Semantic Information Modeling for Federations Proposal

"There is a population explosion among the logical systems used in computer science. Examples include first order logic, equational logic, Horn clause logic, higher order logic, infinitary logic, dynamic logic, intuitionistic logic, order-sorted logic, and temporal logic; moreover, there is a tendency for each theorem prover to have its own idiosyncratic logical system. We introduce the concept of *institution* to formalize the informal notion of 'logical system'."

Institutions: Abstract Model Theory for Specification and Programming

Goguen and Burstall - 1992

6.5.1 SIMF Conceptual Model Concepts

We provide this section in (partial) satisfaction of the requirement to define a kernel of the Conceptual Domain Model. As stated in the Request For Proposals, the Conceptual Domain Model is not a model of a traditional domain like finance, strategy or human resources. Like a traditional domain model though, the SIMF Conceptual Domain Model is a statement of the terms and an explanation of the concepts in an area of concern, or domain. What's different is that the area of concern, or domain specified here describes the flow of information among diverse languages and logics. We infer this domain from the requirements of the Request For Proposals. The specified domain also includes the formalization of information flow as institutions. Institutions add a critical element to the domain. Institutions abstract notions of logical systems such that truth can be preserved under change of notation, both language and logic. The domain then is Information Flow and Institutions. (avoid not a domain model argument, semiotics as a domain)

Background and Motivation

The Request for Proposals identifies a wide array of languages whose use in independently conceived implementations require integration across both languages and logics. By integration we mean preserving satisfaction of constraints on a run-time system across languages and logics. In order to define a Conceptual Domain Model, or the kernel for such a model, we first observe that none of the languages cited in the Request for Proposals provide a solution to the problem statement. Thankfully though, a search of the literature shows both prior and ongoing evidence of the specification of the proposed domain in computer science, various consortia and standards organizations. We leverage selected aspects of that work for the (partial) specification of the kernel of our proposed Conceptual Domain Model. More specifically we leverage selected elements of Barwise and Seligman's Information Flow: the Logic of Distributed Systems, the Information Flow Framework, Goguen and Burstall's Institutions, and the Common Algebraic Specification Langage.

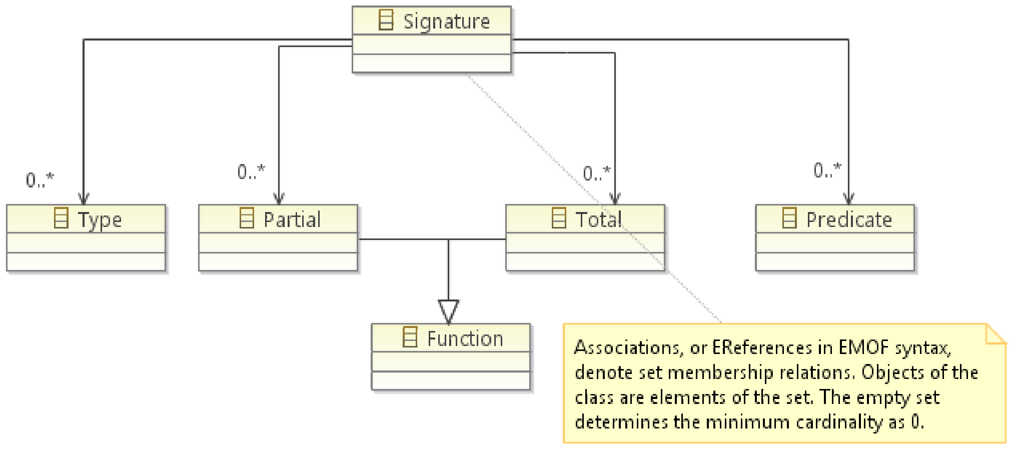
The approach we describe may seem unfamiliar to some whose interest and experience is specific to one or two languages, or logics. But, on further reflection you will find we simply apply the well known architectural principles of abstraction, composition and refinement to the problem statement in the Request For Proposals. At a later date we'll provide working examples of Information Flow and Institutions applied to the various languages cited in the Request For Proposals. (OK for this audience)

In the section that follows our approach is to present a concept by introducing a term in the domain followed by its definition. The definition for each term is presented with a syntax for its construction. Associated with each term we also present a diagram in a notation familiar to members of the Object Management Group. Following each diagram we present an explanation of the concept and the diagram. For the purposes of presenting a Conceptual Domain Model, we avoid making a commitment to an implementation in the Unified Modeling Language, or the Meta Object Facility (MOF), though an implementation of “the MOF” provided a convenient visualization and some handy syntax.

