The Portal of Tradition

`trad`

An IFF portal is point of entrance; it is a namespace that serves as an interface and is used for communication between an external representation and the IFF Model Theory Ontology.

The namespace of models uses terminology that is appropriate for a model theory based upon the theory of classifications. The resulting terminology in that namespace is somewhat abstract. The portal of tradition uses more elementary terminology than the namespace of models, and hopefully uses terminology that is closer to that of traditional model theory. One can view this traditional portal as a generic portal into the IFF Model Theory Ontology. Other portals, such as the portal of conceptual graphs, are more specifically oriented to a particular representation.

Table 1 lists the components of 1st order logic, both syntax and semantics. Non-constant function symbols are ignored in this table.

Table 1: Components of First Order Logic

LOGIC (SYNTAX)		MODEL THEORY (FORMAL SEMANTICS)	
1st order language	$\mathsf{L} = \langle \mathsf{C}, \mathsf{P} \rangle$	model for language L	$\mathsf{M} = \langle \mathsf{D}, \mathsf{I} \rangle$
		domain, universe	$dom(\mathsf{M}) = \mathsf{D}$
set of constants	$const(\mathsf{L}) = \mathsf{C}$	interpretation	$\mathsf{I} : const(\mathsf{L}) \to dom(\mathsf{M})$
set of predicates	$pred(\mathsf{L}) = \mathsf{P}$		$\mathsf{I} : pred(\mathsf{L}) \to \wp tuple(dom(\mathsf{M}))$
valence	$val_{\mathsf{L}} : pred(\mathsf{L}) \to Natno$		
set of formulas	$fmla(\mathsf{L})$	extended interpretation	$\mathsf{I} : fmla(\mathsf{L}) \to \wp tuple(dom(\mathsf{M}))$
set of sentences	$sent(\mathsf{L}) \subseteq fmla(\mathsf{L})$		$\mathsf{I} : sent(\mathsf{L}) \to truth = \wp 1 \cong \{\mathsf{f}, \mathsf{t}\}$
variable	$x \in var$		
constant	$c \in const(\mathsf{L})$	element, member	$\mathsf{I}[c] \in dom(\mathsf{M})$
predicate	$p \in pred(\mathsf{L}), val_{\mathsf{L}}(p) = n$	predicate, relation	$\mathsf{I}[p] \subseteq dom(\mathsf{M})^n$
formula	$\varphi \in fmla(\mathsf{L})$	predicate, relation	$\mathsf{I}[\varphi] \subseteq dom(\mathsf{M})^n$
sentence	$\sigma \in sent(\mathsf{L})$	truth value	$\mathsf{I}[\sigma] \in truth$
theory	$\Sigma \subseteq sent(\mathsf{L})$		
entailment	$\Sigma \vdash \sigma$		
		satisfaction	$\mathsf{M} \models \sigma$
theory of a model	$th(\mathsf{M})$		
models of a theory	$mod(\Sigma)$		

Any model $\boldsymbol{A}$ in the IFF Model Theory Ontology has an associated *traditional sorted model* of the type language $\boldsymbol{L} = typ(\boldsymbol{A})$. The following comments compare the components of a model $\boldsymbol{A}$ in the IFF Model Theory Ontology with both the interpretations in Table 1 and standard logic definitions (see Enderton) for many-sorted languages and logics.

Table 2 lists the terminology in the portal of tradition.

Table 2: The terminology of the tradition portal

	Class	Function
trad .lang	traditional-language	variable entity reference relation arity signature language
		expression sentence atom negation conjunction disjunction implication equivalence existential-quantification universal-quantification substitution
trad .mod	traditional-model	variable universe pair entity-type type-reference entity-extent entity reference tuple relation-type type-arity type-signature relation-extent relation signature model type
		satisfaction

Logic (Syntax)

trad.lang

The notion of a traditional 1st-order language in Table 1 corresponds to an IFF type language $\boldsymbol{L}$. There are two differences: the IFF type languages are sorted, whereas the languages described in Table 1 are unsorted; the languages in the current version of the IFF Model Theory Ontology do not have constants, function symbols or terms – the next version of the IFF Model Theory Ontology will incorporate these. Traditional languages will be mapped to IFF type languages below.

- A *traditional* (sorted) 1st-order *language* $\boldsymbol{L}$ has the following components.
  - a set of *variables* $\mathit{var}(\boldsymbol{L})$
  - a set of *entity types* $\mathit{ent}(\boldsymbol{L})$
  - a *reference* (sort) function $*_{\boldsymbol{L}} = \mathit{refer}(\boldsymbol{L}) : \mathit{var}(\boldsymbol{L}) \to \mathit{ent}(\boldsymbol{L})$

  Enderton's sort set S is represented by the set of entity types $\mathit{ent}(\boldsymbol{L}) = \mathit{tgt}(\mathit{refer}(\boldsymbol{L}))$ and his partition of sorted variables $\{\mathit{var}_s \mid s \in S\}$ is represented by the reference function $\mathit{refer}(\boldsymbol{L})$. The set of variables *var*, either in Table 1 or in Enderton, corresponds to the set of variables $\mathit{var}(\boldsymbol{L}) = \mathit{src}(\mathit{refer}(\boldsymbol{L}))$.

