Core Considerations

Core Modifications

The category design principle is a primary goal in the design of the lower metalevel of the IFF. The following categorical design principle is critically important in meta-ontology design.

Categorical Principle: *The design of any module situated in the lower metalevel of the IFF Foundation Ontology should adhere to the categorical property that its axiomatic representation is strictly category-theoretic:*

- *All axioms are expressed in terms of either terms from another lower metalevel module or category-theoretic notions (upper metalevel notions), such as classes, binary relations, order, the composition and identity class functions, the pullback of diagrams of class functions, etc.*
- *No axioms use explicit logical notation: No variables, quantification (*'`forall`', '`exists`'*) or logical connectives (*'`and`', '`or`', '`not`', '`=>`','`<=>`'*) are used.*
- *No axioms use terms from the IFF Basic KIF ontology, other than pair bracketing* '`[-]`' *or pair projection* '`(- 1)`', '`(- 2)`'.

In order to attain this goal, several additions to the IFF must be made. These comments apply mainly to the IFF Upper Core (meta) Ontology (IFF-UCO).

Composition and Composite

Often, a class parameter A is needed when axiomatizing in terms of composition in some category C. Consider Figure 1, where *mor*(C) denotes the morphism class, $mor(C)\times_{obj(C)}mor(C)$ denotes the class of composable morphism pairs,

$1^{st}_C : mor(C)\times_{obj(C)}mor(C) \to mor(C)$

$2^{nd}_C : mor(C)\times_{obj(C)}mor(C) \to mor(C)$

$\circ_C : mor(C)\times_{obj(C)}mor(C) \to mor(C)$

denote the first and second projections and composition functions, and

$[f_1, f_2] : A \to mor(C)\times_{obj(C)}mor(C)$

denotes the pairing of two class functions

$f_1 : A \to mor(C)$

$f_2 : A \to mor(C)$

that form a cone over the composable opspan.

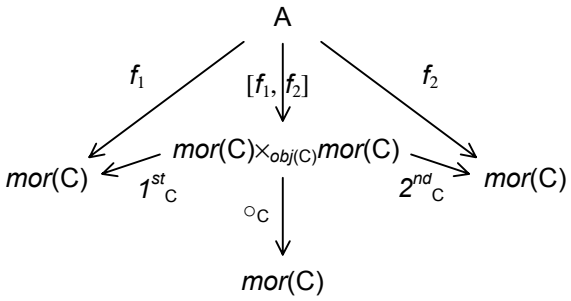


Figure 1: Composite

In the following, we make comments about particular categories, such as Set, Pfn, Expr-Lang, etc.

- In order to define various functions we use a multiple abbreviation from the function namespace of the IFF Upper Core (meta) Ontology (IFF-UCO). The KIF function

  ```
  (SET.LIM.PBK$pairing set.mor$composable-opspan)
  ```

  is a convenient abbreviation for the 'mediator' function for set function composition, where we only need to indicate the pair of first and second functions for a class pullback cone, not the whole cone

itself. Furthermore, concerning the pullback situation for set function composition (opspan, cones, pullback, mediator and equivalent pairing), the function

```
(set.mor$composite [?fn1 ?fn2])
```

is an abbreviation for the class function composition

```
(SET.FTN$composition
     [((SET.LIM.PBK$pairing set.mor$composable-opspan) [?fn1 ?fn2])
      set.ftn$composition])
```

The full code is

```
(KIF$function set.mor$composite)
(= (KIF$source set.mor$composite) SET.FTN$function-pair)
(= (KIF$target set.mor$composite) SET.FTN$function)
(forall (?f1 (SET.FTN$function ?f1) ?f1 (SET.FTN$function ?f1)
         ((SET.LIM.PBK$pairable set.mor$composable-opspan) ?f1 ?f2))
    (= (set.mor$composite [?f1 ?f2])
       (SET.FTN$composition
           [((SET.LIM.PBK$pairing set.mor$composable-opspan) [?f1 ?f2])
            set.mor$composition])))
```

- We can do the same for the category of partial set functions, getting the function

```
(set.pfn$composite [?pfn1 ?pfn2])
```

as an abbreviation for the class function composition

```
(SET.FTN$composition
     [((SET.LIM.PBK$pairing set.pfn$composable-opspan) [?pfn1 ?pfn2])
      set.pfn$composition])
```

- We can also do the same for the category of expression language morphisms, getting the function

```
(expr.mor$composite [?f1 ?f2])
```

as an abbreviation for the class function composition

```
(SET.FTN$composition
     [((SET.LIM.PBK$pairing expr.mor$composable-opspan) [?f1 ?f2])
      expr.mor$composition])
```

