

The Architecture, Portals and Implementation of the IFF

Robert E. Kent

November 1, 2005

Contents

1 Rationale	1
2 Architecture	2
3 Cantor’s Diagonal Argument	7
4 Portals	9
4.1 First Order Logic	9
5 Implementation	14
5.1 Data Structures	18

1 Rationale

Why the IFF? Why the metashell? To quote David Whitten: “We [in the knowledge sciences, technologies and industries] are now in a situation where we don’t have a common vocabulary at all. We now can’t (really) evaluate if two systems are the same or not, because we don’t have the formalized packages which express their distinctions. We don’t have a computational architecture which is rich enough, and formally defined enough that we can (formally) point out the differences in two different approaches to the same problem.” An important feature of the IFF is its tremendous extensibility. In its applications, beyond making ontological commitments to a vast collection of things, the IFF serves as a language for naming and describing these things. In one sense, the IFF is a nested collection of metalanguages with the IFF metashell at its core.

2 Architecture

The IFF architecture (Figure 1) consists of metalevels, namespaces and meta-ontologies. Within each level, the terminology is partitioned into namespaces. The number of namespaces and the content may vary over time: new namespaces may be created or old namespaces may be deprecated, and new terminology and axiomatization within any particular namespace may change (new versions). In addition, within each level, various namespaces are collected together into meaningful composites called meta-ontologies. At any particular metalevel, these meta-ontologies cover all the namespaces at that level, but they may overlap. The number of meta-ontologies and the content of any meta-ontology may vary over time: new meta-ontologies may be created or old meta-ontologies may be deprecated, and new namespaces within any particular meta-ontology may change (new versions).

The IFF terminology is managed in terms of namespace prefixes – each namespace is given a unique prefix (with perhaps a few synonyms) in order to avoid clash of terminology¹. Namespaces can be nested². This means that the axiomatization of one namespace may be included within the axiomatization of another namespace, when these are closely connected. An outermost namespace is not included in any other. An inner namespace is so included — it is not outermost. An atomic namespace is innermost — no other namespace is included in it. A molecular namespace has others included — it is not atomic. The architecture of the IFF namespace mechanism is flat — namespace prefixes are like tags: by using namespace prefixes the complete IFF terminology is the disjoint union of the terminology in the atomic IFF namespaces. The IFF architecture can be thought of as a two dimensional structure (Figure 1) consisting of metalevels, which are partitioned into outermost namespaces representing basic concepts such as diagram (`'dgm'`), graph (`'gph'`), category (`'cat'`) or institution (`'ins'`). The various levels are indexed by the natural numbers `'0'`, `'1'`, `'2'`, `'3'`, \dots , `' ∞ '` or by their natural language correlates: for the first four and infinite levels these are `'obj = 0'`, `'sm1 = 1'`, `'lrg = 2'`, `'vlrg = 3'` and `'ur = ∞ '`.

Along the vertical dimension (see Figure 1), the IFF architecture is partitioned into three parts: the objective part, level `'0'`, is where the object-level ontologies are located; the natural part contains the namespaces located at metalevels `'1' \dots ' ∞ '`; the supra-natural part contains the metashell axiomatization. Along the horizontal dimension, the natural part is partitioned into pure and applied aspects, and the pure part is partitioned into core and structural components. The pure aspect contains meta-ontologies, such as the IFF Core (meta) Ontology (**IFF-CORE**) and the IFF Category theory (meta) Ontology (**IFF-CAT**), axiomatizing the needed set-theoretic and category-theoretic founda-

¹For example, the term `'morphism'` is introduced in the contexts of models and theories for first order logic to represent two distinct, but analogous, concepts. When these concepts need to be used in other contexts, the namespace prefixes `'fol.mod.mor'` and `'fol.th.mor'` could be used to distinguish them, thus resulting in the distinct representations `'fol.mod.mor:morphism'` and `'fol.th.mor:morphism'`.

²This nesting does not refer to either the vertical or horizontal dimensions of the metalevel structure in Figure 1, but instead to an implicit third dimension of the architecture.

tions. The applied aspect contains meta-ontologies, such as the IFF Institution theory (meta) Ontology (**IFF-INS**), the IFF First Order Logic (meta) Ontology (**IFF-FOL**) and the IFF Ontology (meta) Ontology (**IFF-ONT**), providing the terminology and axiomatization for any needed logical and semiotic functionality. In the pure aspect of the natural part of the IFF, the axiomatization for any concept is in two component modules: the generic component gives the axiomatization at level n ($1 \leq n < \infty$) for that concept; the Ur component at level ∞ summarizes the axiomatization for all finite metalevels. The connection between the generic and Ur level (Figure 2) axiomatizations relies heavily upon the fundamental subset, (optimal-)restriction and abridgment partial orders for sets, (unary) functions and (binary) relations, respectively. The supra-natural part consists of only three namespaces: the medium-sized ‘**meta**’ namespace (**IFF-META**), located just above the natural part, which services the Ur and generic metalevels; the small-sized ‘**type**’ namespace, located just above this, which provides typing terminology for the ‘**meta**’ namespace and defines the four fundamental partial orders of subset, (optimal-)restriction and abridgment; and the tiny ‘**type**’ namespace, located at the very top of the IFF architecture, which provides the terminology for the three fundamental kinds of set, (unary) function and (binary) relation.

To locate any namespace one can use its level-concept pair, a concatenation called the full prefix of the name of the namespace. The full namespace prefix acts as a unique identifier of that namespace. For example, the namespace axiomatizing very large categories would be denoted by ‘**vlrg.cat**’ or ‘**3.cat**’. Inner namespaces need further qualification. For example, the inner namespace of small graph morphisms would be denoted by ‘**sm1.gph.mor**’. It is assumed that each basic concept has a particular metalevel that is in common use. These are indicated by the filled squares (■) for the particular concepts in Figure 1. For such commonly used namespaces, the level notation may be omitted. For example, the namespace that axiomatizes large categories could be denoted by ‘**lrg.cat**’ or ‘**2.cat**’, but a simpler notation is ‘**cat**’.

The namespace mechanism has been made backward compatible, by allowing special namespace prefix notation that is synonymous with the general format just described. For example, the namespace axiomatizing large categories would be denoted by the general prefixes ‘**cat**’, ‘**lrg.cat**’ or ‘**2.cat**’ or by the special prefix ‘**CAT**’ that was used in earlier versions of the **IFF-CAT** axiomatization. Hence, for any basic concept, both the general prefix and the common level need to be declared, and for any namespace axiomatization the special prefixes also need to be declared. The prefixes denoting inner namespaces should be compatible with outermost prefixes; for example, at level 2 the general prefix for the outermost namespace for limits would be ‘**lrg.lim**’ or ‘**2.lim**’ with the special prefix ‘**LIM**’, whereas the general prefix for the inner (non-atomic) namespace of pullbacks would be ‘**lrg.lim.pbk**’ with the special prefix ‘**LIM.PBK**’. As these examples illustrate, only the lower case is needed for the general namespace prefix notation.

