A Formalization of Multi-level Zachman Frameworks

Richard Martin
Tinwisle Corp.
205 N College Ave
Bloomington IN 47404
tinwisle@cs.indiana.edu

Edward Robertson
Computer Science Dept.
Indiana University
Bloomington IN 47405
robertson@cs.indiana.edu

1. Introduction

This paper presents a formalism for the structure and semantics of Zachman frameworks. The ultimate goal is, of course, to formalize the framework development process; formalizing the artifacts produced by such development is an essential precursor for the process. Also, the Zachman Enterprise Architecture Model has matured to a point where formalization can enable further significant development.

In his *IBM Systems Journal* articles, John Zachman describes a Framework for Information Systems that defines an architectural structure for organizing institutional artifacts (1)(2). The expression of these artifacts reveals the interconnections and abstractions among structural elements and thereby facilitates the management of operational systems. In its latest form, “The Zachman Framework for Enterprise Architecture” presents an architectural model that is currently seeing more frequent use. The extensiveness and flexibility of the Zachman approach is demonstrated in a variety of applications from the design and construction of data warehouses(3), to use in the aerospace industry(6), to a GIS used in a regional planning context(8). The role of Zachman in academics is shown in the development of course syllabi (5) and text books(7). Descriptions of several reference architectures are influenced by Zachman, *e.g.* (4), and others seek to address the same range of issues in essentially the same structural manner (9).

Several reference architectures have been identified targeting manufacturing processes and enterprise operations. In a time when ERP systems are bringing large scale homogeneity to automated applications, the understanding of complex architectures becomes lost in the levels of detail while important levels of abstracted direction grow dormant.
⋆1⋆ (10)

One characteristic of the framework concept as set forth by Zachman is recursion, reflecting a top-down analysis of the world being modeling. Yet, like many situations in which recursive constructs could be of use, the tendency is to avoid those possibilities and the “excruciating levels of detail”, as Zachman says, that must be managed. As described in (3), recursion is an important part of Zachman’s approach and we believe an examination of its formal properties to be prescriptive.

Therefore a major tenet – and challenge – in our formalization is that a framework is a recursive artifact. In particular, a framework is defined as a structure composed of multiple frames. These frames are labeled in a top-down manner, but the semantics revealed by the framework work bottom-up.

This paper begins with a “bare bones” formalization of frames and frameworks. It

then describes how this formalism provides tools for building a framework reflecting the real world. This material is brought together in a detailed, specific example formalizing a slice of an enterprise architecture. Finally, we discuss using entire frameworks as situated
⋆2⋆ in the world and manipulations of these frameworks.

2. Formalization

This section first introduces the components used to describe and construct a framework. Then it discusses how a framework is composed of individual frames. Finally, it gives a formal definition of these frames.

A word about fonts used in this paper: the vocabulary used in describing a framework will be written in a small san-serif font. Later, when we talk about populating a framework with particular instance information, that information will be written in a `small fixed-width font`. This is analogous to relational databases, where we may talk of a date with value `July 4, 1776`.

2.1. Component Vocabulary

A framework's structure is always defined in terms of two fixed sets, $\mathbf{R}$ and $\mathbf{I}$, and a domain of discourse $\mathcal{D}$. When a frame is presented graphically, it is almost always as a table with columns corresponding to elements of $\mathbf{I}$ and rows corresponding to elements of
⋆3⋆ $\mathbf{R}$. Thus, for $i \in \mathbf{I}$ or $r \in \mathbf{R}$, we will write of "column i" or "row r", respectively. Path names constructed from the three sets label the individual frames of a framework.

R a fixed set of *roles*. $\mathbf{R}$ is ordered and, for $r \in \mathbf{R}$, r' indicates the successor of r in the order on $\mathbf{R}$.

Typical vocabularies used for $\mathbf{R}$ are {"conceptual", "logical", "physical"}, {"owner", "designer", "builder"}, and {"enterprise", "system", "technology"}. The vocabulary {"equity", "ecology", "economy"} has also been proposed. $\mathbf{R}$ is extended new with elements $\ominus$ and $\oplus$, respectively below and above all elements of $\mathbf{R}$. In Zachman terminology, $\ominus$ corresponds to "context" and $\oplus$ to "out of context". Separating $\mathbf{R}$ from $\ominus$ and $\oplus$ recognizes that the top and bottom rows of a typical Zachman frame picture are boundaries, which are important for passing information, rather than sites for analysis and further decomposition. Figure 1 is the typical way we draw an individual column, using a traditional set of role names; the distinction between cells the merely communicate and those that can be further decomposed is emphasized by drawing the former as ellipses and the later as rectangles. The fact that the ellipses are wider than the rectangles indicate that they are shared across all columns.