- This idea works also for set quartets. There is a function

```
(set.qtt$horizontal-composite [?q1 ?q2])
```

which is an abbreviation for the class function composition

```
(SET.FTN$composition
     [((SET.LIM.PBK$pairing set.qtt$horizontal-composable-opspan) [?q1 ?q2])
      set.qtt$horizontal-composition])
```

And there is a function

```
(set.qtt$vertical-composite [?q1 ?q2])
```

which is an abbreviation for the class function composition

```
(SET.FTN$composition
     [((SET.LIM.PBK$pairing set.qtt$vertical-composable-opspan) [?q1 ?q2])
      set.qtt$vertical-composition])
```

Relative Power Construction

A *monad* $T = \langle T, \eta, \mu \rangle$ in a category A consists of an endofunctor $T : \mathsf{A} \to \mathsf{A}$ and two natural transformations $\eta : id \Rightarrow T$ and $\mu : T \circ T \Rightarrow T$ that make the diagrams in Figure 2 commute. The natural transformation η is called the *unit* of the monad, and the natural transformation μ is called the *multiplication* of the monad.

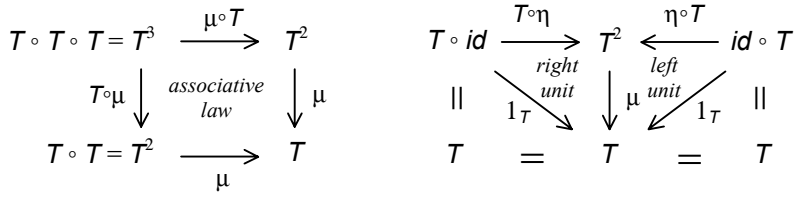


Figure 2: Associative and Unit Laws for a Monad

For any set B, the *comma category* $(\mathsf{Set} \downarrow B)$ has sets over B as objects and functions over B as morphisms. A set over B is a pair $\langle A, a\rangle$ consisting of a set A and a set function $a : A \to B$. A function over B $h : \langle A_1, a_1\rangle \to \langle A_2, a_2\rangle$, with source $\langle A_1, a_1\rangle$ and target $\langle A_2, a_2\rangle$, is a function $h : A_1 \to A_2$ that commutes with source/target: $h \cdot a_2 = a_1$. There is an obvious *projection functor* $\mathsf{P} : (\mathsf{Set} \downarrow B) \to \mathsf{Set}$. For the terminal singleton set $B = \mathbf{1}$, we have the isomorphism $(\mathsf{Set} \downarrow \mathbf{1}) \cong \mathsf{Set}$. That is, the traditional case of the nonempty power set of a set A is a special case of the nonempty power of $\langle A, !_A\rangle$, the unique function $!_A : A \to \mathbf{1}$.

The nonempty power construction in set theory gives a standard example of a monad $\wp = \langle \wp, \{\text{-}\}, \cup\rangle$ called the *power monad* in the category Set, where the endofunctor is powerset-direct-image, the unit is singleton and the multiplication is union. The power monad in Set can be lifted and generalized to a power monad $\wp = \langle \wp, \{\text{-}\}, \cup\rangle$ in the comma category $(\mathsf{Set} \downarrow B)$ for any set B. This might be called the power construction over B.

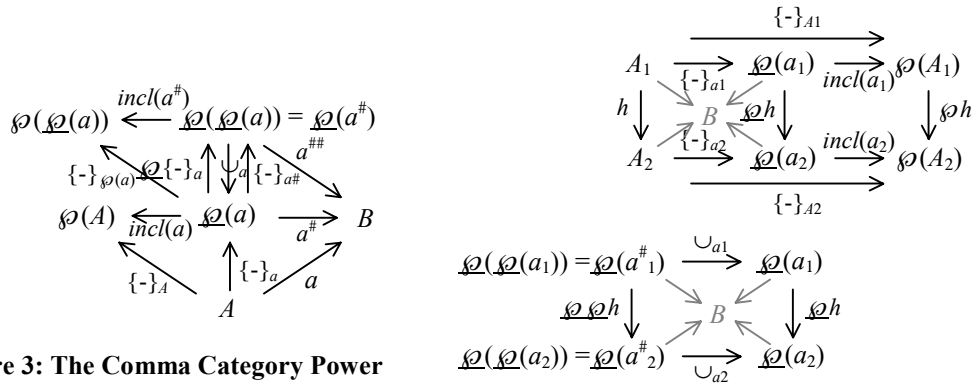


Figure 3: The Comma Category Power

- The endofunctor: $\wp : (\mathsf{Set} \downarrow B) \to (\mathsf{Set} \downarrow B)$