The meta namespace in the metashell serves as an interface between the metashell and the natural part of the IFF. It does this by servicing the Ur

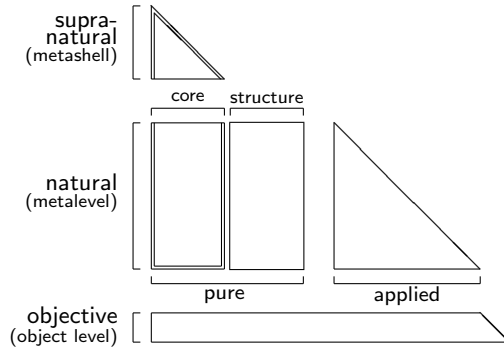
and generic metalevels. A design constraint is that the natural part should be fully conformable to the categorical design principle; this means that all axiomatization is atomic, consisting of either declarations, equations are relational expressions. Since the object-level axiomatization is expressed as atoms, this leaves only the metashell with full first order logical expressions. The metashell is used by the various meta-ontologies in the pure aspect of the natural part: the single meta-ontology in the core component being the (IFF-CORE); or the various meta-ontologies in the structural component including the (IFF-CAT), the IFF 2-Category theory (meta) Ontology (IFF-2CAT), and the future IFF Double Category theory (meta) Ontology (IFF-DCAT). The metashell acts as a bootstrapping mechanism from which the natural part can be unfurled, starting with the the IFF-CORE, IFF-CAT, IFF-2CAT and IFF-DCAT.

The kernel (an outermost namespace) of the core meta-ontology IFF-CORE contains atomic namespaces for sets, functions, relations and finite limits. The core meta-ontology provides an adequate set-theoretic foundation for representing object-level ontologies in general and for defining other metalevel ontologies in particular. The core meta-ontology includes the specialization of the metashell namespace IFF-META. The IFF-CAT follows Mac Lane's beginning axiomatization for category theory [2] in that it introduces terminology and provides an axiomatization for this terminology, but it does not give a formal interpretation using set theory — it only gives an informal, intuitive interpretation. The IFF-CORE completes such a formal interpretation.

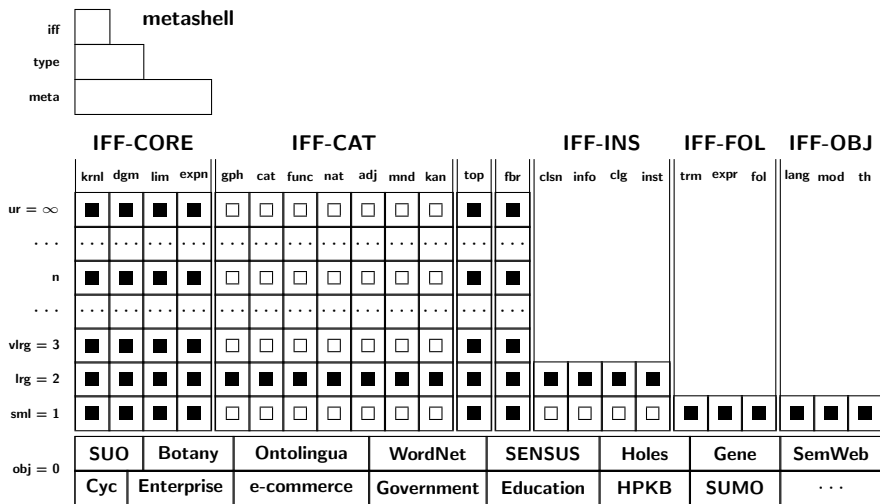
The IFF-CORE axiomatization contains many terms, partitioned according to whether the term is a basic term (sets, functions or relations), a diagram term or a limit term. Although the IFF-CORE axiomatizes finite limits, the current version does not represent finiteness, but instead explicitly represents the several finite shapes needed. A future version of the IFF-CORE may experiment with Dedekind's abstract definition of finiteness [1]. Following the principle of conceptual warrant, terminology has been placed in the core meta-ontology IFF-CORE only when it is needed in the more peripheral namespaces. All metalevel ontologies import and use, either directly or indirectly, the core meta-ontology. This includes the graph and category theory meta-ontologies.

The IFF-CORE axiomatization is in adjunctive form. The generic part plays a central role in the IFF axiomatization — this is currently the most referenced namespace at the metalevel. The kernel and finite limits namespace axiomatizations rely heavily upon the subset, (optimal-)restriction and abridgment relations for sets, (unary) functions and (binary) relations, respectively. The finite colimits namespace is a categorical dual to the finite limits namespace. The exponents namespace axiomatization is new.

[To Be Continued]



(iconic version)



(detailed version)

Figure 1: The IFF Architecture

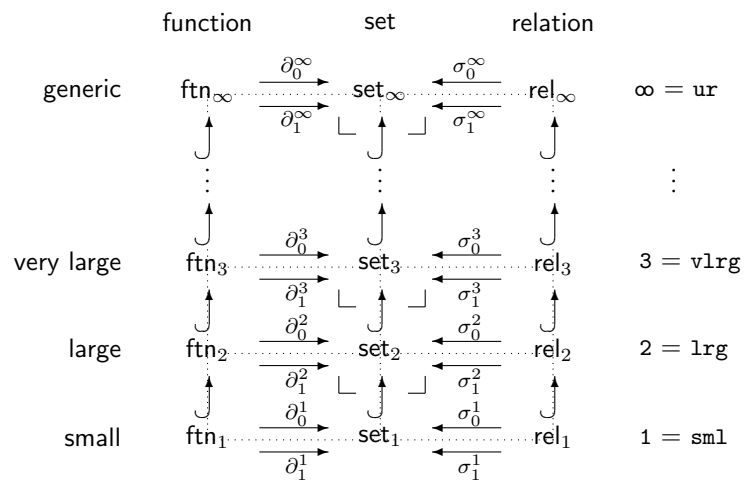


Figure 2: Metastack Structural Framework

3 Cantor's Diagonal Argument

Because of its importance in the IFF metastack, we discuss here in some detail Cantor's diagonal argument as presented in the book *Sets for Mathematics* by Lawvere and Rosbrugh [1]. As discussed in that book, in the nineteenth century George Cantor proved an important theorem using a diagonal argument, which implies the result

$$X < 2^X = \wp X$$

for all sets X^3 . The philosopher Bertrand Russell later used Cantor's diagonal argument to demonstrate the inconsistency of a system of logic proposed by Frege. Since then philosophers have referred to Cantor's theorem as Russell's paradox.