We make no claims at this point regarding completeness of the kernel of Conceptual Domain Model and owe many citations. Likewise, the diagrams should not be construed as models in the “formal sense.” For now, we choose to present just the concepts as required by the Request for Proposals and welcome feedback on this proposal.

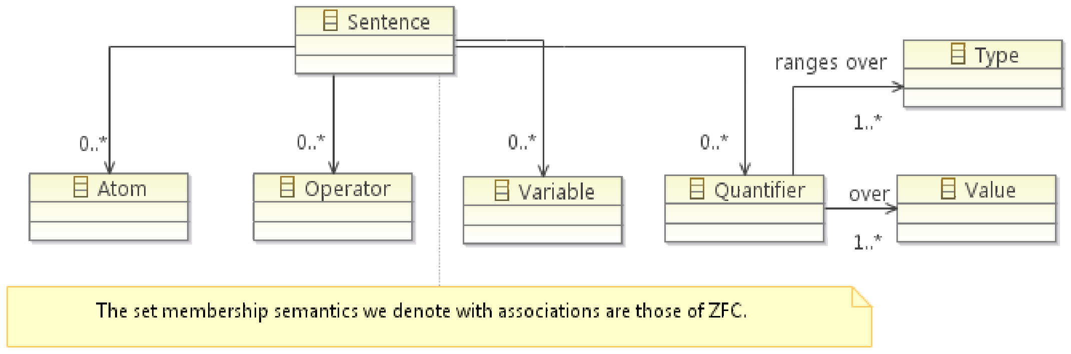
(Partial) Conceptual Domain Model Kernel

Signature: A signature $\Sigma = (\theta, \mathrm{TF}, \mathrm{PF}, P)$ consists of a set θ of types, sets of total functions TF and partial functions PF and a set of predicates P.


Signature
0..*
0..*
0..*
0..*
Type
Partial
Total
Predicate
Function
Associations, or EReferences in EMOF syntax, denote set membership relations. Objects of the class are elements of the set. The empty set determines the minimum cardinality as 0.

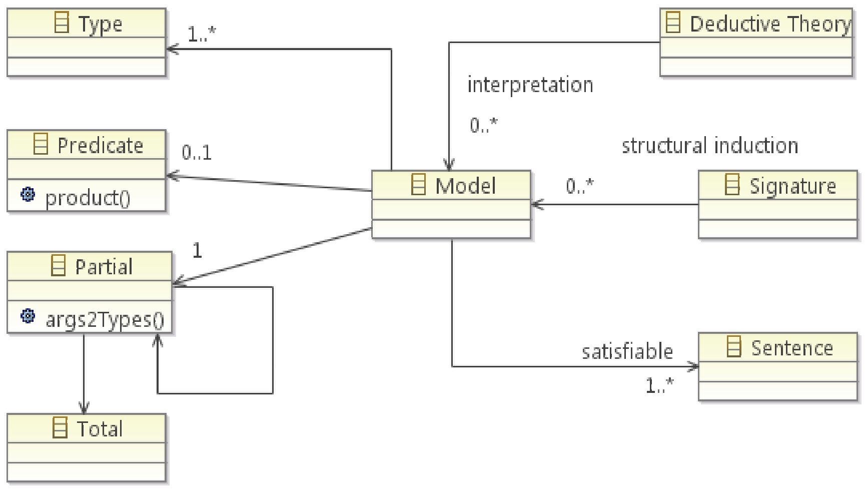
Explanation: We specify the signature of a language from its types, functions and predicates to abstract language-specific elements into the domain. We make no attempt to unify terms from all languages into the terms chosen for our conceptual domain model. (?sorts, types, classes, kinds?) Terms used to describe the elements for a specific language may differ from the terms we use to describe elements of the signature here. Notice the convention established with naming roles on association ends to represent set membership.