  Note, however that each language contains its own set of variables. Morphisms between languages have a component function between their respective variable sets. Although such variable functions are unconstrained for Language, the category of IFF type languages, they are constrained to be bijections for IFF model morphisms. We can have variable sets of any cardinality. However, two models are only connected by a morphism, if their variable sets have the same cardinality.

  The set of variables $\mathit{var}(\boldsymbol{L})$ generalizes the usual case where natural numbers and sequences are used – the advantage for this generality is elimination of the dependency on sequences and natural numbers. Sequence, along with other traditional data structures, will be represented by the IFF Data Structures Ontology in the lower metalevel. One way to return to sequence-like structures is to define a set of indexed variables $\mathit{var}(\boldsymbol{L}) = \mathit{index} \times \mathit{ent}(\boldsymbol{L})$ for some index set such as $\mathit{index} = \mathit{Natno}$, and to let the reference function be defined as the projection $\mathit{refer}(\boldsymbol{L}) = \pi : \mathit{index} \times \mathit{ent}(\boldsymbol{L}) \to \mathit{ent}(\boldsymbol{L})$.

```
(1) (SET$class traditional-language)

(2) (SET.FTN$function variable)
    (= (SET.FTN$source variable) traditional-language)
    (= (SET.FTN$target variable) set$set)

(3) (SET.FTN$function entity)
```

```
    (= (SET.FTN$source entity) traditional-language)
    (= (SET.FTN$target entity) set$set)

(4) (SET.FTN$function reference)
    (= (SET.FTN$source reference) traditional-language)
    (= (SET.FTN$target reference) set.ftn$function)
    (= (SET.FTN$composition [reference set.ftn$source]) variable)
    (= (SET.FTN$composition [reference set.ftn$target]) entity)
```

- Continuing, a *traditional* (sorted) 1st-order *language* $\boldsymbol{L}$ has the following additional components.
  - a set of *relation types* $\mathit{rel}(\boldsymbol{L})$
  - an *arity* function $\#_{\boldsymbol{A}} = \mathit{arity}(\boldsymbol{L}) : \mathit{rel}(\boldsymbol{L}) \to \wp \mathit{var}(\boldsymbol{L})$, or equivalently,
  - a *signature* function $\partial_{\boldsymbol{A}} = \mathit{sign}(\boldsymbol{L}) : \mathit{rel}(\boldsymbol{L}) \to \mathit{sign}(\mathit{refer}(\boldsymbol{L}))$

  where the signature of a relation type element $\rho \in \mathit{rel}(\boldsymbol{L})$ is the restriction of the reference function to its $\mathit{arity}(\boldsymbol{L})(\rho)$:

  $\mathit{sign}(\boldsymbol{L})(\rho)(x) = \mathit{refer}(\boldsymbol{L})(x)$ for all variables $x \in \mathit{arity}(\boldsymbol{L})(\rho)$.

  The set of predicates in Table 1 corresponds to the set of relation types $\mathit{rel}(\boldsymbol{L}) = \mathit{src}(\mathit{sign}(\boldsymbol{L}))$. The set of variables $\mathit{arity}(\boldsymbol{L})(\rho)$ that reference the entity types in the signature of a relation type $\rho \in \mathit{rel}(\boldsymbol{L})$ is called its arity. The arity of relation types can overlap. The number $|\mathit{arity}(\boldsymbol{L})(\rho)|$ of variables in the arity of a relation type is called its *valence*. This corresponds to the valence in Table 1.