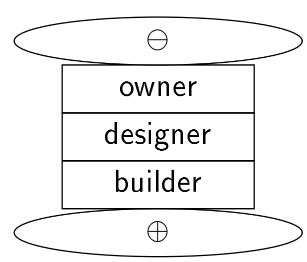

Figure 1: A Column

I a fixed set of *interrogatives*.

Typical vocabularies for $\mathbf{I}$ are {"what", "how", "where", "who", "when", "why"} or {"data", "function", "network", "people", "time", "motivation"}.

Any textual model is constructed with a set of names; following mathematical usage, we call this the *domain.* This term appears in common usage as well, as in the "domain of discourse".

$\mathcal{D}$ — a finite set, the *domain*

2.2. Paths and Labels

When we decompose a frame, we first form a grid of cells indexed by $\mathbf{R} \times \mathbf{I}$ and then designate components of a cell according to values in $\mathcal{D}$. Thus each frame is connected to its subframes by triples of the form $\langle \mathsf{r}, \mathsf{i}, \mathsf{d} \rangle$, which we call *edge labels.* A succession of edges forms a *path* and thus a sequence of edge labels is a *path label.*

$\mathcal{L}_1$ — $\{\langle r, i, d\rangle : r \in \mathbf{R},\ i \in \mathbf{I}, \text{ and } d \in \mathcal{D}\}$

$\mathcal{L}$ — a set of *path labels* defined by the BNF $\mathcal{L} := \epsilon | \mathcal{L}\, \mathcal{L}_1$, where ϵ denotes the empty (or root) path.

Since paths are typically more relevant, we will use the (unqualified) term "label" to refer to members of $\mathcal{L}$. In this paper, paths are typically denoted by lower-case Greek letters – α, β, γ, *etc.* Looking ahead, a crucial definition will be $\mathfrak{F}_\alpha$, the frame reached by the path labeled α.

In addition to sequences of edge labels that label frames, we will also need sequences of the form $\alpha\,\ell$, where $\alpha \in \mathcal{L}$ and $d \in \mathcal{D}$. This does not designate a (sub)frame and hence it is not a "label" in our nomenclature. Instead, $\alpha\, d$ makes specific the use of d in the frame labeled by α (that is, frame $\mathfrak{F}_\alpha$).

2.3. Frames and Frameworks

Frameworks are constructed recursively from frames indexed by $\mathbf{I} \times \mathbf{R} \times \mathcal{D}$. A frame's domain constrains the scope of consideration and allows a consistent treatment of artifacts within that domain. Our approach distinguishes two kinds of frames and relies on the
⋆4⋆ "context" and "out of context" linkage referenced in Zachman's approach.

$\mathfrak{F}$ — a *framework* is a finite set of *frames* $\{\mathfrak{F}_\alpha : \alpha \in \mathcal{L}\}$, where a frame $\mathfrak{F}_\alpha$ is either a XXX frame or a YYY frame, both of which are characterized below.

$Paths_{\mathfrak{F}}$ — $\{\alpha : \mathfrak{F}_\alpha \text{ is defined}\}$ for a framework $\mathfrak{F}$. $\mathfrak{F}$ is typically understood from context.

The labels of $\mathfrak{F}$ (*i.e.* $Paths$) must correspond to a tree structure, in which XXX frames are internal nodes and YYY frames are leaves. That is, all intervening nodes between a leaf and the root must be labeled by elements of $Paths$; formally, if $\alpha\,\ell \in Paths$, for $\alpha \in \mathcal{L}$ and $\ell \in \mathcal{L}_1$, then also $\alpha \in Paths$. However, there is no constraint that the actual frame $\mathfrak{F}_\alpha$ is distinct from $\mathfrak{F}_\beta$. That is, paths α and β may lead to the same frame instance. Thus the actual structure of $\mathfrak{F}$ is a labeled graph with nodes in $\{\mathfrak{F}_\alpha\}$ and edges labeled by $\mathcal{L}_1$. However, we require that the graph be acyclic; formally: $\mathfrak{F}_{\alpha\beta}$ is distinct from $\mathfrak{F}_\alpha$ unless β is the empty label ϵ.

With this notation, $\mathfrak{F}_\epsilon$ is the "root" frame, that is the broad, top-level frame that presents the entire framework to the outside world.