  [objects:] $\wp$ maps $\langle A, a\rangle$, a set over B, to $\langle \underline{\wp}a, a^{\#}\rangle$, another set over B, where $\underline{\wp}a \subseteq \wp A$ denotes the collection of all nonempty subsets of A with constant a value: $C \in \underline{\wp}a$ when $a|_C : C \to B$ is a constant function; that is, $C \in \underline{\wp}a$ when $C \in \wp A$ and there is an element $b \in B$ such that $a(c) = b$ for all $c \in C$, and $a^{\#} : \underline{\wp}a \to B$ is the evaluation function that maps any nonempty set $C \in \underline{\wp}a$ to the element $a(c) \in B$ for $c \in C$.

  [morphisms:] $\wp$ maps $h : \langle A_1, a_1\rangle \to \langle A_2, a_2\rangle$, a function over B, to $\underline{\wp}h : \langle \underline{\wp}a_1, a_1^{\#}\rangle \to \langle \underline{\wp}a_2, a_2^{\#}\rangle$, another function over B, where $\underline{\wp}h : \underline{\wp}a_1 \to \underline{\wp}a_2$ is the direct-image power function (for any subset $C_1 \subseteq A_1$, if a_1 is constant on C_1, then a_2 is constant on the direct image $\wp h(C_1) \subseteq A_2$) that satisfies the condition $\underline{\wp}h \cdot a_2^{\#} = a_1^{\#}$.

- The unit natural transformation: $\eta : \mathit{id} \Rightarrow \wp : (\mathsf{Set} \downarrow B) \to (\mathsf{Set} \downarrow B)$ has as its $\langle A, a\rangle^{\text{th}}$ component the *singleton morphism* $\{\text{-}\}_a : \langle A, a\rangle \to \langle \wp(a), a^{\#}\rangle = \wp(\langle A, a\rangle)$, a function over B. This is the function $\{\text{-}\}_a : A \to \underline{\wp}a$ that restricts the standard singleton function $\{\text{-}\}_A : A \to \wp A$. The singleton morphism satisfies the defining condition: $\{\text{-}\}_a \cdot a^{\#} = a$. It satisfies the naturality condition: $h \cdot \{\text{-}\}_{a2} = \{\text{-}\}_{a1} \cdot \underline{\wp}h$ for all set functions $h : a_1 \to a_2$. And it also satisfies the condition: $\{\text{-}\}_a \cdot \mathit{incl}(a) = \{\text{-}\}_A$.

- The multiplication natural transformation: $\mu : \wp \circ \wp \Rightarrow \wp : (\mathsf{Set} \downarrow B) \to (\mathsf{Set} \downarrow B)$ has as its $\langle A, a\rangle^{\text{th}}$ component the *nonempty union morphism* $\cup_a : \wp(\wp(\langle A, a\rangle)) = \langle \underline{\wp}\underline{\wp}a, a^{\#\#}\rangle \to \langle \underline{\wp}(a), a^{\#}\rangle = \wp(\langle A, a\rangle)$, a function over B. The iterated application yields $\wp(\wp(\langle A, a\rangle)) = \wp(\langle \underline{\wp}a, a^{\#}\rangle) = \langle \underline{\wp}\underline{\wp}a, a^{\#\#}\rangle$, a set over B, consisting of the collection of all nonempty collections of nonempty subsets of A with constant a value, and $a^{\#\#}$ is the evaluation function $a^{\#\#} : \underline{\wp}a^{\#} = \underline{\wp}\underline{\wp}a \to B$. The nonempty union morphism satisfies the defining condition: $\cup_a \cdot a^{\#} = a^{\#\#}$. It also satisfies the naturality condition: $\underline{\wp}\underline{\wp}h \cdot \cup_{a2} = \cup_{a1} \cdot \underline{\wp}h$ for all set functions $h : a_1 \to a_2$.

In summary, for any set function $a : A \to B$

- There is an inclusion function $\mathit{incl}(a) : \underline{\wp}a \to \wp A$.
- There is an extension of a to $a^{\#} : \underline{\wp}a \to B$. This satisfies the condition: $\{\text{-}\}_a \cdot a^{\#} = a$, asserting that singleton $\{\text{-}\}_a : \langle A, a\rangle \to \langle \underline{\wp}(a), a^{\#}\rangle$ is function over B (morphism in the comma category $(\mathsf{Set} \downarrow B)$.