  Enderton's (possibly empty) set of predicates of sort w for each tuple w of sorts corresponds to the (possibly empty) set $\mathit{sign}(\boldsymbol{L})^{-1}(\tau) \subseteq \mathit{rel}(\boldsymbol{L})$ of relation types of signature τ for each signature of entity types $\tau \in \mathit{sign}(\mathit{refer}(\boldsymbol{L}))$. Such a signature is a function $\tau : \mathit{arity}(\mathit{refer}(\boldsymbol{L}))(\tau) \subseteq \mathit{var}(\boldsymbol{L}) \to \mathit{ent}(\boldsymbol{L})$, which is a restriction of the reference function: $\tau(x) = \mathit{refer}(\boldsymbol{L})(x)$ for all $x \in \mathit{arity}(\mathit{refer}(\boldsymbol{L}))(\tau)$. Thus, the sort-indexed set of predicates corresponds to the signature function $\mathit{sign}(\boldsymbol{L})$. This function is equivalent to the arity function $\mathit{arity}(\boldsymbol{L})$.

```
(5) (SET.FTN$function relation)
    (= (SET.FTN$source relation) traditional-language)
    (= (SET.FTN$target relation) set$set)

(6) (SET.FTN$function arity)
    (= (SET.FTN$source arity) traditional-language)
    (= (SET.FTN$target arity) set.ftn$function)
    (= (SET.FTN$composition [arity set.ftn$source]) relation)
    (= (SET.FTN$composition [arity set.ftn$target])
       (SET.FTN$composition [variable set$power]))

(7) (SET.FTN$function signature)
    (= (SET.FTN$source signature) traditional-language)
    (= (SET.FTN$target signature) set.ftn$function)
    (= (SET.FTN$composition [signature set.ftn$source]) relation)
    (= (SET.FTN$composition [signature set.ftn$target])
       (SET.FTN$composition [reference set.ftn$signature]))

(8) (forall (?l (traditional-language ?l)
             ?rho ((relation ?l) ?rho))
        (and (= (set.ftn$composition [((signature ?l) ?rho) set.ftn$source])
                ((arity ?l) ?rho))
             (= (set.ftn$composition [((signature ?l) ?rho) set.ftn$target])
                (entity ?l))
             (set.ftn$restriction ((signature ?l) ?rho) (reference ?l))))
```

- Any traditional language $\boldsymbol{L}$ is an IFF type *language* $\mathit{lang}(\boldsymbol{L})$. This term marshals all the elements of a traditional language and maps into the type language namespace of the IFF Model Theory Ontology.

```
(9) (SET.FTN$function language)
    (= (SET.FTN$source language) traditional-language)
    (= (SET.FTN$target language) lang$language)
    (= (SET.FTN$composition [language lang$reference]) reference)
    (= (SET.FTN$composition [language lang$signature]) signature)
```

- For convenience of reference and access, additional terminology of a traditional language is given:

- the set of *expressions* $expr(\boldsymbol{L})$,
- the various kinds of expressions
  - *atomic* expression $atom(\boldsymbol{L})$,
  - *negations* $neg(\boldsymbol{L})$,
  - etcetra, and
- the expressions of empty arity called *sentences* $sent(\boldsymbol{L})$.

Formulas are called expressions in the IFF Model Theory Ontology. The set of formulas $fmla(\mathsf{L})$ in Table 1 corresponds to the set of expressions $expr(\boldsymbol{L}) = rel(expr(lang(\boldsymbol{L})))$, the relation types of the expression type language $expr(lang(\boldsymbol{L}))$ that extends the type language $lang(\boldsymbol{L})$ from relation types to expressions, but shares the same set of variables and the same set of entity types. To reiterate, in the IFF Model Theory Ontology expressions are relations types. Sentences are defined to be the set of expressions having empty arity $sent(\boldsymbol{L}) = arity(expr(lang(\boldsymbol{L})))^{-1}(\varnothing) \subseteq rel(expr(lang(\boldsymbol{L})))$. This is the $\varnothing$-fiber of the expression arity function

$$arity(expr(lang(\boldsymbol{L}))) : rel(expr(lang(\boldsymbol{L}))) \to \wp var(\boldsymbol{L}).$$

```
(10) (SET.FTN$function expression)
     (= (SET.FTN$source expression) traditional-language)
     (= (SET.FTN$target expression) set$set)
     (= expression (SET.FTN$composition [language lang.expr$set]))

(11) (SET.FTN$function atom)
     (= (SET.FTN$source atom) traditional-language)
     (= (SET.FTN$target atom) set$set)
     (= atom (SET.FTN$composition [language lang.expr$atom]))

(12) (SET.FTN$function negation)
     (= (SET.FTN$source negation) traditional-language)
     (= (SET.FTN$target negation) set$set)
     (= negation (SET.FTN$composition [language lang.expr$negation]))

(13) (SET.FTN$function conjunction)
     (= (SET.FTN$source conjunction) traditional-language)
     (= (SET.FTN$target conjunction) set$set)
     (= conjunction (SET.FTN$composition [language lang.expr$conjunction]))

(14) (SET.FTN$function disjunction)
     (= (SET.FTN$source disjunction) traditional-language)
     (= (SET.FTN$target disjunction) set$set)
     (= disjunction (SET.FTN$composition [language lang.expr$disjunction]))

(15) (SET.FTN$function implication)
     (= (SET.FTN$source implication) traditional-language)
     (= (SET.FTN$target implication) set$set)
     (= implication (SET.FTN$composition [language lang.expr$implication]))

(16) (SET.FTN$function equivalence)
     (= (SET.FTN$source equivalence) traditional-language)
     (= (SET.FTN$target equivalence) set$set)
     (= equivalence (SET.FTN$composition [language lang.expr$equivalence]))

(17) (SET.FTN$function existential-quantification)
     (= (SET.FTN$source existential-quantification) traditional-language)
     (= (SET.FTN$target existential-quantification) set$set)
     (= existential-quantification
        (SET.FTN$composition [language lang.expr$existential-quantification]))

(18) (SET.FTN$function universal-quantification)
     (= (SET.FTN$source universal-quantification) traditional-language)
     (= (SET.FTN$target universal-quantification) set$set)
     (= universal-quantification
        (SET.FTN$composition [language lang.expr$universal-quantification]))

(19) (SET.FTN$function substitution)
     (= (SET.FTN$source substitution) traditional-language)
```

```
        (= (SET.FTN$target substitution) set$set)
        (= substitution (SET.FTN$composition [language lang.expr$substitution]))

   (20) (SET.FTN$function sentence)
        (= (SET.FTN$source sentence) traditional-language)
        (= (SET.FTN$target sentence) set$set)
        (= sentence (SET.FTN$composition [language lang.expr$sentence]))
```