A *XXX frame* $\mathfrak{F}_\alpha$ is characterized by the 4-tuple:

$\mathcal{IC}_\alpha \subseteq \mathcal{D}$	*input (context)*
$\mathcal{OC}_\alpha \subseteq \mathcal{D}$	*output (out of context)*
$\mathcal{SF}_\alpha : \mathbf{R} \times \mathbf{I} \times \mathcal{D} \to (\mathfrak{F} \cup Paths)$	*subframe mapping*
$\Phi_\alpha \subseteq outputs \times inputs$ (details below)	*parameter passing relationship*

A *YYY frame* $\mathfrak{F}_\alpha$ is characterized by the 3-tuple:

$\mathcal{IC}_\alpha \subseteq \mathcal{D}$	*input (context)*
$\mathcal{OC}_\alpha \subseteq \mathcal{D}$	*output (out of context)*
$\mathfrak{S}_\alpha$ (details below)	*operations of the system modeled by the framework*

One YYY frame which commonly occurs at the end of many paths is the completely empty frame $\langle \emptyset, \emptyset, \emptyset \rangle$, denoted by $\mathfrak{E}$. There is no technical requirement that a frame be present and hence no specific need for $\mathfrak{E}$. However, $\mathfrak{E}$ can be used to indicate that a particular subframe is specifically known to be empty rather than merely overlooked.

A "dot" notation for subcomponents may also be used. Thus, for $r \in \mathbf{R}$, $i \in \mathbf{I}$, and $d \in \mathcal{D}$, the notations "$\mathfrak{F}_{\alpha\langle r,i,d\rangle}$", "$\mathcal{SF}_\alpha(r,i,d)$", and "$\mathfrak{F}_\alpha.\mathcal{SF}(r,i,d)$" are equivalent. We will often use $\mathcal{IC}$, $\mathcal{OC}$, Φ, *etc.* ambiguously for $\mathcal{IC}_\alpha$, $\mathcal{OC}_\alpha$, Φ_α, *etc.* respectively. The unqualified form could equivalently be formally defined, with $\mathcal{IC}$ and abbreviation for
⋆5⋆ $\bigcup_{\alpha \in Paths} \Phi_\alpha$ and so on.

Frames of both kinds also can have a variety of *annotations* as necessary. Such annotations are independent of the formalism. One common annotation fixes a single identifier to correspond to a path α – that is, it *names* the frame. We discuss annotations along
⋆6⋆ with abbreviations in a later section.

We have introduced the formal structure of both XXX and YYY frames at this point because it is important that both kinds of frames have $\mathcal{IC}$ and $\mathcal{OC}$ components.

2.4. XXX frame details

The $\mathcal{IC}$, $\mathcal{OC}$, and $\mathcal{SF}$ components reflect the recursive structure of a framework and as such are relatively simple and straightforward. In particular, $\mathcal{IC}$ and $\mathcal{OC}$ together correspond to the formal parameters of a procedure declaration. $\mathcal{SF}$ maps to either a subframe or a path, the latter to allow a form of indirection. That is, if $\mathcal{SF}(r,i,d) = \beta \in Paths$; this signifies a connection to $\mathfrak{F}_\beta$. The burden of describing how information passes from subframe to subframe falls on the relationship Φ. Because this information passing occurs in complex ways, the definition of Φ is messier. Note that Φ defines <u>only</u> interconnections; all decompositions and transformations are expressed in subframes, which means transformations ultimately appear only in YYY frames. Think of Φ as describing a plugboard or wiring patch panel.

The set of names $\mathcal{D}$ is global to an entire framework $\mathfrak{F}$. Because it is sometimes useful
⋆7⋆ to be able to talk about only those names used in one frame, we define

$$\mathcal{D}_\alpha \quad = \{d \in \mathcal{D} : \exists r \in \mathbf{R}, \exists i \in \mathbf{I} \quad (\mathcal{SF}_\alpha(r,i,d) \text{ is defined})\}$$

In order to set the stage for the definition of Φ, the definitions of $\mathcal{IC}$ and $\mathcal{OC}$ must be "extended" in two ways. First, each occurrence of elements of $\mathcal{D}$ is embellished with the name of the path in which it occurs (thus creating the variation of a path discussed at the end of subsection 2.1). Second, the $\mathcal{IC}$'s and $\mathcal{OC}$'s of a frame's subframes are collected across each entire row; this happens on a row-by-row basis because the information "flow" through a frame is also row-by-row. Formally, for $r \in \mathbf{R}$,

$$\mathcal{EIC}_{\alpha,r} = \{\alpha\langle r,i,d\rangle\, d' : i \in \mathbf{I} \;\&\; d \in \mathcal{D} \;\&\; d' \in \mathcal{IC}_{\alpha\,\langle r,i,d\rangle}\}$$

$$\mathcal{EOC}_{\alpha,r} = \{\alpha\langle r,i,d\rangle\, d' : i \in \mathbf{I} \;\&\; d \in \mathcal{D} \;\&\; d' \in \mathcal{OC}_{\alpha\,\langle r,i,d\rangle}\}$$

Furthermore, as a notational convenience, which allows us to treat boundary transitions (from "context" and to "out of context") with the same mechanism used for internal transitions, define

$$\mathcal{EOC}_{\alpha,\ominus} = \{\alpha\, d : d \in \mathcal{IC}_\alpha\}$$

$$\mathcal{EIC}_{\alpha,\oplus} = \{\alpha\, d : d \in \mathcal{OC}_\alpha\}$$

Finally we may formally define Φ:

$$\Phi_\alpha \quad \subseteq \bigcup_{r \in \mathbf{R} \cup \{\ominus\}} \big(\mathcal{EOC}_{\alpha,r} \times \mathcal{EIC}_{\alpha,r'}\big)$$