Definition 1 *Let Y be any object in a some category. An endomorphism $\tau : Y \rightarrow Y$ has an element $y : 1 \rightarrow Y$ as a fixed point when $y \cdot \tau = y$. The endomorphism $\tau : Y \rightarrow Y$ is fixed point free when no element of Y is a fixed point. The object Y has the fixed point property when no endomorphism $\tau : Y \rightarrow Y$ is fixed point free; that is, every endomorphism $\tau : Y \rightarrow Y$ has at least one fixed point.*

Although there are objects in the category of continuous spaces with the fixed point property (Brouwer: n -dimensional unit ball), most abstract sets do not have the fixed point property. We prove the contrapositive of Cantor's theorem.

Theorem 1 *Suppose there is a set X and a function $\phi : X \times X \rightarrow Y$ whose curry $\hat{\phi} : X \rightarrow Y^X$, where $\hat{\phi}(a) \doteq \phi(a, -)$ for all $a \in X$, is surjective; that is, such that for every function $f : X \rightarrow Y$ there is at least one element $a \in X$ such that $f = \hat{\phi}(a) = \phi(a, -)$. Then Y has the fixed point property.*

Proof: Consider any endofunction $\tau : Y \rightarrow Y$. We must show that τ has a fixed point. Using the diagonal function $\delta_X : X \rightarrow X \times X$, define the function $f = \delta_X \cdot \phi \cdot \tau : X \rightarrow X \times X \rightarrow Y \rightarrow Y$. In other words, for all $x \in X$, $f(x) = \tau\phi(x, x)$. By the assumption, $f = \phi(a, -)$ for some element element $a \in X$; that is, $f(x) = \phi(a, x)$ for some element element $a \in X$ and all elements $x \in X$. Hence, $\tau\phi(x, x) = \phi(a, x)$ for all $x \in X$. In particular, letting $x = a$, $\tau\phi(a, a) = \phi(a, a)$. Hence, $\phi(a, a) = \phi(\delta_X(a))$ is a fixed point of τ . ■

Corollary 1 (Cantor) *If a set Y has at least one endofunction $\tau : Y \rightarrow Y$ with no fixed points, then for every set X there is no surjection $X \rightarrow Y^X$.*

Corollary 2 *For any set X ,*

$$X < 2^X = \wp X.$$

³For sets X and Y , the inequality $X \leq Y$ means that there exists at least one injection from X to Y , and the strict inequality $X < Y$ means $X \leq Y$ and that no surjection $X \rightarrow Y$ exists.

Proof: Logical negation $\neg : 2 \rightarrow 2$ has no fixed point. Hence, there is no surjection $X \rightarrow 2^X = \wp X$. But, singleton $\{-\} : X \rightarrow 2^X = \wp X$ is injective. ■

Corollary 3 *There cannot exist a “universal set” U for which every set X is a subset $X \subseteq U$.*

Proof: If so, then the inclusion $X \hookrightarrow U$ is an injection, hence the exponent $2^U \rightarrow 2^X$ is a surjection. Define $X = 2^U$ to get a contradiction. ■

Corollary 4 *The set of all sets \mathbf{set} is not a set.*

Proof: If \mathbf{set} were a set, then $U = \bigcup \mathbf{set} = \bigcup \{X \mid X \in \mathbf{set}\}$, the union of all sets, would be a well-defined set. But, U would clearly be a “universal set”. Hence, the assumption that \mathbf{set} is a set leads to a contradiction. ■

Let us take inventory of the assumptions on sets that were needed to prove this. We have assumed that sets have: (**basic**) singleton maps, diagonal maps, local unions, the two-element (truth-value) set 2, logical negation; (**finite limits**) terminal elements, finite products; and (**exponents**) exponent sets, curry operators, exponent functions.

What is the meaning of “local union”. A local union operator at level n maps a level n set of level n sets to a level n set (consisting of all elements in at least one of the component sets), and hence is a function $\bigcup : \wp \mathbf{set}_n \rightarrow \mathbf{set}_n$, where $\wp \mathbf{set}_n = \{X \subset \mathbf{set}_n \mid X \in \mathbf{set}_n\}$. A local union is equivalent to an “indexed union”, which takes an level n indexed collection of level n sets $\{X_i \in \mathbf{set}_n \mid i \in I \in \mathbf{set}_n\}$ to a level n set. These notions are stronger than an ordinary “powerset union” operator $\bigcup : \wp \wp X \rightarrow \wp X$ for some level n set $X \in \mathbf{set}_n$.

Let \mathbf{set}_1 denote the collection of all sets. One consequence of Cantor’s diagonal argument ([1], [2]) is that this is not a set. We discuss Cantor’s diagonal argument in Appendix 3. In working practice, mathematicians distinguish between ordinary collections, such as the collection of all the current occupants of Paris or the collection of all of the stars in the Milky Way galaxy, from “larger” collections such as \mathbf{set}_1 . The latter kind of set is called a large set or a proper class, while the ordinary kind of set is called a small set or just set. Let \mathbf{set}_2 denote the collection of all classes. All small sets are classes $\mathbf{set}_1 \subset \mathbf{set}_2$, and the collection of all small sets is a class $\mathbf{set}_1 \in \mathbf{set}_2$, but there are some classes that are not small sets, such as $\mathbf{set}_1 \notin \mathbf{set}_1$. In practice, in particular in the application of category theory, mathematicians sometimes need to distinguish an even “larger” kind of collection, called a very large set or a proper collection. This results in the beginning of a chain: $\mathbf{set}_1 \subset \mathbf{set}_2 \subset \mathbf{set}_3 \dots$. This chain is at the heart of the IFF metastack, and results in the architecture visualized in Figure 2, which includes not only a chain of different kinds of sets, but also (using pullbacks, in particular optimal restriction on functions and abridgment on relations) corresponding chains of functions and (binary) relations. The supremum of all three chains are the generic sets, functions and relations axiomatized in the Ur meta-ontology. The necessary ingredients for describing the items in the Ur meta-ontology is axiomatized in the meta namespace.

4 Portals

4.1 First Order Logic

The first order logic (FOL) representation of object-level ontologies is of great interest in the online world. Here we discuss how the IFF handles this topic. Any standard representation mechanism, such as the Resource Description Framework (RDF), the Web Ontology Language (OWL), the Common Logic standard (CL), or the Conceptual Graphs standard (CG), will need to have a corresponding IFF module defined for transformation of logical expressions into the IFF. Such a module is called a *portal*. Currently, the IFF has documents describing tentative versions of portals for CGs, traditional logic and CL (this is not up-to-date, since the CL standard is currently under intense development). Clearly, portals need to be defined for RDF and OWL, and perhaps other standards such as the Suggested Upper Merged Ontology (SUMO), the formal specification notation Z based on set theory and first order predicate logic, the logical language created by Cycorp called CycL, etc.