- There is a power function $\underline{\wp}h : \underline{\wp}a_1 \to \underline{\wp}a_2$, which is compatible with the traditional power function. This satisfies the naturality conditions (the last follows from the other two: $\{\text{-}\}_{a1} \cdot \underline{\wp}h = h \cdot \{\text{-}\}_{a2}$, $\underline{\wp}h \cdot incl(a_2) = incl(a_1) \cdot \wp h$, and $\{\text{-}\}_{A1} \cdot \wp h = h \cdot \{\text{-}\}_{A2}$.
- The traditional singleton function $\{\text{-}\}_A : A \to \wp A$ restricts to a singleton function $\{\text{-}\}_a : A \to \underline{\wp}a$. This satisfies the defining condition: $\{\text{-}\}_a \cdot a^{\#} = a$. It satisfies the naturality condition: $h \cdot \{\text{-}\}_{a2} = \{\text{-}\}_{a1} \cdot \underline{\wp}h$ for all set functions $h : a_1 \to a_2$. And it also satisfies the condition: $\{\text{-}\}_a \cdot incl(a) = \{\text{-}\}_A$.
- There is a nonempty union function $\cup_a : \underline{\wp}\,\underline{\wp}a \to \underline{\wp}a$. This satisfies the defining condition: $\cup_a \cdot a^{\#} = a^{\#\#}$. It also satisfies the naturality condition: $\underline{\wp}\,\underline{\wp}h \cdot \cup_{a2} = \cup_{a1} \cdot \underline{\wp}h$ for all set functions $h : a_1 \to a_2$.
- There is a membership relation $\underline{\in}_a \subseteq \underline{\wp}a \times A$ that is the abridgment of the traditional membership relation $\in_A \subseteq \wp A \times A$. This satisfies the condition: $\phi_{12}(\underline{\in}_a) = incl(a)$.