Model Theory (Formal Semantics)

`trad.mod`

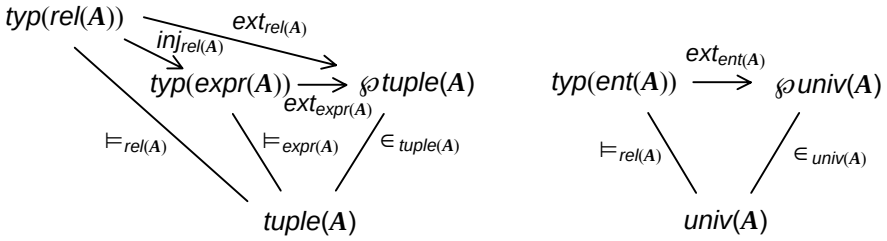

Figure 1: Entity, Relation and Expression Extent Infomorphisms

The extent infomorphisms (Figure 1) of the entity, relation and expression classifications reveal the various components of the interpretation functions of a traditional model ***A***.

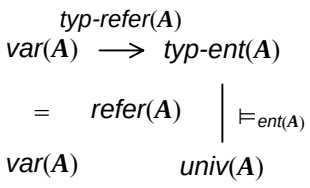

Figure 2a: Reference Semidesignation

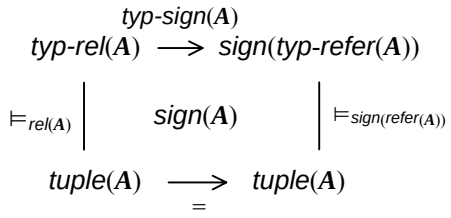

Figure 2b: Signature Designation

- A *traditional* (sorted) 1st-order *model* ***A*** has the following components.
  - a set of *variables* $var(\mathbf{A})$
  - a *universe* of discourse $univ(\mathbf{A})$
  - a *pair* of sets $pr(\mathbf{A}) = \langle var(\mathbf{A}), univ(\mathbf{A}) \rangle$
  - a set of *entity types* $ent\text{-}typ(\mathbf{A})$
  - a *reference* (sort) function $*_{\mathbf{A}} = typ\text{-}refer(\mathbf{A}) : var(\mathbf{A}) \to ent\text{-}typ(\mathbf{A})$
  - an *entity extent* function $ent\text{-}ext(\mathbf{A}) : ent\text{-}typ(\mathbf{A}) \to \wp univ(\mathbf{A})$, or equivalently,
  - an *entity* classification $ent(\mathbf{A}) = \langle univ(\mathbf{A}), ent\text{-}typ(\mathbf{A}), \models_{ent(\mathbf{A})} \rangle$, and
  - a *reference* semidesignation $refer(\mathbf{A}) = \langle pr(\mathbf{A}), typ\text{-}refer(\mathbf{A}) \rangle : var(\mathbf{A}) \leftrightarrow ent(\mathbf{A})$ (Figure 2a).