The basic mechanism for the transformation of external logical expression into the IFF by means of a portal is relatively straightforward. Each standard will have a grammar, permitting any FOL logical expression to be parsed. Appropriate semantic actions can be associated with the production rules for such a grammar. Each parse tree resulting from the parse of any FOL expression (formula) will have an associated syntax tree for the expression. One set of semantic actions will specify the construction of this syntax tree. However, of much more interest to an IFF portal is the set of semantic actions that call upon appropriate elements of the IFF to build the internal representation of the expression. The transformation of an external logical expression has two phases: build the constituent internal term tuples, then build the internal expression.

Let $\mathbf{L} = \langle \text{var}(\mathbf{L}), \text{ent}(\mathbf{L}), \text{rel}(\mathbf{L}), \#_{\mathbf{L}}, \partial_{\mathbf{L}} \rangle$ denote any many-sorted FOL *language*⁴⁵ consisting of an adequate, possibly denumerable, set of indexing elements (*variables*) $\text{var}(\mathbf{L})$ ⁶, a set of sorts (*entity* types) $\text{ent}(\mathbf{L})$, a set of relation symbols (*relation* types) $\text{rel}(\mathbf{L})$, an *arity* function $\#_{\mathbf{L}} : \text{rel}(\mathbf{L}) \rightarrow \wp \text{var}(\mathbf{L})$ mapping relation symbols to the associated indicia⁷, and a *signature* function $\partial_{\mathbf{L}} : \text{rel}(\mathbf{L}) \rightarrow \text{tuple}[\text{var}(\mathbf{L}), \text{ent}(\mathbf{L})]$ mapping relation symbols to the associated tuple of indexed argument sorts, where $\#_{\mathbf{L}}(\rho) = \text{src}(\partial_{\mathbf{L}}(\rho))$ for every relation symbol $\rho \in \text{rel}(\mathbf{L})$ ⁸.

Term Tuples: Because we have ignored function symbols, term tuples are

⁴For simplicity we have omitted function symbols (*function* types) and the resulting Lawvere category of *term tuples*. When function symbols are included, term tuples replace substitutions.

⁵Languages are called *signatures* in the theory of institutions.

⁶This generalizes the usual case where sequences are used – the advantage for this generality is elimination of the dependency on sequences and natural numbers.

⁷An *indicia* X is a subset of indexing elements $X \subseteq \text{var}(\mathbf{L})$.

⁸For any relation symbol $\rho \in \text{rel}(\mathbf{L})$, an external arity form is $(\rho, x_1, x_2, \dots, x_n)$ where $\#_{\mathbf{L}}(\rho) = \{x_1, x_2, \dots, x_n\}$, and an external signature form is $(\rho, x_1:\alpha_1, x_2:\alpha_2, \dots, x_n:\alpha_n)$ where $\partial_{\mathbf{L}}(\rho) = (x_1:\alpha_1, x_2:\alpha_2, \dots, x_n:\alpha_n)$.

somewhat trivial. They are called substitutions. For any language \mathbf{L} , a *substitution* $h : \text{src}(h) \rightarrow \text{tgt}(h)$ ⁹ of \mathbf{L} is a surjective map between variable sets $\text{src}(h), \text{tgt}(h) \subseteq \text{var}(\mathbf{L})$ ¹⁰. Let $\text{subst}(\mathbf{L})$ denote the set of substitutions. There is a trivial Lawvere category $\text{Law}(\mathbf{L})$, whose object set is the set of indicia $\wp\text{var}(\mathbf{L})$ and whose morphism set is the set of substitutions $\text{subst}(\mathbf{L})$.

Expressions: An *expression* (formula) of \mathbf{L} is just like a relation symbol, it has an arity and a signature; in fact, as we discuss below, expressions are natural extensions to relations with each language inducing (and contained in) an associated expression language $\text{expr}(\mathbf{L}) = \langle \text{var}(\mathbf{L}), \text{ent}(\mathbf{L}), \text{expr}(\mathbf{L}), \#_{\text{expr}(\mathbf{L})}, \partial_{\text{expr}(\mathbf{L})} \rangle$. The set $\text{expr}(\mathbf{L})$ of expressions (formulas) of \mathbf{L} , along with its arity function $\#_{\text{expr}(\mathbf{L})} : \text{expr}(\mathbf{L}) \rightarrow \wp\text{var}$ and signature function $\partial_{\text{expr}(\mathbf{L})} : \text{expr}(\mathbf{L}) \rightarrow \text{tuple}[\text{var}(\mathbf{L}), \text{ent}(\mathbf{L})]$, can be inductively defined.

- An *atom* is a relational expression or substitutable pair (ρ, h) , consisting of a relation symbol ρ and a substitution h , where the relational arity is the substitution source¹¹. Thus, the set of atoms is defined to be $\text{atom}(\mathbf{L}) = \text{rel}(\mathbf{L}) \times_{\text{var}(\mathbf{L})} \text{subst}(\mathbf{L}) = \{(\rho, h) \mid \rho \in \text{rel}(\mathbf{L}), h \in \text{subst}(\mathbf{L}) \mid \#_{\mathbf{L}}(\rho) = \text{src}(h)\}$. The arity of an atom (ρ, h) is $\text{tgt}(h)$ and its signature is defined along the substitution map h . Note that the substitution trail remains in this notion of an atom: for example, the two marital relational expressions
 $(\text{married}, \text{man}:\text{Person}, \text{woman}:\text{Person})[\text{man} \mapsto \text{husband}, \text{woman} \mapsto \text{wife}]$
 $(\text{married}, \text{first}:\text{Person}, \text{second}:\text{Person})[\text{first} \mapsto \text{husband}, \text{second} \mapsto \text{wife}]$,
 which normally quotient to
 $(\text{married}, \text{husband}:\text{Person}, \text{wife}:\text{Person})$,
 are distinct, though logically equivalent, expressions. There is an injection $\iota_{\text{atom}} : \text{atom}(\mathbf{L}) \hookrightarrow \text{expr}(\mathbf{L})$.
- Let $\varphi \in \text{expr}(\mathbf{L})$ be any expression. The negation is an expression $\neg\varphi \in \text{expr}(\mathbf{L})$ with the same arity and signature. Define the set of negations as $\text{neg}(\mathbf{L}) = \{\neg\} \times \text{expr}(\mathbf{L}) \cong \text{expr}(\mathbf{L})$. There is an injection $\iota_{\neg} : \text{neg}(\mathbf{L}) \hookrightarrow \text{expr}(\mathbf{L})$ that differs from the above bijection.
- A pair of expressions is compatible when their signatures match on the overlap of their arities. The pullback of compatible pairs of expressions is $\text{expr}(\mathbf{L}) \otimes \text{expr}(\mathbf{L}) = \text{expr}(\mathbf{L}) \times_{\wp\text{var}(\mathbf{L})} \text{expr}(\mathbf{L}) = \{(\varphi, \psi) \mid \varphi, \psi \in \text{expr}(\mathbf{L}), (\forall x)(x \in \#_{\mathbf{L}}(\varphi) \cap \#_{\mathbf{L}}(\psi) \Rightarrow \partial_{\mathbf{L}}(\varphi)(x) = \partial_{\mathbf{L}}(\psi)(x))\}$. Let $\varphi \in \text{expr}(\mathbf{L})$ and $\psi \in \text{expr}(\mathbf{L})$ be any two compatible expressions. The conjunction, disjunction, implication and equivalence are expressions $(\varphi \wedge \psi), (\varphi \vee \psi), (\varphi \Rightarrow \psi), (\varphi \Leftrightarrow \psi) \in \text{expr}(\mathbf{L})$, whose arities are the union of those for φ and ψ and whose signatures are defined accordingly.