Sentence: A sentence $\Phi = (A, O, V, Q)$ consists of sets of atoms (A), logical operators (O), variables (V) and quantifiers (Q) both over variables as usual in first order logic, and types as in second order logic. Every sentence $\Phi \varepsilon \mathrm{Sen}(\Sigma)$ is a member of the set of sentences over the signature Σ.



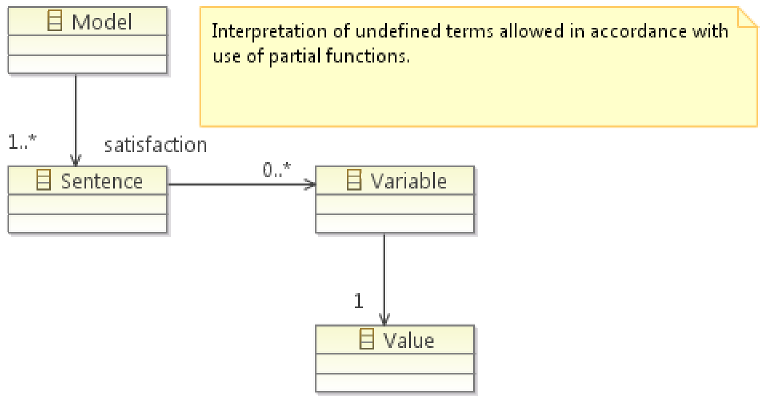
Explanation: As usual in first order logic, sentences are built from atoms and variables using quantifiers and the logical operators. We make no commitment to a specific set of operators. The set of sentences over the signature is defined by induction on the structure of the signature. (does this adequately explain the over relation?)

Model: A model $M = (\alpha^M, f^M, p^M)$ over a signature Σ consists of a non-empty (carrier) set α^M for each type $\alpha \,\varepsilon\, \theta$, a partial function f^M from w^M to α^M for each function symbol $f \,\varepsilon\, \mathrm{TF}_{w,\alpha}$ or $f \,\varepsilon\, \mathrm{PF}_{w,\alpha}$ and a predicate $p^M \subseteq w^M$ for each predicate symbol $p \,\varepsilon\, P_w$ where w is the cartesian product of argument types.



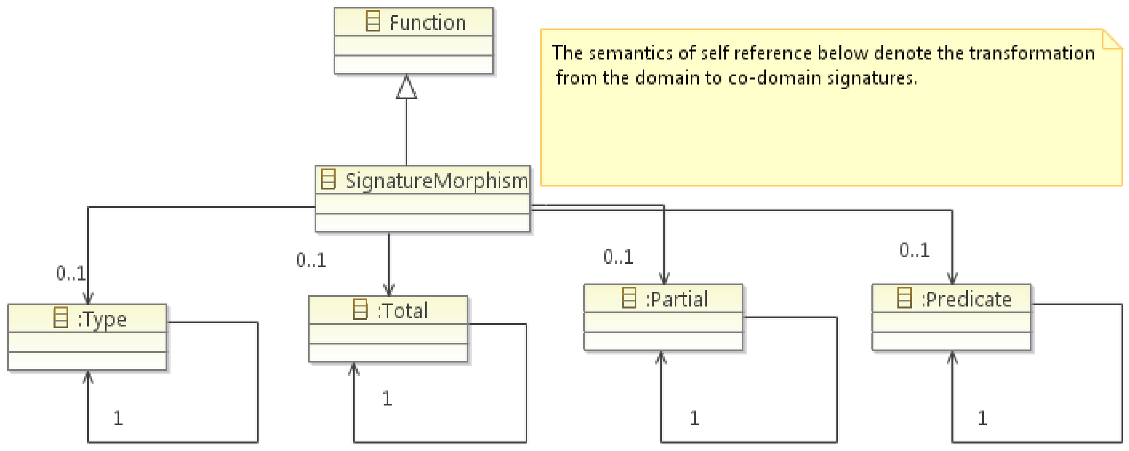
Explanation: A model is an interpretation of a deductive theory over the signature. The set of models over the signature is defined by induction on the structure of the signature. At least one sentence must be satisfiable in a model.