Note that the definition of Φ is a union over products, not a product of unions. The notation $r \in \mathbf{R} \cup \{\ominus\}$ selects a single role interface, between r and r', and constrains Φ to work across that interface. There are two boundary conditions at the least and greatest elements of the order on $\mathbf{R}$: the first is when r (in the expression defining Φ_α) is $\ominus$ and r' is the least element in $\mathbf{R}$; the second is when r is the greatest element in $\mathbf{R}$ and r' is $\oplus$. In the first case, the above notation allows us to substitute $\mathcal{EIC}_\alpha$ for $\mathcal{EOC}_{\alpha,\ominus}$, so that $\mathcal{EOC}_{\alpha,\ominus} \times \mathcal{EIC}_{\alpha,r_0}$ becomes $\mathcal{EIC}_\alpha \times \mathcal{EIC}_{\alpha,r_0}$, where r_0 is the first element of $\mathbf{R}$. Thus the inputs to a frame are connected into its subframes by the same mechanisms that connects outputs from one subframe to be inputs to the next. In the second case, the product is $\mathcal{EOC}_{\alpha,r_\ell} \times \mathcal{EOC}_\alpha$, where r_ℓ is the last element of $\mathbf{R}$. Figure 2 illustrates how Φ is built out
$\star 8 \star$ of its various pieces. The rows are labeled $r_0, \cdots, r_\ell$ to indicate the generality of $\mathbf{R}$.

The net impact of the definition of Φ as discussed above is that Φ is defined so that it operates across columns but only from one row to the next.

2.5. YYY frame details

Frameworks allow us to model the structure of a world of discourse. It is still necessary to populate this structure with the items from the world. Thus we distinguish between
$\star 9 \star$ the *model* and the *instance*. This distinction is exactly paralleled in such areas as relational databases, which commonly appear in the "data" column of frames. The model, or "schema", of a relation is a structure, such as

```
Employee:  SSN, name, birthdate, ...
```

An instance of that relation is a set of tuples (records) which the relation contains

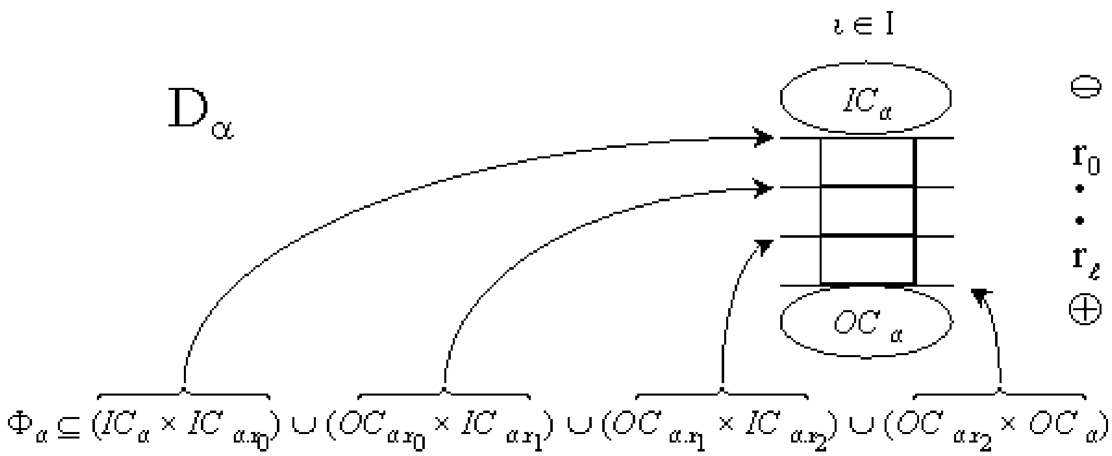

Figure 2: How Φ Relates to Frames

123-45-6789	Bob Smith	7 Aug 1954	
478-22-1836	Sue Davis	13 May 1962	$\cdots$

Note that a given relation has one fixed schema but infinitely many possible instances.

Because we may wish to deal, at the instance level, with individual items or sets of items, we use the notation *Types* to distinguish how we deal with instances.

$$Types_\alpha \quad = \mathcal{IC}_\alpha \cup \{\text{“SET OF } \iota\text{”} : \iota \in \mathcal{IC}\}$$

The definition $\mathcal{S}_\alpha$ completes the definition of YYY frames.

$$\mathcal{S}_\alpha \qquad \subseteq \bigcup_{n \in \mathbb{N}} {Types_\alpha}^n$$

The *Types* are the primary linkage between a framework as a model and an instance
⋆10⋆ of that framework. When a framework is instantiated, the signatures[1] in the various $\mathcal{S}_\alpha$ become the operations which do the work of the system. The *Types* are thus analogous to table schema in a relational database system, as opposed to the records in those tables which are the instances.