⁹An external form for h is $(x_1 \mapsto y_1, x_2 \mapsto y_2, \dots, x_n \mapsto y_n)$, where $\text{src}(h) = \{x_1, x_2, \dots, x_n\}$ and $y_1, y_2, \dots, y_n \in \text{tgt}(h)$.

¹⁰A *change-of-variables* is a substitution that is a bijection.

¹¹A suggestive external form is $(\rho, x_1:\alpha_1, x_2:\alpha_2, \dots, x_n:\alpha_n)[h]$ for an atom.

- Define the set of conjunctions as $\text{conj}(\mathbf{L}) = \{\wedge\} \times \text{expr}(\mathbf{L}) \otimes \text{expr}(\mathbf{L}) \cong \text{expr}(\mathbf{L}) \otimes \text{expr}(\mathbf{L})$. There is an injection $\iota_{\wedge} : \text{conj}(\mathbf{L}) \hookrightarrow \text{expr}(\mathbf{L})$.
 - Define the set of disjunctions as $\text{disj}(\mathbf{L}) = \{\vee\} \times \text{expr}(\mathbf{L}) \otimes \text{expr}(\mathbf{L}) \cong \text{expr}(\mathbf{L}) \otimes \text{expr}(\mathbf{L})$. There is an injection $\iota_{\vee} : \text{disj}(\mathbf{L}) \hookrightarrow \text{expr}(\mathbf{L})$.
 - Define the set of implications as $\text{impl}(\mathbf{L}) = \{\Rightarrow\} \times \text{expr}(\mathbf{L}) \otimes \text{expr}(\mathbf{L}) \cong \text{expr}(\mathbf{L}) \otimes \text{expr}(\mathbf{L})$. There is an injection $\iota_{\Rightarrow} : \text{impl}(\mathbf{L}) \hookrightarrow \text{expr}(\mathbf{L})$.
 - Define the set of equivalences as $\text{equiv}(\mathbf{L}) = \{\Leftrightarrow\} \times \text{expr}(\mathbf{L}) \otimes \text{expr}(\mathbf{L}) \cong \text{expr}(\mathbf{L}) \otimes \text{expr}(\mathbf{L})$. There is an injection $\iota_{\Leftrightarrow} : \text{equiv}(\mathbf{L}) \hookrightarrow \text{expr}(\mathbf{L})$.
- Any language \mathbf{L} has a set of cases or roles¹² that is the coproduct of its arity function (regarded as a discrete diagram of sets) $\text{case}(\mathbf{L}) = \sum \#_{\mathbf{L}} = \sum_{\rho \in \text{rel}(\mathbf{L})} \#_{\mathbf{L}}(\rho) = \{(\rho, x) \mid \rho \in \text{rel}(\mathbf{L}), x \in \#_{\mathbf{L}}(\rho)\}$. Let $(\varphi, x) \in \text{case}(\text{expr}(\mathbf{L}))$ be an expression case, consisting of an expression $\varphi \in \text{expr}(\mathbf{L})$ and a variable $x \in \#_{\text{expr}(\mathbf{L})}(\varphi)$ in its arity. The existential and universal quantifications are expressions $(\exists x)\varphi, (\forall x)\varphi \in \text{expr}(\mathbf{L})$, whose arities are the difference $\#_{\text{expr}(\mathbf{L})}((\exists x)\varphi) = \#_{\text{expr}(\mathbf{L})}((\forall x)\varphi) = \#_{\text{expr}(\mathbf{L})}(\varphi) - \{x\}$ and whose signatures are defined accordingly.
- Define the set of existentially quantified expressions as $\text{exist}(\mathbf{L}) = \{\exists\} \times \sum_{\varphi \in \text{expr}(\mathbf{L})} \#_{\text{expr}(\mathbf{L})}(\varphi) = \{\exists\} \times \text{case}(\text{expr}(\mathbf{L})) \cong \text{case}(\text{expr}(\mathbf{L}))$. There is an injection $\iota_{\exists} : \text{exist}(\mathbf{L}) \hookrightarrow \text{expr}(\mathbf{L})$.
 - Define the set of universally quantified expressions as $\text{forall}(\mathbf{L}) = \{\forall\} \times \sum_{\varphi \in \text{expr}(\mathbf{L})} \#_{\text{expr}(\mathbf{L})}(\varphi) = \{\forall\} \times \text{case}(\text{expr}(\mathbf{L})) \cong \text{case}(\text{expr}(\mathbf{L}))$. There is an injection $\iota_{\forall} : \text{forall}(\mathbf{L}) \hookrightarrow \text{expr}(\mathbf{L})$.

The set of expressions is formally defined as the fixpoint solution

$$\begin{aligned}
\text{expr}(\mathbf{L}) &\cong \text{atom}(\mathbf{L}) \\
&\quad + \text{neg}(\mathbf{L}) \\
&\quad + \text{conj}(\mathbf{L}) + \text{disj}(\mathbf{L}) + \text{impl}(\mathbf{L}) + \text{equiv}(\mathbf{L}) \\
&\quad + \text{exist}(\mathbf{L}) + \text{forall}(\mathbf{L}) \\
&\cong \text{atom}(\mathbf{L}) \\
&\quad + \{\neg\} \times \text{expr}(\mathbf{L}) \\
&\quad + \{\wedge, \vee, \Rightarrow, \Leftrightarrow\} \times \text{expr}(\mathbf{L}) \otimes \text{expr}(\mathbf{L}) \\
&\quad + \{\exists, \forall\} \times \text{case}(\text{expr}(\mathbf{L}))
\end{aligned}$$

This is a sum (disjoint union or coproduct) with sum (coproduct) projections being the above injections (Figure 3). Amongst many other things, the IFF Ontology (meta) Ontology (IFF-ONT) and the IFF First Order Logic (meta) Ontology (IFF-FOL) contain representations for

Languages: The original many-sorted FOL language \mathbf{L} , and the expression language $\text{expr}(\mathbf{L})$;

¹²The terminology of case relations or thematic roles is introduced in IFF languages for two reasons: to define the passage from languages (hypergraphs) to spangraphs; and for use in defining expressions.