The IFF Lower Core (meta) Ontology IFF-LCO represents this idea of a "power monad over a set". The following code should appear in an appropriate namespace in the IFF-LCO ontology.

```
(power-object ?b) : (set-over-set ?b) → (set-over-set ?b)
(power-morphism ?b) : (function-over-set ?b) → (function-over-set ?b)
(eta-component ?b) :  (set-over-set ?b) → (function-over-set ?b)
(mu-component ?b) :  (set-over-set ?b) → (function-over-set ?b)
```

```
(1) (KIF$function set-over-set)
    (= (KIF$source set-over-set) set.obj$set)
    (= (KIF$source set-over-set) SET$class)

(2) (KIF$function set-over)
    (= (KIF$source set-over) set.obj$set)
    (= (KIF$source set-over) SET.FTN$function)
    (forall (?b (set.obj$set ?b))
        (and (= (SET.FTN$source (set-over ?b)) (set-over-set ?b))
             (= (SET.FTN$target (set-over ?b)) set.obj$set)))

(3) (KIF$function connecting-function)
    (= (KIF$source connecting-function) set.obj$set)
    (= (KIF$source connecting-function) SET.FTN$function)
    (forall (?b (set.obj$set ?b))
        (and (= (SET.FTN$source (connecting-function ?b)) (set-over-set ?b))
             (= (SET.FTN$target (connecting-function ?b)) set.mor$function)
             (= (SET.FTN$composition [(connecting-function ?b) set.mor$source])
                (set-over ?b))
             (= (SET.FTN$composition [(connecting-function ?b) set.mor$target]) ?b)))

(4) (SET.FTN$function nonempty-power)
    (= (SET.FTN$source nonempty-power) set.mor$function)
    (= (SET.FTN$source nonempty-power) set.obj$set)
    (forall (?a (set.mor$function ?a))
        (and (set.obj$subset (nonempty-power ?a) (set.obj$power (set.mor$source ?a)))
             (forall (?c ((set.mor$source ?a) ?c))
                 (<=> ((nonempty-power ?a) ?c)
                      (and (set.obj$nonempty ?c)
                           (exists ((set.mor$target ?a) ?b))
                               (= ((source-restriction ?a) ?c)
                                  ((set.ftn$constant [?c (set.mor$target ?a)]) ?b)))))))))

(5) (SET.FTN$function extension-function)
    (= (SET.FTN$source extension-function) set.mor$function)
    (= (SET.FTN$source extension-function) set.mor$function)
    (= (SET.FTN$composition [extension-function set.mor$source])
       (SET.FTN$composition [set.mor$source nonempty-power]))
    (= (SET.FTN$composition [extension-function set.mor$target]) set.mor$target)
    (forall (?a (set.mor$function ?a)
             ?c ((set.mor$source ?a) ?c) ?x (?c ?x))
        (= ((extension-function ?a) ?c) (?a ?x)))

(4) (KIF$function power-object)
    (= (KIF$source power-object) set.obj$set)
    (= (KIF$source power-object) SET.FTN$function)
    (forall (?b (set.obj$set ?b))
```

```
        (and (= (SET.FTN$source (power-object ?b)) (set-over-set ?b))
             (= (SET.FTN$target (power-object ?b)) (set-over-set ?b))
             (= (SET.FTN$composition [(power-object ?b) (set-over ?b)])
                (SET.FTN$composition [(connecting-function ?b) nonempty-power)
             (= (SET.FTN$composition [(power-object ?b) (connecting-function ?b)])
                (SET.FTN$composition [(connecting-function ?b) extension-function)))
```

```
(2) (SET.FTN$function inclusion)
    (= (SET.FTN$source inclusion) set.mor$function)
    (= (SET.FTN$source inclusion) set.mor$function)
    (= (SET.FTN$composition [inclusion set.mor$source]) power)
    (= (SET.FTN$composition [inclusion set.mor$target])
       (SET.FTN$composition [set.mor$source set.obj$power))
    (forall (?a (set.mor$function ?a))
        (= (inclusion ?a)
           (set.mor$inclusion [(power ?a) (set.obj$power ?a)])))

(3) (SET.FTN$function extension)
    (= (SET.FTN$source extension) set.mor$function)
    (= (SET.FTN$source extension) set.mor$function)
    (= (SET.FTN$composition [extension set.mor$source]) power)
    (= (SET.FTN$composition [extension set.mor$target]) set.mor$target)
    (forall (?a (set.mor$function ?a))
        (= (set.mor$composition [(singleton ?a) (extension ?a)]) ?a))

(4) (SET.FTN$function power-power)
    (= (SET.FTN$source power-power) set.mor$function)
    (= (SET.FTN$source power-power) set.obj$set)
    (= power-power (SET.FTN$composition [extension power]))

(5) (SET.FTN$function union)
    (= (SET.FTN$source union) set.mor$function)
    (= (SET.FTN$source union) set.mor$function)
    (= (SET.FTN$composition [union source]) power-power)
    (= (SET.FTN$composition [union target]) power)
    (forall (?a (set.mor$function ?a))
        (= (set.mor$composition [(singleton ?a) (inclusion ?a)])
           (set.obj$singleton (source ?a)))

(4) (SET.FTN$function singleton)
    (= (SET.FTN$source singleton) function)
    (= (SET.FTN$source singleton) function)
    (= (SET.FTN$composition [singleton source]) source)
    (= (SET.FTN$composition [singleton target]) power)
    (forall (?a (function ?a))
        (= (set.mor$composition [(singleton ?a) (inclusion ?a)])
           (set.obj$singleton (source ?a)))

(5) (SET.FTN$function member)
    (= (SET.FTN$source member) set.mor$function)
    (= (SET.FTN$source member) rel$relation)
    (= (SET.FTN$composition [member rel$set1]) power)
    (= (SET.FTN$composition [member rel$set2]) source)
    (= (SET.FTN$composition [member rel$fiber12]) inclusion)
    (forall (?a (set.mor$function ?a))
        (rel$abridgement (member ?a) (set.obj$member (source ?a))))
```

Core Theory

Quartets and Natural Transformations

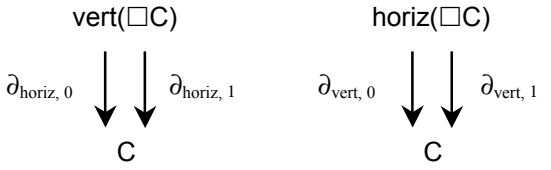


Figure 4: The Categories and Functors implicit within □C the Quartet of C

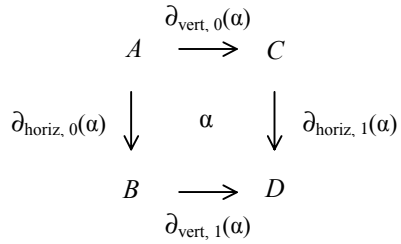
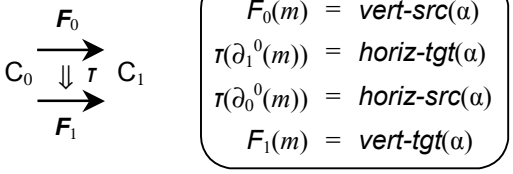