Satisfaction: The satisfaction $\vDash_S = (M, \Phi)$ of a sentence Φ in a model M is determined by the assignment of values to variables in atomic formulas. The satisfaction relation is defined inductively over the structure of sentences, in the usual way as done for first-order logic. Satisfaction of atomic sentences differs from first-order logic in the following way: interpretation of terms may be undefined due to the presence of partial functions; undefinedness is propagated from subterms to superterms. An equality holds if either both sides are undefined, or both sides are defined and equal. An application of a predicate to terms is true iff the interpretation of the terms are defined and yields a tuple in the relation interpreting the predicate.


Model
Interpretation of undefined terms allowed in accordance with use of partial functions.
1..*
satisfaction
Sentence
0..*
Variable
1
Value

Explanation: Partial functions allow for mapping a subset of a domain to its co-domain.

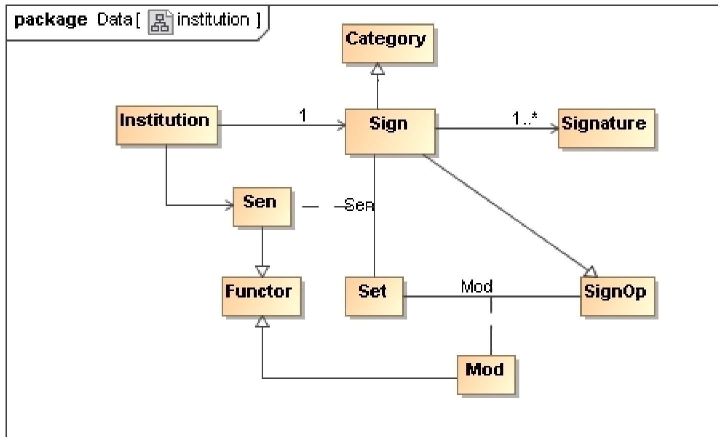
Signature Morphism: A signature morphism $\sigma : (\theta, \mathrm{TF}, \mathrm{PF}, P) \to (\theta', \mathrm{TF}', \mathrm{PF}', P')$ consists of a translation from θ to θ' and a translation between the corresponding sets of function and predicate symbols. The signature morphism $\sigma = \Sigma \to \Sigma'$ is also a translation function Sen (σ) on sentences, mapping Sen (Σ) to Sen (Σ') and a (reduct) function Mod (σ) on models, mapping Mod $(\Sigma')$to Mod (Σ).


Function
The semantics of self reference below denote the transformation from the domain to co-domain signatures.
SignatureMorphism
0..1
:Type
1
0..1
:Total
1
0..1
:Partial
1
0..1
:Predicate
1

Explanation: Satisfaction is preserved in the signature morphism. For the proof system to be sound, sentences inferred in the translation are always consequences of the premises. A partial function symbol may be mapped to a total function symbol, however a total function symbol may not be mapped to a partial function symbol. A signature morphism would be considered one of the model bridging relations in the Request for Proposals.