We use $\mathfrak{I}$ for any YYY frame which merely connects its input to its output – that is, $\mathcal{IC} = \mathcal{OC}$ and $\mathcal{S}$ is the identity. Formally, such usage is ambiguous because there is an identity frame for each distinct $\mathcal{IC}$, $\mathcal{OC}$ pair but practically it is easy to handle.

3. Building on the Frameworks

1. A signature is an ordered list of the types that participate in a function or relation. Thus one signature describes the “input/output format” of a family of relations. For example, addition, subtraction, multiplication and division have the same signature, namely two numbers in and one returned. In a relation database, the signature of a relation is just its schema.

3.1. Abbreviations, Annotations, and Indirections

We have already noted that a path may be abbreviated by a single name. There are several other conventions that help the presentation and comprehension of a framework.

First, unique and meaningful names may be given to each cell of the $\mathbf{R} \times \mathbf{I}$ grid that structures a frame. For example, in Zachman's "Enterprise Architecture", the conceptual,data cell is called the "Semantic Model" and the conceptual,motivation cell is the "Business Plan".

Second, there are names associated with the contents of a cell labeled by an $\mathbf{R}$, $\mathbf{I}$ pair. Thus an object within the conceptual,data cell (that is, any object with path name ⟨conceptual, data, —⟩) is given the name "Entity" or "Relationship", while "Semantic Model" designates the entire collection of objects in that cell.

Third, the $\mathbf{R}$ and $\mathbf{I}$ components of a collection of paths are often separated from the $\mathcal{D}$ components. This is especially powerful when combined with the naming of cells in the $\mathbf{R} \times \mathbf{I}$ grid. This is particularly useful when there are many paths that all have the same form, as is true with ⟨conceptual, data, —⟩ ⟨conceptual, data, —⟩. We call the conceptual,data cell "Relation" at the first level and "attribute" at the second. Thus the statement "Relation.attribute includes Employee.SSN and Employee.name" means that there is a frame with label "⟨conceptual, data, Employee⟩⟨conceptual, data, SSN⟩" and one with "⟨conceptual, data, Employee⟩ ⟨conceptual, data, name⟩". This example is further elaborated in section 4.

Fourth, if a name in $\mathcal{D}$ appears in only one frame, then entire paths are abbreviated. Say that d belongs uniquely to $\mathcal{D}_\alpha$, then $\langle r, i, d\rangle$ can be used in place of $\alpha\langle r, i, d\rangle$. Again this works well with other abbreviations. For example, if Employee appears only at the top level, then Entity Employee indicates a unique subframe.

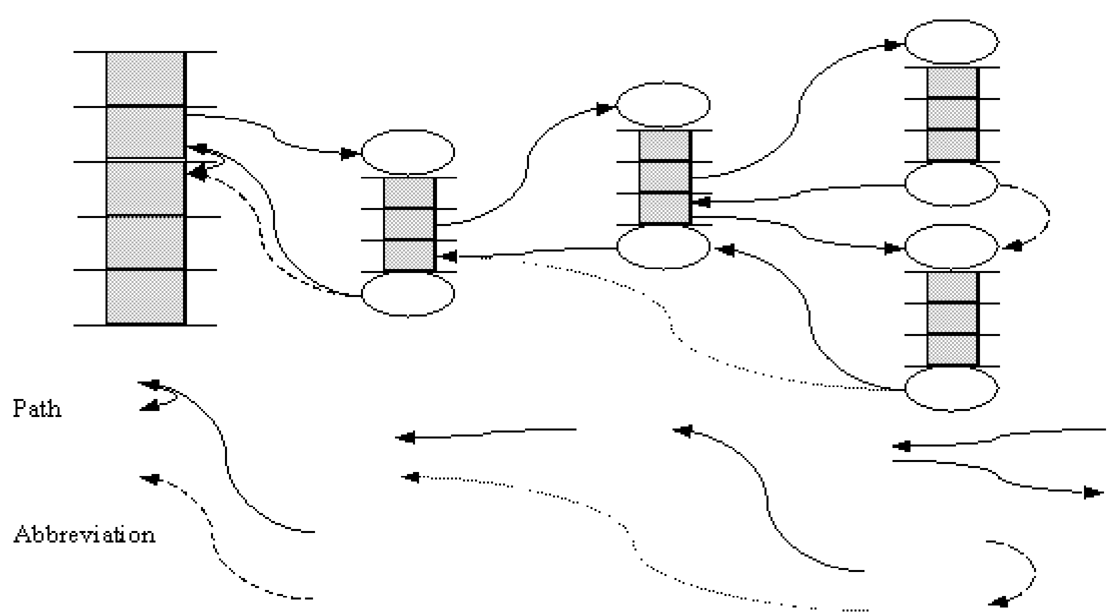

Figure 3: Successive Refinements of Frames with Abbreviations

Fifth, a succession of Φ connections[2] is often abbreviated by a single [[connection]]. Figure 3 illustrates this kind of abbreviation, where connections from the underlying Φ relations are shown as solid arrows and at the abbreviations are dashed lines. The bottom of the figure breaks out each abbreviation and the underlying Φ relations that it abbreviates.