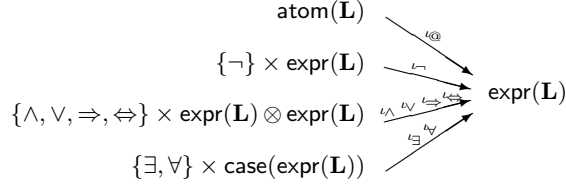


Figure 3: The set of expressions as a coproduct

Sets: Variables $\text{var}(\mathbf{L})$, indicia $\wp\text{var}(\mathbf{L})$, relation symbols $\text{rel}(\mathbf{L})$, function symbols $\text{ftn}(\mathbf{L})$, substitutions $\text{subst}(\mathbf{L})$, terms $\text{trm}(\mathbf{L})$, term tuples $\text{tpl}(\mathbf{L})$, relational expressions, equations $\text{eqn}(\mathbf{L})$, atoms $\text{atom}(\mathbf{L})$, negations $\text{neg}(\mathbf{L})$, conjunctions $\text{conj}(\mathbf{L})$, disjunctions $\text{disj}(\mathbf{L})$, implications $\text{impl}(\mathbf{L})$, equivalences $\text{equiv}(\mathbf{L})$, cases $\text{case}(\mathbf{L})$, existentially-quantified expressions $\text{exist}(\mathbf{L})$, universally-quantified expressions $\text{forall}(\mathbf{L})$ and expressions $\text{expr}(\mathbf{L})$;

Functions: The arity function $\#\mathbf{L} : \text{rel}(\mathbf{L}) \rightarrow \wp\text{var}(\mathbf{L})$, the signature function $\partial\mathbf{L} : \text{rel}(\mathbf{L}) \rightarrow \text{tuple}[\text{var}(\mathbf{L}), \text{ent}(\mathbf{L})]$, the atom injection $\iota_{\text{atom}} : \text{atom}(\mathbf{L}) \hookrightarrow \text{expr}(\mathbf{L})$, the negation injection $\iota_{\neg} : \text{neg}(\mathbf{L}) \hookrightarrow \text{expr}(\mathbf{L})$, the conjunction injection $\iota_{\wedge} : \text{conj}(\mathbf{L}) \hookrightarrow \text{expr}(\mathbf{L})$, the disjunction injection $\iota_{\vee} : \text{conj}(\mathbf{L}) \hookrightarrow \text{expr}(\mathbf{L})$, the implication injection $\iota_{\Rightarrow} : \text{conj}(\mathbf{L}) \hookrightarrow \text{expr}(\mathbf{L})$, the equivalence injection $\iota_{\Leftrightarrow} : \text{conj}(\mathbf{L}) \hookrightarrow \text{expr}(\mathbf{L})$, the existential injection $\iota_{\exists} : \text{exist}(\mathbf{L}) \hookrightarrow \text{expr}(\mathbf{L})$ and the universal injection $\iota_{\forall} : \text{forall}(\mathbf{L}) \hookrightarrow \text{expr}(\mathbf{L})$.

Many of these will be used to build an internal representation of an external expression. For example, consider the logical expression

$$\varphi = (\forall x : \text{object})(\forall y : \text{object})(\text{above}(x, y) \Leftrightarrow (\text{on}(x, y) \vee (\exists z)(\text{on}(x, z) \wedge \text{above}(z, y)))),$$

which renders the natural language expression

“An object X is above another object Y if X is directly above Y or else there is another object Z directly below X such that Z is above Y .”

Let us assume that φ is an expression in a many-sort FOL language $\mathbf{L} = \text{world}$, $\varphi \in \text{expr}(\text{world})$. Also, φ contains one entity type (sort) $\text{object} \in \text{ent}(\text{world})$, and two relation symbols $\text{on}, \text{above} \in \text{rel}(\text{world})$. The expression φ has the following constituents of various kinds.

abstraction	expression	abbreviation	kind
o	$object$		entity
α	$(on, 0:o, 1:o)$		relation
β	$(above, 0:o, 1:o)$		relation
h_1	$(0 \mapsto x, 1 \mapsto y)$		substitution
h_2	$(0 \mapsto x, 1 \mapsto z)$		substitution
h_3	$(0 \mapsto z, 1 \mapsto y)$		substitution
θ_1	$\iota_{@}(\alpha, h_1)$	$on(x, y)$	atom
θ_2	$\iota_{@}(\alpha, h_2)$	$on(x, z)$	atom
ψ_1	$\iota_{@}(\beta, h_1)$	$above(x, y)$	atom
ψ_2	$\iota_{@}(\beta, h_3)$	$above(z, y)$	atom
γ	$\iota_{\wedge}(\theta_2, \psi_2)$	$(on(x, z) \wedge above(z, y))$	conjunction
$\tilde{\gamma}$	$\iota_{\exists}(\gamma, z)$	$(\exists z)(on(x, z) \wedge above(z, y))$	existential
δ	$\iota_{\vee}(\theta_1, \tilde{\gamma})$	$(on(x, y) \vee (\exists z)(on(x, z) \wedge above(z, y)))$	disjunction
ε	$\iota_{\Leftrightarrow}(\psi_1, \delta)$	$(above(x, y) \Leftrightarrow (on(x, y) \vee (\exists z)(on(x, z) \wedge above(z, y))))$	equivalence
$\hat{\varepsilon}$	$\iota_{\forall}(\varepsilon, y)$	$(\forall y : object)(above(x, y) \Leftrightarrow (on(x, y) \vee (\exists z)(on(x, z) \wedge above(z, y))))$	universal
φ	$\iota_{\forall}(\hat{\varepsilon}, x)$	$(\forall x : object)(\forall y : object)(above(x, y) \Leftrightarrow (on(x, y) \vee (\exists z)(on(x, z) \wedge above(z, y))))$	universal

Clearly, the internal representation for the expression φ can be incrementally built from suitable functionality in an implementation of the meta-ontologies IFF-ONT and IFF-FOL. In order for this to be possible, various aspects of the terminology in these meta-ontologies must be suitably implemented.