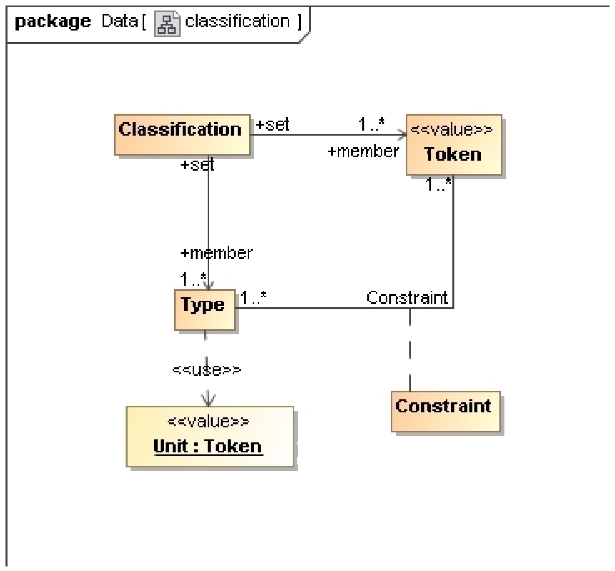
Institution: An institution $I = (\mathrm{Sign}^I, \mathrm{Sen}^I, \mathrm{Mod}^I, \models^I)$ consists of the following: A category Sign^I, the objects of which are signatures Σ. A functor $\mathrm{Sen}^I : \mathrm{Sign}^I \to \mathrm{Set}\ (\Phi)$, giving for each signature the set of sentences Sen (Σ); and for each signature morphism $\sigma = \Sigma \to \Sigma'$ the sentence translation map $\varphi = \mathrm{Sen}\,(\sigma) =$

$\mathrm{Sen}(\Sigma) \to \mathrm{Sen}\left(\Sigma'\right)$where often $\mathrm{Sen}^I(\sigma)(\varphi)$ is written as $\sigma(\varphi)$. A functor $\mathrm{Mod}^I : \mathrm{Sign}^{\mathrm{op}} \to \mathrm{Set}\,(M)$ giving for each signature Σ, the category of models $\mathrm{Mod}^I(M)$ and for each signature morphism $\sigma = \Sigma \to \Sigma'$, the reduct functor $\mathrm{Mod}^I(\sigma) = \mathrm{Mod}^I\left(\Sigma'\right) \to \mathrm{Mod}^I(\Sigma)$, where often $\mathrm{Mod}^I(\sigma)\left(M'\right)$ is written as $M'(\sigma)$. And a satifaction relation $\vDash^I_\Sigma \subseteq (\mathrm{Mod}^I(\Sigma), \mathrm{Sen}^I(\Sigma))$ for each $\Sigma\,\varepsilon\,\mathrm{Sign}^I$.



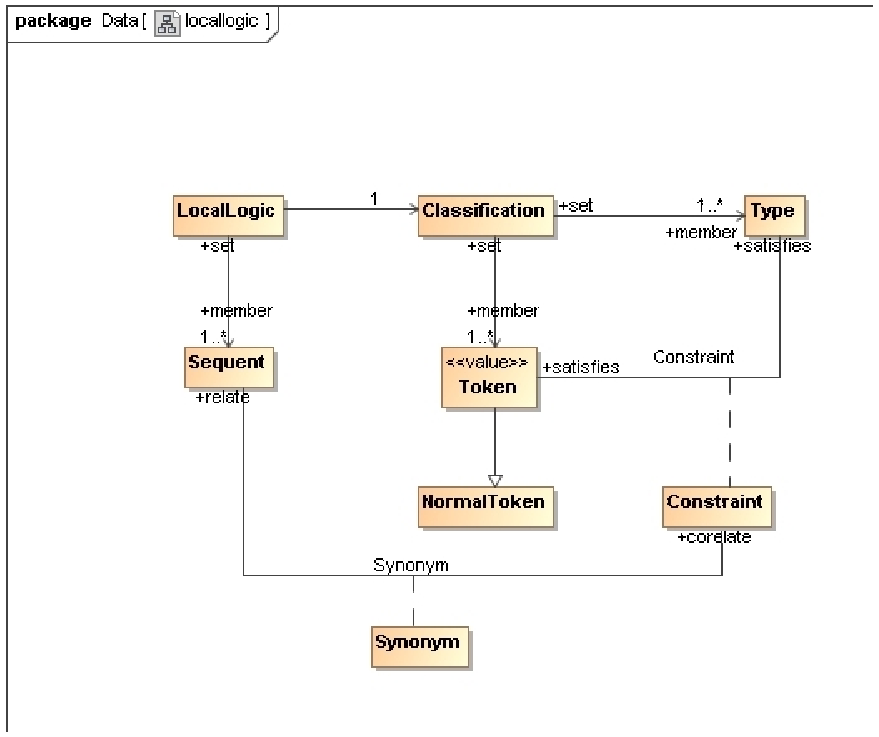
Explanation: Categories are mathematical abstractions composed of objects and mappings. Functors are mappings among categories and their components. Institutions use categories and functors to explain model bridging relations while under the stated constraints preserving satisfaction and allowing a change in language and logic. Reduct functors, sometimes called forgetful, map a structured object into the set of elements in a related structure without regard for the semantics of the related structure.