  The domain or universe $dom(\mathsf{M}) = \mathsf{D}$ in Table 1 corresponds to $univ(\mathbf{A}) = inst(ent(\mathbf{A}))$, the universe (or universal domain) for ***A,*** which is the set of instances of the entity classification $ent(\mathbf{A}) = inst(ent(\mathbf{A}))$. Enderton's set of sorts S corresponds to the set of entity types $ent\text{-}typ(\mathbf{A})$**,** which is the set of types of the entity classification $ent\text{-}typ(\mathbf{A}) = typ(ent(\mathbf{A}))$. Enderton's interpretation assignment of a set (intuitively, a type) T_s to each sort $s \in S$ is represented by the entity extent function $ent\text{-}ext(\mathbf{A})$, which is the extent function of the entity classification

  $$ent\text{-}ext(\mathbf{A}) = ext_{ent(\mathbf{A})} : typ(ent(\mathbf{A})) \to \wp univ(\mathbf{A}).$$

  This is the type function (Figure 1, right side) of the canonical eta infomorphism

  $$\eta_{ent(\mathbf{A})} = \langle id, ext_{ent(\mathbf{A})} \rangle : ent(\mathbf{A}) \rightleftarrows \wp univ(\mathbf{A}).$$

  But set-valued functions and classifications are equivalent, with each set-valued function inducing, by fiber or inverse image, a classification, and each classification having a set-valued extent function from its set of types to the power set of its set of instances. The induced classification of the set-valued en-

tity extent function is the entity classification $\mathit{induc}(\mathit{ent\text{-}ext}(\boldsymbol{A})) = \mathit{ent}(\boldsymbol{A})$. Both the entity classification and the universe instance power

$$\mathit{inst\text{-}pow}(\mathit{univ}(\boldsymbol{A})) = \langle \mathit{univ}(\boldsymbol{A}),\ \wp\, \mathit{univ}(\boldsymbol{A}),\ \in_{\mathit{univ}(\boldsymbol{A})} \rangle$$

have universe elements as their instances. The following axiomatization builds the reference semidesignation of Figure 2a.

```
(1) (SET$class traditional-model)

(2) (SET.FTN$function variable)
    (= (SET.FTN$source variable) traditional-model)
    (= (SET.FTN$target variable) set$set)

(3) (SET.FTN$function universe)
    (= (SET.FTN$source universe) traditional-model)
    (= (SET.FTN$target universe) set$set)

(4) (SET.FTN$function pair)
    (= (SET.FTN$source pair) traditional-model)
    (= (SET.FTN$target pair) set.pr$pair)
    (= (SET.FTN$composition [pair set.pr$set1]) variable)
    (= (SET.FTN$composition [pair set.pr$set2]) universe)

(5) (SET.FTN$function entity-type)
    (= (SET.FTN$source entity-type) traditional-model)
    (= (SET.FTN$target entity-type) set$set)

(6) (SET.FTN$function type-reference)
    (= (SET.FTN$source type-reference) traditional-model)
    (= (SET.FTN$target type-reference) set.ftn$function)
    (= (SET.FTN$composition [type-reference set.ftn$source]) variable)
    (= (SET.FTN$composition [type-reference set.ftn$target]) entity-type)

(7) (SET.FTN$function entity-extent)
    (= (SET.FTN$source entity-extent) traditional-model)
    (= (SET.FTN$target entity-extent) set.ftn$function)
    (= (SET.FTN$composition [entity-extent set.ftn$source]) entity-type)
    (= (SET.FTN$composition [entity-extent set.ftn$target])
       (SET.FTN$composition [universe set$power]))

(8) (SET.FTN$function entity)
    (= (SET.FTN$source entity) traditional-model)
    (= (SET.FTN$target entity) cls$classification)
    (= (SET.FTN$composition [entity cls$instance]) universe)
    (= (SET.FTN$composition [entity cls$type]) entity-type)
    (= entity (SET.FTN$composition [entity-extent set.ftn$induction]))
    (= entity-extent (SET.FTN$composition [entity cls$extent]))

(9) (SET.FTN$function reference)
    (= (SET.FTN$source reference) traditional-model)
    (= (SET.FTN$target reference) cls.sdsgn$semidesignation)
    (= (SET.FTN$composition [reference cls.sdsgn$set]) variable)
    (= (SET.FTN$composition [reference cls.sdsgn$classification]) entity)
    (= (SET.FTN$composition [reference cls.sdsgn$instance]) pair)
    (= (SET.FTN$composition [reference cls.sdsgn$type]) type-reference)
```