The entirety of the abbreviations used in a framework effectively comprise the meta-structure of that framework. Zachman's original work was aptly called an architecture because it provided names ("Semantic Model", "Business Process Model", "Logistics Network", $\cdots$) for each cell. Providing an abbreviation "Entity" for objects in the Semantic Model cell, for example, clearly directs our analysis. Thus this abbreviation structure is "meta" in the sense that it characterizes how we define and work with a framework structure, not how the framework describes the structure and functionality of the real world. If a framework is limited to only paths for which there are abbreviations (or initial segments of those paths), then these abbreviations do indeed characterize the abstract structure of the framework.

3.2. Constraints

There are two ways in which constraints arise in developing a framework model: *a priori* and derived. An *a priori* constraint is discovered and explicitly declared during the modeling process. This is similar to the way foreign key constraints are handled in a relational database implementation. Derived constraints appear after the model has been developed as the result of some constraint discovery process.

The most important *a priori* constraint at the framework structure level identifies identities between outputs. Formally, it uses

$$\Xi_\alpha \qquad \subseteq \bigcup_{r \in \mathbf{R}} \mathcal{OC}_{\alpha,r} \times \mathcal{OC}_{\alpha,r}$$

It is required that Ξ_α be a true equivalence relation (it is reflexive, symmetric, and transitive). The actual constraint is then that $x\,\Phi_\alpha\,z$ and $y\,\Phi_\alpha\,z$ imply $x\,\Xi_\alpha\,y$. Figure 4 illustrates this situation. Using Ξ_α insures that an input connected to multiple sources receive only consistent information.

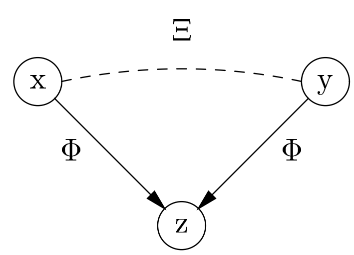

Figure 4: Ξ and Φ

It is also possible to treat Ξ as a derived constraint, as the transitive closure of the base equivalence $x\,\Xi_\alpha\,y$ iff $x\,\Phi_\alpha\,z$ & $y\,\Phi_\alpha\,z$, for some z.

We say a framework $\mathfrak{F}$ is *thorough* provided that, whenever a XXX frame $\mathfrak{F}_\alpha$ is defined, $\mathcal{SF}_\alpha(r, i, d)$ is defined for all $r \in \mathbf{R}$, $i \in \mathbf{I}$, and $d \in \mathcal{D}_\alpha$.

In order to fully capture the world being modeled, it is expected that there will be additional *ad hoc* constraints.

3.3. Views

A *view* is simply a subgraph of a framework $\mathfrak{F}$, possibly with non-branching paths collapsed to a single edge. Once again the analogy is with relational database systems,

2. Technically, this is a composition of various Φ components.

which have a well-defined view mechanism. As with relational views, complexly defined views may have update anomalies.

collapse Φ along paths ... problems with indirections

⋆11⋆ A view is *coherent* if any collapsing happens consistently across all paths. That is, when a path $\langle r_1, i_1, d_1\rangle, \ldots, \langle r_n, i_n, d_n\rangle$ collapses, then so do all paths $\langle s_1, j_1, d_1\rangle, \ldots, \langle s_n, j_n, d_n\rangle$. For example, in the context of a shipping system, only the weight of an item may be significant, so `Item.weight` is always collapsed to `Weight`.

Figure 5 illustrates the fact that a view may be used to conceptually rearrange a framework. On the right, a frame column is decomposed into a structure we notate as ((A.a,A.b,A.c) (B.a,B.b,B.c) (C.a,C.b,C.c)). The projections on the left give us individual
⋆12⋆ structures (A.a,B.a,C.a), (A.b,B.b,C.b), and (A.c,B.c,C.c).