Classification: A classification $C = (A, \theta, \vDash_C)$ consists of a set A of objects to be classified called the tokens, or alternatively values of C, a set θ of objects used to classify the tokens called the types of C and a binary relation $\vDash_C$between A and θ that tells one which tokens are classified as being of which types. The unit value () denotes *undefined*. A classification may be said to be *proximal* or *distal* in relation to the core of an information channel defined below.



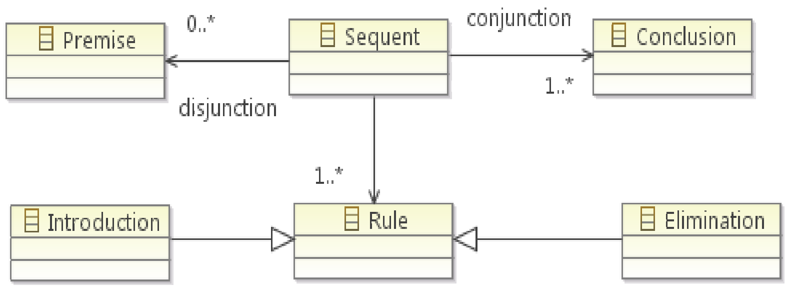
Explanation: A classification assigns token to types according to the notion of a constraint on the run-time system. A classification is a weaker notion than a signature. The classification lacks the total and partial functions and predicates of the signature. Information Flow and Institutions are presented separately without any attempt to unify the structures.

Local Logic: A local logic $L = (C, \vDash_L, N_L)$ consists of a classification C, a set $\vDash_L$ of sequents satisfying introduction and elimination rules on the types θ of C called the constraints of L, and a subset $N_L \subseteq A$ called the *normal tokens* of L which satisfy all the constraints of $\vDash_L$.



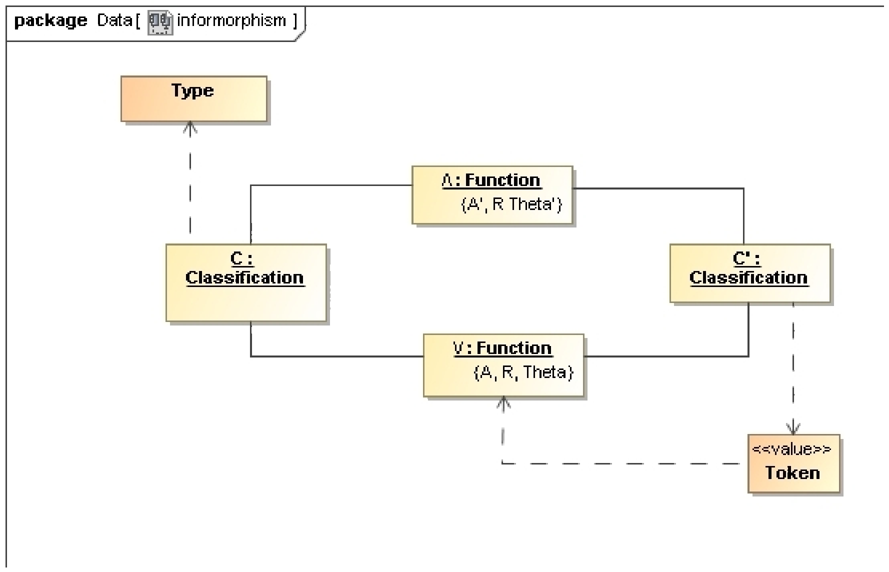
Explanation: Local logic is a term that reinforces the concept of a proximal logic introduced above. Local logics also introduce sequents as the mechanism to describe constraints on the run-time system. Local logics also introduce normal tokens and differentiate normal tokens from tokens. Normal tokens are tokens that satisfy the constraints on the system established through sequents.