Diagram 1: Set Quartet



$$\begin{aligned} F_0(m) &= \textit{vert-src}(\alpha) \\ \tau(\partial_1^{\,0}(m)) &= \textit{horiz-tgt}(\alpha) \\ \tau(\partial_0^{\,0}(m)) &= \textit{horiz-src}(\alpha) \\ F_1(m) &= \textit{vert-tgt}(\alpha) \end{aligned}$$

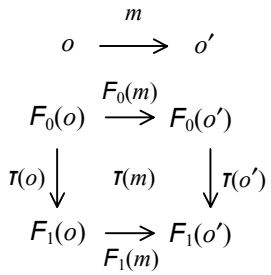


Figure 5: Natural Transformation

Diagram 2: Naturality – Commutative Square for a Natural Transformation

Quartets: This document makes reasonably heavy use of the notion of a *quartet*, which is axiomatized in the IFF Lower Core (meta) Ontology (IFF-LCO). Functions are related through set quartets. In general, for any category C, the objects, morphisms and quartets of C form a special double category □C called the *quartet* of C (Figure 4). A C-*quartet* (C-*morphism square*) α (Diagram 1), horizontally from C-morphism $\partial_{\text{horiz, 0}}(\alpha)$ to C-morphism $\partial_{\text{horiz, 1}}(\alpha)$, $\alpha = \langle \partial_{\text{vert, 0}}(\alpha), \partial_{\text{vert, 1}}(\alpha) \rangle : \partial_{\text{horiz, 0}}(\alpha) \to \partial_{\text{horiz, 1}}(\alpha)$, and vertically from C-morphism $\partial_{\text{vert, 0}}(\alpha)$ to C-morphism $\partial_{\text{vert, 1}}(\alpha)$, $\alpha = \langle \partial_{\text{horiz, 0}}(\alpha), \partial_{\text{horiz, 1}}(\alpha) \rangle : \partial_{\text{vert, 0}}(\alpha) \to \partial_{\text{vert, 1}}(\alpha)$, consists of a commutative Diagram 1 of these four C-morphisms, $\partial_{\text{vert, 0}}(\alpha) \cdot \partial_{\text{horiz, 1}}(\alpha) = \partial_{\text{horiz, 0}}(\alpha) \cdot \partial_{\text{vert, 1}}(\alpha)$, which implies that the four C-morphisms suitably match at source and target C-objects: $\textit{src}(\partial_{\text{vert, 0}}(\alpha)) = \textit{src}(\partial_{\text{horiz, 0}}(\alpha))$, $\textit{tgt}(\partial_{\text{vert, 1}}(\alpha)) = \textit{tgt}(\partial_{\text{horiz, 1}}(\alpha))$, $\textit{composable}(\partial_{\text{vert, 0}}(\alpha), \partial_{\text{horiz, 1}}(\alpha))$, and $\textit{composable}(\partial_{\text{horiz, 0}}(\alpha), \partial_{\text{vert, 1}}(\alpha))$.

Natural Transformations: A very common kind of quartet is the commutativity square for a natural transformation. Natural transformations are axiomatized in the IFF Category Theory (meta) Ontology (IFF-CAT). Assume that two functors $F_0, F_1 : \mathsf{C}_0 \to \mathsf{C}_1$ share common source and target categories C_0 and C_1. A natural transformation τ from source functor F_0 to target functor F_1, written 1-dimensionally as $\tau : F_0 \Rightarrow F_1 : \mathsf{C}_0 \to \mathsf{C}_1$ or visualized 2-dimensionally in Figure 5, is a collection of morphisms in the target category parameterized by objects in the source category that link the functorial images: $\tau = \{\tau(o) : F_0(o) \to F_1(o) \mid o \in \textit{obj}(\mathsf{C}_0)\}$. A natural transformation is determined by its (source-functor, target-functor, component) triple. The components of a natural transformation interact with the functorial images by satisfying the *fundamental property*, which for any morphism $m \in \textit{mor}(\mathsf{C}_0)$ consists of a commutative Diagram 2 of four functions, $F_0(m) \circ^1 \tau(\partial_1^{\,0}(m)) = \tau(\partial_0^{\,0}(m)) \circ^1 F_1(m)$. This implies that the four functions suitably match at source and target sets: $\textit{src}(F_0(m)) = \textit{src}(\tau(\partial_0^{\,0}(m)))$, $\textit{tgt}(F_1(m)) = \textit{tgt}(\tau(\partial_1^{\,0}(m)))$, $\textit{composable}(F_0(m), \tau(\partial_1^{\,0}(m)))$, and $\textit{composable}(\tau(\partial_0^{\,0}(m)), F_1(m))$. Hence, for any morphism $m \in \textit{mor}(\mathsf{C}_0)$, there is a set quartet (vertically) $\tau(m) = \langle \tau(\partial_0^{\,0}(m)), \tau(\partial_1^{\,0}(m)) \rangle : F_0(m) \to F_1(m)$. This quartet is the transpose of the usual presentation of natural transformations (see the drawings in Mac Lane or the image on the home page of CT04) with the components being horizontal arrows. We have chosen this way, with the components being vertical arrows, so that the vertical composition of two natural transformations corresponds to the vertical composition of the associated quartets. Thus, a natural transformation is a