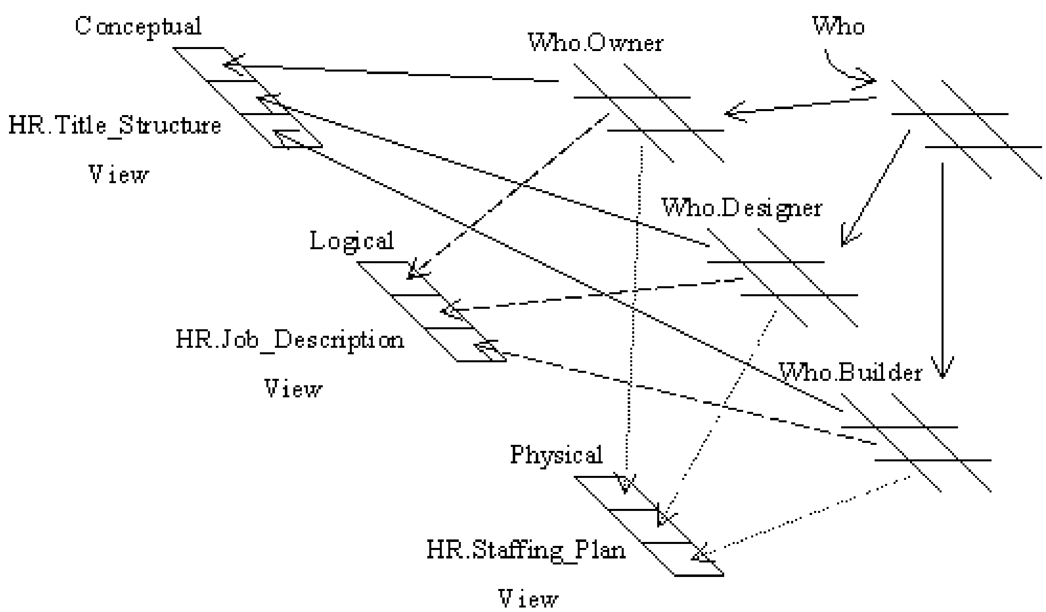

Figure 5: Rearranging with a View

4. Example

In this section, we show how the framework formalism can be used to express the Entity–Relationship model and carry this model forward to the logical design of the database tables. We are intentionally basing this example on aspects of the Enterprise Architecture that already have their own well-defined and widely-excepted formal specifications. There is difficulty formalizing columns such as motivation because the target lacks [[explicit structure]].

We already noted above that the first-level conceptual,data cell provides the enterprise's Semantic Model and that we may identify any frame labeled $\langle$conceptual, data, —$\rangle$ as an entity or relationship.[3] In fact, we have agreed to abbreviate any such triple as $\langle$Entity, —$\rangle$ or $\langle$Relationship, —$\rangle$. One such entity is named Employee; that is, the frame $\mathfrak{F}_{\langle \text{conceptual,data,Employee} \rangle}$ represents the Employee entity, containing (at least) the attributes of that entity. Formally, $\mathfrak{F}_{\langle \text{conceptual,data,Employee} \rangle}.\mathcal{IC} \subseteq \{\text{SSN, name, hiredate, skill}, \cdots\}$. Or, in abbreviation, $\mathcal{IC}_{\langle \text{Entity,Employee} \rangle}$
⋆13⋆ includes SSN, name, hiredate, skill, *etc.*

At its simplest manifestation, the frame for entity named Employee could be just an identity frame $\mathfrak{I}$, so all it does is define the decomposition of Employee into Employee.SSN, Employee.name, *etc.* Each attribute of an entity in the Semantic Model is then carried forward to be an attribute of a relation in the Logical Data Model. That is SSN as it appears in $\mathfrak{F}_{\langle \text{conceptual,data,Employee} \rangle}$ is mapped by $\mathfrak{F}_{\epsilon}.\Phi$ to SSN in $\mathfrak{F}_{\langle \text{logical,data,Employee} \rangle}$, and similarly with name, hiredate, *etc.* Precisely, the relation

$$\Phi\ (\ \langle \text{conceptual, data, Employee} \rangle \text{SSN},\ \ \langle \text{logical, data, Employee} \rangle \text{SSN}\ )$$

holds. Observe that the way Φ was defined removes any ambiguity from the occurrences of "SSN": the first occurrence comes from the $\mathcal{OC}$ of $\mathfrak{F}_{\langle \text{conceptual,data,employee} \rangle}$ and the second from the $\mathcal{IC}$ of $\mathfrak{F}_{\langle \text{logical,data,employee} \rangle}$.

As with "conceptual,data", we abbreviate "logical,data" as "Relation" at the first level and "attribute" at the second. Thus the frame labeled $\langle$Relation, Employee$\rangle$ has subframes $\langle$Relation, Employee$\rangle\langle$attribute, SSN$\rangle$, $\langle$Relation, Employee$\rangle\langle$attribute, name$\rangle$, $\langle$Relation, Employee$\rangle\langle$attribute, hiredate$\rangle$, $\cdots$.

Because the attribute named skill is in fact multi-valued, we must create a new relation Emp_Skill. Thus $\mathfrak{F}_{\langle \text{Relation,Emp\_Skill} \rangle}.\mathcal{IC} = \{\text{SSN, skill\_code}\}$.