5 Implementation

Here we begin a summary discussion of the IFF implementation. The IFF designer/axiomatizer can offer advice about the desirable *behavior* of the IFF implementation, but the IFF implementer should make the decisions about the proper *realization* of the IFF implementation.

In the large scope, the IFF has metalevels, namespaces and meta-ontologies. Namespaces are either atomic or molecular. Atomic namespaces (such as the object namespace for pullback diagrams with namespace prefix `dgm.pbk.obj`), are not further resolvable, whereas molecular namespaces, (such as the namespace for pullback diagrams with namespace prefix `dgm.pbk`, which groups together the atomic namespaces for pullback diagram objects and pullback diagram morphisms, or the namespace for diagrams with namespace prefix `dgm`, which groups together all molecular and atomic namespaces for diagrams of limits and colimits) group allied molecules and atoms. Namespaces and meta-ontologies are also called modules.

The metalevels are numbered with natural numbers: level 0 indexes the object level; level 1 and higher index meta levels, where level 1 indexes modules that involve only set-theoretically “small” structures, such as ordinary groups and group homomorphisms, topological spaces and continuous maps, vector spaces and linear transformations, classifications and infomorphisms, concept lattices and concept morphisms, etc; level 2 indexes modules that involve either set-theoretically “small” structures such as the above or set-theoretically “large” structures, such as large categories, functors between large categories, large classifications needed for institutions, etc; the Ur level contains all of the lower metalevels indexed by whole numbers. The Ur level and its subsumed lower metalevels are called the *natural part* of the IFF architecture. An important design goal: the natural part of the IFF should satisfy the categorical design principle — all axioms are atomic, being either relational expressions, equations or set declarations.

Just above the natural part of the IFF is the *meta namespace*. Full first order expression is allowed in the meta namespace, which services the natural part of the IFF. In order to declare and express the axiomatization of the meta namespace according to the IFF grammar, the *type namespace* is axiomatized above the meta namespace. The type namespace introduces the four architectural partial orders of subset, (optimal-)restriction and abridgment. The *iff namespace* is axiomatized above the type namespace, in order to introduce the three architectural kinds of set, (unary) function and (binary) relation. In reference to the IFF kernel architecture illustrated in Figure 2, the iff namespace corresponds to the horizontal structure and the type namespace corresponds to the vertical structure. The meta namespace, the type namespace and the iff namespace are closely related to the IFF grammar (see the IFF Syntax document), which specifies the correct form for IFF expressions.

Set, Function, Relation: The IFF grammar can be used to construct data structures or a “database” for the set-function-relation type aspect of the

IFF. This type aspect consists of the set, (unary) function and (binary) relation symbols, as illustrated in the mathematical terminology of the IFF metashell. All the type information in the metashell should be constructed by the implementation.

- Whenever an atomic expression (type declaration) in one of the forms

“(iff:set A)”,
“(type:set A)” or
“(meta:set A)”

is encountered during the parse, the term “A” always represents a set A at the corresponding metalevel.

- Whenever an atomic expression triple (type declaration) in one of the forms

“(iff:function f)” “(= (iff:source f) A)” “(= (iff:target f) B)”,
“(type:function f)” “(= (type:source f) A)” “(= (type:target f) B)” or
“(meta:function f)” “(= (meta:source f) A)” “(= (meta:target f) B)”

is encountered during the parse, the term triple (“f”, “A”, “B”) always represents a function $f : A \rightarrow B$ with source $\partial_0(f) = A$ and target $\partial_1(f) = B$ at the corresponding metalevel.

- Whenever an atomic expression triple (type declaration) in one of the forms

“(iff:relation r)” “(= (iff:set0 r) A0)” “(= (iff:set1 r) A1)”,
“(type:relation r)” “(= (type:set0 r) A0)” “(= (type:set1 r) A1)” or
“(meta:relation r)” “(= (meta:set0 r) A0)” “(= (meta:set1 r) A1)”

is encountered during the parse, the term triple (“r”, “A0”, “A1”) always represents a relation $r : A_0 \rightarrow A_1$ with domain $\sigma_0(r) = A_0$ and codomain $\sigma_1(r) = A_1$ at the corresponding metalevel.

Subset, (Optimal-)Restriction, Abridgement: The IFF grammar can be used to construct data structures or a “database” for the subset-restriction-abridgment type constraint aspect of the IFF. This type constraint aspect consists of the subset, (optimal) restriction and abridgment binary relationships between set, function and relation symbols, respectively. All the type constraint information in the metashell tables should be constructed by the implementation.

- Whenever an atomic expression (type constraint) in one of the forms

“(type:subset A B)” or
“(meta:subset A B)”

is encountered during the parse, the set A represented by the term “A” should always be regarded as a subset of the set B represented by the term “B”: $A \subseteq B$.

- Whenever an atomic expression (type constraint) in one of the forms

“(type:restriction f g)” or
“(meta:restriction f g)”

is encountered during the parse, the function $f : \partial_0(f) \rightarrow \partial_1(f)$ represented by the term “f” should always be regarded as a restriction of the function $g : \partial_0(g) \rightarrow \partial_1(g)$ represented by the term “g”: $f \sqsubseteq g$, with $\partial_0(f) \subseteq \partial_0(g)$, $\partial_1(f) \subseteq \partial_1(g)$ and $\partial_0(f) \subseteq g^{-1}(\partial_1(f))$.

- Whenever an atomic expression (type constraint) in one of the forms

“(type:optimal-restriction f g)” or
“(meta:optimal-restriction f g)”

is encountered during the parse, the function $f : \partial_0(f) \rightarrow \partial_1(f)$ represented by the term “f” should always be regarded as the optimal restriction of the function $g : \partial_0(g) \rightarrow \partial_1(g)$ represented by the term “g”: $f \dot{\sqsubseteq} g$, with $\partial_0(f) \subseteq \partial_0(g)$, $\partial_1(f) \subseteq \partial_1(g)$ and $\partial_0(f) = g^{-1}(\partial_1(f))$. Clearly, optimal restriction is a subrelation of restriction.

- Whenever an atomic expression (type constraint) in one of the forms

“(type:abridgment r s)” or
“(meta:abridgment r s)”

is encountered during the parse, the relation $r : \sigma_0(r) \rightarrow \sigma_1(r)$ represented by the term “r” should always be regarded as the abridgment of the relation $s : \sigma_0(s) \rightarrow \sigma_1(s)$ represented by the term “s”: $r \preceq s$, with $\sigma_0(r) \subseteq \sigma_0(s)$, $\sigma_1(r) \subseteq \sigma_1(s)$ and the extent of r the optimal restriction of the extent of s to the component sets of r , $\text{ext}(r) \dot{\sqsubseteq} \text{ext}(s)$. Pointwise, for all $x_0 \in \sigma_0(r)$ and $x_1 \in \sigma_1(r)$, $r(x_0, x_1)$ iff $s(x_0, x_1)$. This is a limit (multipullback) relationship between extents.