functor $T : \mathsf{C}_0 \to \square\mathsf{C}_1$ where the source/target functors are defined by composition with projection: $F_0 = T \circ \partial_{\text{vert, 0}}$ and $F_1 = T \circ \partial_{\text{vert, 1}}$, and the component function is defined by $\tau = id_{C0} \cdot T \cdot \partial_{\text{horiz, 0}} = id_{C0} \cdot T \cdot \partial_{\text{horiz, 1}}$. For any morphism $m \in mor(\mathsf{C}_0)$, we symbolize the m-quartet of a natural transformation by $\tau(m) : F_0(m) \Rightarrow F_1(m)$.

Pointwise Limits

If a category B is complete, then for any other category A, the functor category [A, B] is also complete and limits can be computed pointwise. Since this situation happens in the IFF First Order Logic (meta) Ontology (IFF-FOL), we will review some of the details here. Let G be any graph and let $D : \mathsf{G} \to [\mathsf{A}, \mathsf{B}]$ be any diagram of shape G in the functor category [A, B]. This consists of a collection of functors

$$\{D(n) : \mathsf{A} \to \mathsf{B} \mid n \in node(\mathsf{G})\}$$

indexed by the nodes of G, and a collection of natural transformations

$$\{D(e) : D(m) \Rightarrow D(n) \mid e \in edge(\mathsf{G}), e : m \to n\}$$

indexed by the edges of G (Figure 6a illustrates the transpose of the associated quartet).

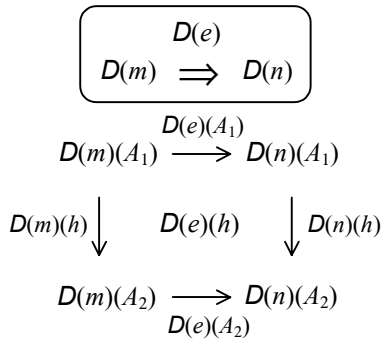


Figure 6a: Diagram Component Quartet^{op} (Naturality)

- We want to define the *limit* of D in [A, B]. Denote this functor by

  $$\prod D : \mathsf{A} \to \mathsf{B}.$$

  [**Objects:**] For any object $A \in obj(\mathsf{A})$, the A^{th} *coordinate* of D is the diagram $D_A : \mathsf{G} \to \mathsf{B}$, where $D_A(n) = D(n)(A) \in obj(\mathsf{B})$ for all nodes $n \in node(\mathsf{G})$, and $D_A(e) = D(e)(A) : D_A(m) \to D_A(n)$ is the A^{th} component of the natural transformation $D(e) : D(m) \Rightarrow D(n)$ for all edges $e \in edge(\mathsf{G})$ with $e : m \to n$. Define the limit B-object "the A^{th} coordinate of the limit is the limit of the A^{th} coordinate"

  $$\prod D(A) = \prod D_A \in obj(\mathsf{B}).$$

  [**Morphisms:**] For any morphism $h \in mor(\mathsf{A})$ with $h : A_1 \to A_2$, the h^{th} *coordinate* of D is the diagram morphism $D_h : D_{A1} \Rightarrow D_{A2} : \mathsf{G} \to \mathsf{B}$, whose n^{th} component is the B-morphism $D_h(n) = D(n)(h) : D(n)(A_1) \to D(n)(A_2)$ for all nodes $n \in node(\mathsf{G})$, and whose e^{th} component is the B-quartet $D(e)(h)$ shown in Figure 6a for all G-edges $e : m \to n$. Define the limit B-morphism "the h^{th} coordinate of the limit is the limit of the h^{th} coordinate"

  $$\prod D(h) = \prod D_h \in obj(\mathsf{B}) \text{ with } \prod D_h : \prod D_{A1} \to \prod D_{A2}.$$

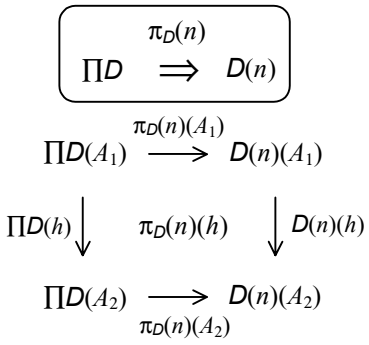


Figure 6b: Projection Quartet^{op} (Naturality)