- Continuing, a *traditional* (sorted) 1st-order *model* $\boldsymbol{A}$ has the following additional components.
  - a set of *relation tuples* $\mathit{tuple}(\boldsymbol{A}) = \mathit{tuple}(\mathit{refer}(\boldsymbol{A}))$
  - a set of *relation types* $\mathit{rel\text{-}typ}(\boldsymbol{A})$
  - an *arity* function $\#_{\boldsymbol{A}} = \mathit{typ\text{-}arity}(\boldsymbol{A}) : \mathit{rel\text{-}typ}(\boldsymbol{A}) \to \wp\, \mathit{var}(\boldsymbol{A})$, or equivalently,
  - a *signature* function $\partial_{\boldsymbol{A}} = \mathit{typ\text{-}sign}(\boldsymbol{A}) : \mathit{rel\text{-}typ}(\boldsymbol{A}) \to \mathit{sign}(\mathit{typ\text{-}refer}(\boldsymbol{A}))$
  - a *relation extent* function $\mathit{rel\text{-}ext}(\boldsymbol{A}) : \mathit{rel\text{-}typ}(\boldsymbol{A}) \to \wp\, \mathit{tuple}(\boldsymbol{A})$, or equivalently,
  - a *relation* classification $\mathit{rel}(\boldsymbol{A}) = \langle \mathit{tuple}(\boldsymbol{A}),\ \mathit{rel\text{-}typ}(\boldsymbol{A}),\ \models_{\mathit{rel}(\boldsymbol{A})} \rangle$, and
  - a *signature* designation $\mathit{sign}(\boldsymbol{A}) = \langle \mathit{id}_{\mathit{tuple}(\boldsymbol{A})},\ \mathit{typ\text{-}sign}(\boldsymbol{A}) \rangle : \mathit{rel}(\boldsymbol{A}) \rightrightarrows \mathit{sign}(\mathit{refer}(\boldsymbol{A}))$ (Figure 2b).

The set $tuple(dom(\mathsf{M}))$ in Table 1 corresponds to $tuple(\boldsymbol{A})$, the set of tuples or (relational) arguments of (the reference semidesignation of) $\boldsymbol{A}$, which is the set of instances of the relation classification $tuple(\boldsymbol{A}) = inst(rel(\boldsymbol{A}))$. The set of predicates $pred(\mathsf{L})$ in Table 1, and Enderton's sort-indexed set of predicates, correspond to the set of relation types $rel\text{-}typ(\boldsymbol{A})$, which is the set of types of the relation classification $rel\text{-}typ(\boldsymbol{A}) = typ(rel(\boldsymbol{A}))$.

The predicate interpretation map $\mathsf{I} : pred(\mathsf{L}) \to \wp\, tuple(dom(\mathsf{M}))$ in Table 1, and Enderton's interpretation assignment of a subset of $T_{s1} \times \ldots \times T_{sn}$ to each predicate of sort $(s1, \ldots, s_n)$, are represented by the composition of the relation extent type function

$$rel\text{-}ext(\boldsymbol{A}) : typ(rel(\boldsymbol{A})) \to \wp\, tuple(\boldsymbol{A})$$

with the power of the instance signature function

$$\wp\, \partial_0 : \wp\, tuple(\boldsymbol{A}) \to \wp\, tuple(refer(\boldsymbol{A}))$$

and the power of the tuple inclusion function

$$\wp\, incl : \wp\, tuple(refer(\boldsymbol{A})) \to \wp\, tuple(univ(\boldsymbol{A})).$$

The relation extent function is the extent function of the relation classification

$$rel\text{-}ext(\boldsymbol{A}) = ext_{rel(\boldsymbol{A})} : typ(rel(\boldsymbol{A})) \to \wp\, tuple(\boldsymbol{A}).$$

This is the type function (Figure 1, left side) of the canonical eta infomorphism

$$\eta_{rel(\boldsymbol{A})} = \langle id, ext_{rel\boldsymbol{A}} \rangle : rel(\boldsymbol{A}) \rightleftarrows \wp\, tuple(\boldsymbol{A}).$$

Using again the equivalency between set-valued functions and classifications, the induced classification of the set-valued relation extent function is the relation classification $induc(rel\text{-}ext(\boldsymbol{A})) = rel(\boldsymbol{A})$. Both the relation classification and the tuple instance power

$$inst\text{-}pow(tuple(\boldsymbol{A})) = \langle tuple(\boldsymbol{A}),\ \wp\, tuple(\boldsymbol{A}),\ \in_{tuple(\boldsymbol{A})} \rangle$$

have tuples as instances.

The relation extent of any relation type $\rho \in rel\text{-}typ(\boldsymbol{A})$ is constrained to tuples $r \in tuple(\boldsymbol{A})$ that are more specialized than ρ:

$$r \in rel\text{-}ext(\boldsymbol{A})(\rho) \text{ implies } arity(pr(\boldsymbol{A}))(r) \supseteq typ\text{-}arity(\boldsymbol{A})(\rho).$$

The relation extent function respects arities (or signatures) in the sense that for any relation type $\rho \in rel\text{-}typ(\boldsymbol{A})$,

$$r \in rel\text{-}ext(\boldsymbol{A})(\rho) \text{ implies } r(x) \in ent\text{-}ext(\boldsymbol{A})(typ\text{-}refer(\boldsymbol{A})(x)) \text{ for all } x \in typ\text{-}arity(\boldsymbol{A})(\rho).$$