Sequent: A sequent $S = (\Gamma, \Delta, \vDash)$ consists of a set of premises Γ, a set of conclusions Δ, and set of judgements $\vDash$, known as consequence relations. Sequents are specified through introduction and elimination rules as in natural deduction.



Explanation: The judgement in the sequent is interpreted that the conjunction of the premises imply the disjunction of the conclusion.

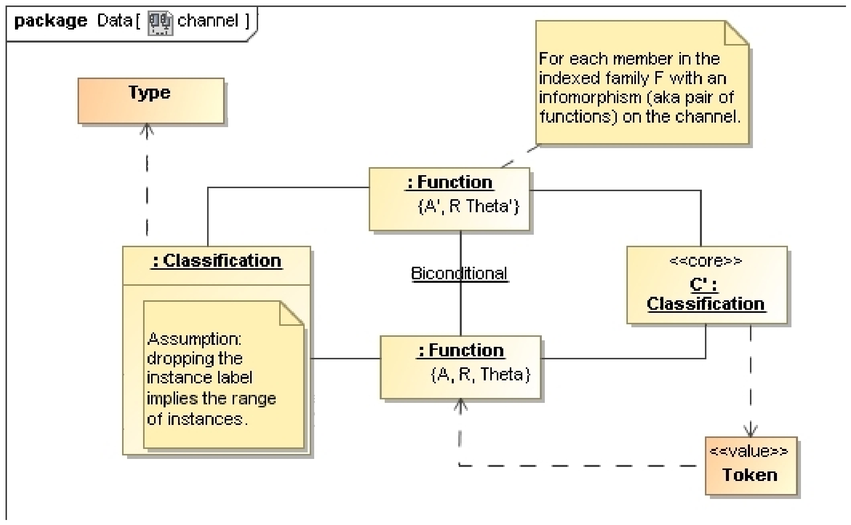
Infomorphism: If $C = (A, \theta, \vDash_C)$ and $C' = (A', \theta', \vDash_{C'})$ are classifications, then an infomorphism is a contravariant pair $f = (f^\wedge, f^\vee)$ of functions satisfying the biconditional constraint $f^\vee(a') \vDash_C \alpha$ iff $a' \vDash_{C'} f^\wedge(\alpha)$ for all tokens a' of C' and all types α of θ.



Explanation: The function with the up arrow maps types in classification C to types in classification C'. The function with the down arrow maps tokens in classification C' to tokens in classification C as illustrated in the diagram below.

$$\begin{tikzcd}
\theta \arrow[r, "f^\wedge"] \arrow[d, dash, "\vDash_C"'] & \theta' \arrow[d, dash, "\vDash_{C'}"] \\
A & A' \arrow[l, "f^\vee"]
\end{tikzcd}$$

Information Channel: An information channel consists of an indexed family $F = \{f_i : C_i \rightleftarrows C'\}_{i \varepsilon I}$ of infomorphisms with a common codomain C' called the core of the channel.


package Data[channel]
Type
For each member in the indexed family F with an infomorphism (aka pair of functions) on the channel.
: Function
{A', R Theta'}
: Classification
Biconditional
<<core>>
C' : Classification
Assumption: dropping the instance label implies the range of instances.
: Function
{A, R, Theta}
<<value>>
Token

Explanation: The information channel provides the central notion of how information flows in a distributed system according to infomorphisms indexed by the information channel with the common codomain.