- We also want to define the *limit projections* of D. Denote these natural transformations by

  $$\pi_D : \Delta(\prod D) \Rightarrow D : \mathsf{G} \to [\mathsf{A}, \mathsf{B}].$$

  For any node $n \in node(\mathsf{G})$, the n^{th} *coordinate* of π_D is the natural transformation $\pi_D(n) : \prod D \Rightarrow D(n) : \mathsf{A} \to \mathsf{B}$, whose A^{th} component is the projection $\pi_D(n)(A) = \pi : \prod D_A \to D_A(n) = D(n)(A)$ for the limit of the diagram $D_A : \mathsf{G} \to \mathsf{B}$, which is a B-morphism, and whose e^{th} component is the B-quartet $\pi_D(n)(h)$ whose transpose is shown in Figure 6b for all G-edges $e : m \to n$.

Example

In the IFF-FOL, there is a substitution operator that takes a matching term-tuple pair and returns the tuple substituted into the term. Matching means that term arity equals tuple index. This is the case of a pullback in a functor category. In particular, in terms of the above discussion, assume (Table 1) that the base category B is the category of sets and functions (which is complete), the indexing category A is the category of first order logic (FOL) languages and FOL language morphisms, the shape graph is G is a graph consisting of three nodes {0, 1, 2} and two edges $10 : 1 \to 0$, $20 : 2 \to 0$, and the diagram D consists of (Figure 7a) the indicia, term and tuple functors, and the term arity and tuple index natural transformations. The limit (pullback) $\prod D$ is the term-tuple functor, and the two term and tuple (pullback) projection natural transformations (Figure 7b). The matching condition for function-tuple pairs (Figure 7c) is that the arity of the function is the index of the tuple: $\pi_{trm} \cdot \# = \pi_{tpl} \cdot \S$.

Table 1: Definitions

B	=	Set
A	=	FOL-Lang
G	=	*opspan* = • → • ← •
D	=	$D(0) = var \circ \wp$: FOL-Lang → Set → Set $D(1) = trm$: FOL-Lang → Set $D(2) = tpl$: FOL-Lang → Set $D(10) = \# = arity : trm \Rightarrow var \circ \wp$: FOL-Lang → Set $D(20) = \S = index : tpl \Rightarrow var \circ \wp$: FOL-Lang → Set
$\prod D$	=	$trm \otimes tpl$: FOL-Lang → Set $\pi_{trm} : trm \otimes tpl \Rightarrow ftn$: FOL-Lang → Set $\pi_{tpl} : trm \otimes tpl \Rightarrow tpl$: FOL-Lang → Set

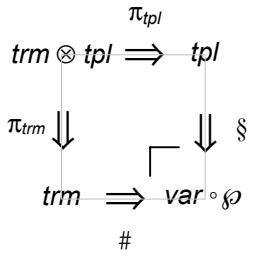


Figure 7c : Pullback Diagram

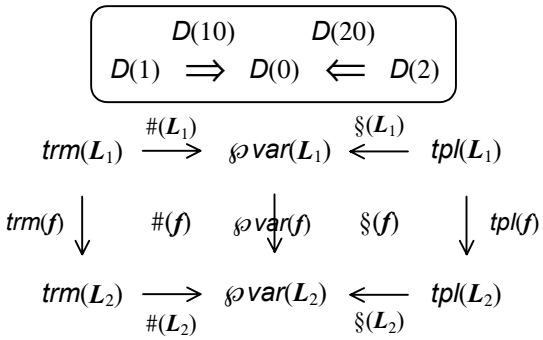


Figure 7a: Diagram Component Quartets (Naturality)

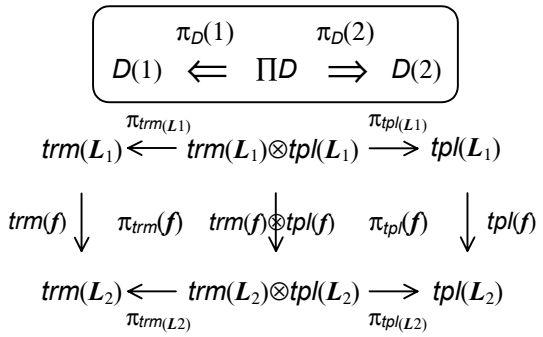


Figure 7b: Projection Quartets (Naturality)