The latter constraint can be expressed in terms of the relation and entity classifications

$$r \models_{rel(\boldsymbol{A})} \rho \text{ implies } r(x) \models_{ent(\boldsymbol{A})} \partial_{\boldsymbol{A}}(\rho)(x) \text{ for all } x \in typ\text{-}arity(\boldsymbol{A})(\rho).$$

This is the constraint required of the signature designation. The following axiomatization builds the signature designation of Figure 2b.

```
(10) (SET.FTN$function tuple)
     (= (SET.FTN$source tuple) traditional-model)
     (= (SET.FTN$target tuple) set$set)
     (= tuple (SET.FTN$composition [reference cls.sdsgn$tuple]))

(11) (SET.FTN$function relation-type)
     (= (SET.FTN$source relation-type) traditional-model)
     (= (SET.FTN$target relation-type) set$set)

(12) (SET.FTN$function type-arity)
     (= (SET.FTN$source type-arity) traditional-model)
     (= (SET.FTN$target type-arity) set.ftn$function)
     (= (SET.FTN$composition [type-arity set.ftn$source]) relation-type)
     (= (SET.FTN$composition [type-arity set.ftn$target])
        (SET.FTN$composition [variable set$power]))

(13) (SET.FTN$function type-signature)
     (= (SET.FTN$source type-signature) traditional-model)
     (= (SET.FTN$target type-signature) set.ftn$function)
     (= (SET.FTN$composition [type-signature set.ftn$source]) relation-type)
     (= (SET.FTN$composition [type-signature set.ftn$target])
```

```
        (SET.FTN$composition [type-reference set.ftn$signature]))

(14) (SET.FTN$function relation-extent)
     (= (SET.FTN$source relation-extent) traditional-model)
     (= (SET.FTN$target relation-extent) set.ftn$function)
     (= (SET.FTN$composition [relation-extent set.ftn$source]) relation-type)
     (= (SET.FTN$composition [relation-extent set.ftn$target])
        (SET.FTN$composition [tuple set$power]))

(15) (forall (?a (traditional-model ?a) ?rho ((relation-type ?a) ?rho) ?r ((tuple ?a) ?r))
         (=> (((relation-extent ?a) ?rho) ?r)
             (and (set$subset
                     ((type-arity ?a) ?rho) ((cls.sdsgn$tuple-arity (reference ?a)) ?r))
                  (forall (?x (((type-arity ?a) ?rho) ?x))
                     (((entity-extent ?a) ((type-reference ?a) ?x)) (?r ?x))))))

(16) (SET.FTN$function relation)
     (= (SET.FTN$source relation) traditional-model)
     (= (SET.FTN$target relation) cls$classification)
     (= (SET.FTN$composition [relation cls$instance]) tuple)
     (= (SET.FTN$composition [relation cls$type]) relation-type)
     (= relation (SET.FTN$composition [relation-extent set.ftn$induction]))
     (= relation-extent (SET.FTN$composition [relation cls$extent]))

(17) (forall (?a (traditional-model ?a) ?rho ((relation-type ?a) ?rho) ?r ((tuple ?a) ?r))
         (=> ((relation ?a) ?r ?rho)
             (and (set$subset
                     ((type-arity ?a) ?rho) ((cls.sdsgn$tuple-arity (reference ?a)) ?r))
                  (forall (?x (((type-arity ?a) ?rho) ?x))
                     ((entity ?a) (?r ?x) ((type-reference ?a) ?x))))))

(18) (SET.FTN$function signature)
     (= (SET.FTN$source signature) traditional-model)
     (= (SET.FTN$target signature) cls.dsgn$designation)
     (= (SET.FTN$composition [signature cls.dsgn$source]) relation)
     (= (SET.FTN$composition [signature cls.dsgn$target])
        (SET.FTN$composition [reference cls.sdsgn$signature]))
     (= (SET.FTN$composition [signature cls.dsgn$instance])
        (SET.FTN$composition [tuple set.ftn$identity]))
     (= (SET.FTN$composition [signature cls.dsgn$type]) type-signature)
```

- Any traditional model $\boldsymbol{A}$ is an IFF *model* $\mathit{mod}(\boldsymbol{A})$. This term marshals all the elements of a traditional model and maps into the model namespace of the IFF Model Theory Ontology.

```
(19) (SET.FTN$function model)
     (= (SET.FTN$source model) traditional-model)
     (= (SET.FTN$target model) mod$model)
     (= (SET.FTN$composition [model mod$reference]) reference)
     (= (SET.FTN$composition [model mod$signature]) signature)
```

