endo} \circ \mu) \bullet \mu = (\mu \circ 1_{endo}) \bullet \mu$,

$(1_{endo} \circ \eta) \bullet \mu = 1_{endo}$ and $(\eta \circ 1_{endo}) \bullet \mu = 1_{endo}$.

```
(50) (FUNC$functor endofunctor)
     (= (FUNC$source endofunctor) language)
     (= (FUNC$target endofunctor) language)
     (= (FUNC$object endofunctor) trm.obj$term)
     (= (FUNC$morphism endofunctor) trm.mor$term)

     (= (FUNC$composition [endofunctor variable]) variable)
     (= (FUNC$composition [endofunctor function]) term)

(51) (NAT$natural-transformation unit)
     (= (NAT$source-category unit) language)
     (= (NAT$target-category unit) language)
     (= (NAT$source-functor unit) (FUNC$identity language))
     (= (NAT$target-functor unit) endofunctor)
     (= (NAT$component unit) trm.obj$embedding)

(52) (NAT$natural-transformation multiplication)
     (= (NAT$source-category multiplication) language)
     (= (NAT$target-category multiplication) language)
     (= (NAT$source-functor multiplication) (FUNC$composition [endofunctor endofunctor]))
     (= (NAT$target-functor multiplication) endofunctor)
     (= (NAT$component multiplication) trm.obj$collapsing)

     (= (NAT$vertical-composition
            [(NAT$horizontal-composition
                 [(NAT$vertical-identity language) multiplication]) multiplication])
        (NAT$vertical-composition
            [(NAT$horizontal-composition
                 [multiplication (NAT$vertical-identity language)]) multiplication]))
     (= (NAT$vertical-composition
            [(NAT$horizontal-composition
```

```
                [(NAT$vertical-identity language) unit]) multiplication])
         (NAT$vertical-identity language))
      (= (NAT$vertical-composition
             [(NAT$horizontal-composition
                  [unit (NAT$vertical-identity language)]) multiplication])
         (NAT$vertical-identity language))
```

Finally, there is a term *monad* *mnd* = ⟨*endo*, η, μ⟩, whose endofunctor is the term endofunctor, whose unit is the term unit and whose multiplication is the term multiplication.

```
(53) (MND$monad monad)
     (= (MND$endofunctor monad) endofunctor)
     (= (MND$unit monad) unit)
     (= (MND$multiplication monad) multiplication)
```