5. Placing a Framework in its Context

This section discusses the interaction of an entire framework with its surroundings.
⋆14⋆ This is, of course, not only outside of our formalism but inherently unformalizable.

6. Merging Frameworks

It is fairly common that two frameworks must be merged — for example, when the two organizations modeled by the frameworks merge. Technically it is very easy to merge the frameworks by taking the union of their respective components; in fact, we use the union symbol to signify such a merger. That is, $\mathfrak{H}$ is the merger of frameworks $\mathfrak{F}$ and $\mathfrak{G}$, written $\mathfrak{F} \cup \mathfrak{G}$, whenever

$$\mathfrak{H}_{\alpha}.\diamondsuit \ = \ \mathfrak{F}_{\alpha}.\diamondsuit \cup \mathfrak{G}_{\alpha}.\diamondsuit$$

where $\diamondsuit$ is any of $\mathcal{IC}$, $\mathcal{OC}$, $\mathcal{SF}$, Φ, or $\mathfrak{S}$.

3. That we can't say for certain whether a conceptual,data subframe is an entity or relationship reflects a fact well-known to data modelers: it is sometimes fuzzy whether something is a relationship or an entity.

Of course any constraints must be merged as well. In the case of *a priori* Ξ, this means first unioning the two relationships and then reconstructing the transitive closure. This may detect violations of the constraint.

Bibliography

(1) "A Framework for Information Systems Architecture." John A. Zachman. IBM Systems Journal,vol. 26, no. 3, 1987. IBM Publication G321-5298.

(2) "Extending and Formalizing the Framework for Information Systems Architecture." J. F. Sowa and J. A. Zachman. IBM Systems Journal, vol. 31, no. 3, 1992. IBM Publication G321-5488.

(3) "Data Stores, Data Warehousing, and the Zachman Framework" by W. H. Inmon, John Zachman, and Jonathan Geiger. McGraw-Hill 1997.

(4) "Workstation Archecture in holonic manufacturing systems", by Wyns J., Van Brussel H., Valckenaers P., and Bongaerts L. 28th CIRP International Seminar on Manufacturing Systems. 1996 http://www.mech.kuleuvan.ac.be/pma/project/goa/extracts/architec.htm

(5) "Information Systems Architecture & Organization Syllabus" by Paul J.A. van Vliet. Information Systems & Quantitative Analysis Department, College of Information Science and Technology, University of Nebraska at Omaha. 1997 http://www.istis.unomaha/isqa/vanvliet/arch/syllabus.htm

(6) "A Framework of Mission Operations System Models" by Karlin J., and Welch D. http://www.esoc.esa.de/external/mso/SpaceOps/1_10/1_10.htm

(7) "Systems Analysis and Design Methods" by Whitten J. L., and Bentley L. D. published by Irwin McGraw-Hill

(8) "North Texas GIS Consortium Spatial Data Warehouse NSDI Demonstration Project Final Report" by Briggs R., Principal Investigator. The Bruton Center for Development Studies, The University of Texas at Dallas. 1997

(9) Williams, T.J., editor (May 1988), Reference Model for Computer Integrated Manufacturing, A Description from the Viewpoint of Industrial Automation, CIM Reference Model Committee, International Purdue Workshop on Industrial Computer Systems, Purdue Laboratory for Applied Industrial Control, Purdue University, West Lafayette, IN; Instrument Society of America (1989), Research Triangle Park, NC.

(10) personal communication with Jacobs, F. R. Kelley School Of Business, Indiana University. 1998


```
* * * * * * * * * Marginal Notes * * * * * * * * *
1. file: nota.tex, page: 1
 Why count every bolt more than once?

2. file: nota.tex, page: 1
 just a reminded to revisit summary when we're all done
```

```
3. file: nota.tex, page: 2
 disc of drawing frame doesn't belong in lead para, Fig shows column

4. file: nota.tex, page: 3
 replace XXX and YYY. How about "interior" and "frontier"?

5. file: nota.tex, page: 4
 This works better if fully qualified paths are used as arg's to Phi

6. file: nota.tex, page: 4
 should names be required rather than optional?  distinct?

7. file: nota.tex, page: 4
 Does this para belong here??  What are elements of domain D called?

8. file: nota.tex, page: 5
 except that Fig is wrong.  Should be "EIC" and "EOC".  Sorry Dick

9. file: nota.tex, page: 5
 qualify iota too??

10. file: nota.tex, page: 6
 explain SETs

11. file: nota.tex, page: 9
 say more in relation to abbrev section

12. file: nota.tex, page: 9
 view definition does not allow coallescing three views into one

13. file: nota.tex, page: 9
 add business rule constraints

14. file: nota.tex, page: 10
 global D
```