- The *type* aspect of any traditional model $\boldsymbol{A}$ is a traditional language $\boldsymbol{L}$. The IFF type language of the type aspect of any traditional model is the type language of the its IFF model.

```
(20) (SET.FTN$function type)
     (= (SET.FTN$source type) traditional-model)
     (= (SET.FTN$target type) traditional-language)
     (= (SET.FTN$composition [type trad.lang$variable]) variable)
     (= (SET.FTN$composition [type trad.lang$entity]) entity-type)
     (= (SET.FTN$composition [type trad.lang$reference]) type-reference)
     (= (SET.FTN$composition [type trad.lang$relation]) relation-type)
     (= (SET.FTN$composition [type trad.lang$arity]) type-arity)
     (= (SET.FTN$composition [type trad.lang$signature]) type-signature)

(21) (= (SET.FTN$composition [type trad.lang$language])
        (SET.FTN$composition [model mod$type]))
```

- For convenience of reference and access, additional terminology of a traditional model is given:
  - an *expression extent* function $\mathit{expr\text{-}ext}(\boldsymbol{A}) : \mathit{expr}(\boldsymbol{A}) \to \wp\, \mathit{tuple}(\boldsymbol{A})$, or equivalently,
  - an *expression* classification $\mathit{expr\text{-}cls}(\boldsymbol{A}) = \langle \mathit{tuple}(\boldsymbol{A}), \mathit{expr}(\boldsymbol{A}), \vDash_{\mathit{expr}(\boldsymbol{A})} \rangle$,

– a *satisfaction* relation (subset of satisfied sentences) $sat(\boldsymbol{A}) \subseteq sent(lang(\boldsymbol{A}))$.

The formula interpretation map $\mathsf{I} : fmla(\mathsf{L}) \to \wp tuple(dom(\mathsf{M}))$ in Table 1 is represented by the composition of the expression extent type function

$$ext_{expr\text{-}cls(\boldsymbol{A})} : typ(expr\text{-}cls(\boldsymbol{A})) \to \wp tuple(\boldsymbol{A})$$

with the power of the instance signature function

$$\wp \partial_0 : \wp tuple(\boldsymbol{A}) \to \wp tuple(refer(\boldsymbol{A}))$$

and the power of the tuple inclusion function

$$\wp incl : \wp tuple(refer(\boldsymbol{A})) \to \wp tuple(univ(\boldsymbol{A})).$$

The expression extent function is the extent function of the expression classification

$$expr\text{-}ext(\boldsymbol{A}) = ext_{expr\text{-}cls(\boldsymbol{A})} : typ(expr\text{-}cls(\boldsymbol{A})) \to \wp tuple(\boldsymbol{A}).$$

This is the type function (Figure 1, left side) of the canonical eta infomorphism

$$\eta_{expr\text{-}cls(\boldsymbol{A})} = \langle id, ext_{expr\text{-}cls(\boldsymbol{A})} \rangle : expr\text{-}cls(\boldsymbol{A}) \rightleftarrows \wp tuple(\boldsymbol{A}).$$

Using again the equivalency between set-valued functions and classifications, the induced classification of the set-valued expression extent function is the expression classification $induc(expr\text{-}ext(\boldsymbol{A})) = expr\text{-}cls(\boldsymbol{A})$. Both the expression classification and the tuple instance power

$$inst\text{-}pow(tuple(\boldsymbol{A})) = \langle tuple(\boldsymbol{A}), \wp tuple(\boldsymbol{A}), \in_{tuple(\boldsymbol{A})} \rangle$$

have tuples as instances.

The satisfaction relation $\mathsf{M} \models \sigma$ in Table 1 between traditional models and sentences is represented by the satisfies subset $sat(\boldsymbol{A}) \subseteq sent(lang(\boldsymbol{A}))$.

```
(22) (SET.FTN$function expression-extent)
     (= (SET.FTN$source expression-extent) traditional-model)
     (= (SET.FTN$target expression-extent) set.ftn$function)
     (= (SET.FTN$composition [expression-extent set.ftn$source]) expression)
     (= (SET.FTN$composition [expression-extent set.ftn$target])
       (SET.FTN$composition [(SET.FTN$composition [model mod$universe]) set$power]))
       (= expression-extent (SET.FTN$composition [model mod$extent]))

(23) (SET.FTN$function expression-classification)
     (= (SET.FTN$source expression-classification) traditional-model)
     (= (SET.FTN$target expression-classification) cls$classification)
     (= (SET.FTN$composition [expression-classification cls$instance]) tuple)
     (= (SET.FTN$composition [expression-classification cls$type])
        (SET.FTN$composition [type trad.lang$expression]))
     (= expression-classification
       (SET.FTN$composition [model mod$expression-classification]))

(24) (SET.FTN$function satisfies)
     (= (SET.FTN$source satisfies) traditional-model)
     (= (SET.FTN$target satisfies) set$set)
     (= satisfies (SET.FTN$composition [model mod$satisfies]))
```