The four architectural endorelations, subset \subseteq , restriction \sqsubseteq , optimal-restriction $\dot{\sqsubseteq}$ and abridgment \preceq , are partial orders¹³. These partial orders, which include the corresponding partial orders at each metalevel, span all metalevels.

Type Correctness: The IFF grammar can be used for type correctness in set membership, function application and relation invocation: an error should be signaled,

¹³The *identity* endorelation $1_A : A \rightarrow A$ is defined by $(a, a') \in 1_A$ when $a = a'$. An arbitrary endorelation $r : A \rightarrow A$ is *reflexive* when it contains the identity endorelation: $1_A \subseteq r$. The *composition* of two relations $r : A \rightarrow B$ and $s : B \rightarrow C$ is the relation $r \circ s : A \rightarrow C$ defined by $(a, c) \in r \circ s$ when there exists an element $b \in B$ such that $(a, b) \in r$ and $(b, c) \in s$. An arbitrary endorelation $r : A \rightarrow A$ is *transitive* when it contains its square: $r \circ r \subseteq r$. The *transpose* of a relation $r : A \rightarrow B$ is the relation $r^{\text{op}} : B \rightarrow A$ defined by $(b, a) \in r^{\text{op}}$ when $(a, b) \in r$. An arbitrary endorelation $r : A \rightarrow A$ is *symmetric* when it contains its transpose: $r^{\text{op}} \subseteq r$. The *meet* of two relations $r : A \rightarrow B$ and $s : A \rightarrow B$ is the relation $r \cap s : A \rightarrow B$ defined by $(a, b) \in r \cap s$ when $(a, b) \in r$ and $(a, b) \in s$. An arbitrary endorelation $r : A \rightarrow A$ is *antisymmetric* when the meet of it and its transpose is contained in the identity: $r \cap r^{\text{op}} \subseteq 1_A$. An arbitrary endorelation $r : A \rightarrow A$ is a *preorder* when it is reflexive and transitive. A preorder $r : A \rightarrow A$ is a *partial order* when it is antisymmetric. An arbitrary endorelation $r : A \rightarrow A$ is an *equivalence relation* when it is reflexive, symmetric and transitive.

1. if a set symbol is being used as a function (in a grammatical term) or a relation (in a relational expression),
2. if a function symbol is being used as a set component for functions and relations or as a relation (in a relational expression), and
3. if a relation symbol is being used as a set component for functions and relations or as a function (in a grammatical term).

Well-formed-ness: The IFF grammar can and should be used to check for well-formed-ness of set membership, function application and relation invocation.

- If the symbol ‘X’ is declared to be a set

(set X)

then we use

(X x)

to express the statement that “‘x’ is a member of ‘X’.” Hence, ‘(X x)’ is a well-formed formula, which follows the prescription: *all IFF sets are unary*. Hence, the following nullary, binary, ternary and higher arity expressions are ill-formed

(X)

(X x0 x1)

(X x0 x1 x2)

...

- Like set membership, function application is represented in prefix notation. All functions are syntactically unary; *n*-ary functions are represented by using term tuple notation.

(= y (f x))

(= z (g [x y]))

If the symbol ‘f’ is declared to be an function with source set ‘X’ and target set ‘Y’

(function f)

(set X) (= (source f) X)

(set Y) (= (target f) Y)

then we use

(X x) (X y)

(= y (f x))

to express the equality statement that “‘y’ is equal to ‘f’ applied to ‘x’.” Hence, ‘(f x)’ is a well-formed term, which follows the prescription: *all IFF functions are unary*. Hence, the following nullary, binary, ternary and higher arity expressions are iff-formed

(= y (f))

(= y (f x0 x1))

(= y (f x0 x1 x2))

...

- Like set membership and function application, relation invocation is represented in prefix notation. All relations are binary.

(r x0 x1)

If the symbol ‘r’ is declared to be a relation with zeroth component set (domain) ‘X0’ and first component set (codomain) ‘X1’

```
(relation r)
(set X0) (= (set0 r) X0)
(set X1) (= (set1 r) X1)
```

then we use

```
(X0 x0) (X1 x1)
(r x0 x1)
```

to express the statement that “‘x0’ is ‘r’-related to ‘x1’.” Hence, ‘(r x0 x1)’ is a well-formed formula, which follows the prescription: *all IFF relations are binary*. Unary predicates are represented by IFF sets. However, the following nullary, unary, ternary and higher arity expressions are iff-formed

```
(r)
(r x0)
(r x0 x1 x2)
...
```

5.1 Data Structures

Here we summarize the databases or data structures (either internal or external) that are needed in the IFF implementation.

Sets and Component Functions:

- The set of all sets at any metalevel
 $\text{set} = \text{set}_{\text{iff}} \supset \text{set}_{\text{type}} \supset \text{set}_{\text{meta}} \supset \text{set}_{\text{ur}}$.
- The set of all (unary) functions at any metalevel
 $\text{ftn} = \text{ftn}_{\text{iff}} \supset \text{ftn}_{\text{type}} \supset \text{ftn}_{\text{meta}} \supset \text{ftn}_{\text{ur}}$, plus the source and target component maps $\text{src} = \partial_0$ and $\text{tgt} = \partial_1$, which form the span

$$\text{set} \xleftarrow{\partial_0} \text{ftn} \xrightarrow{\partial_1} \text{set}.$$

- The set of all (binary) relations at any metalevel
 $\text{rel} = \text{rel}_{\text{iff}} \supset \text{rel}_{\text{type}} \supset \text{rel}_{\text{meta}} \supset \text{rel}_{\text{ur}}$, plus the domain and codomain component maps $\text{set}_0 = \sigma_0$ and $\text{set}_1 = \sigma_1$, which form the span

$$\text{set} \xleftarrow{\sigma_0} \text{rel} \xrightarrow{\sigma_1} \text{set}.$$

Partial Orders

- The subset partial order on sets $\subseteq : \text{set} \rightarrow \text{set}$.
- The restriction partial order on functions $\sqsubseteq : \text{ftn} \rightarrow \text{ftn}$.
- The optimal restriction partial order on functions $\dot{\sqsubseteq} : \text{ftn} \rightarrow \text{ftn}$ with $\dot{\sqsubseteq} \subseteq \sqsubseteq$.
- The abridgment partial order on relations $\preceq : \text{rel} \rightarrow \text{rel}$.

References

- [1] F. William Lawvere and Robert Rosebrugh. *Sets for Mathematics*. Cambridge University Press, 2003.
- [2] Saunders Mac Lane. *Categories for the Working Mathematician*. Springer-Verlag, 1971.