FOLE Equivalence

Robert E. Kent
Ontologos

Abstract. The first-order logical environment FOLE provides a rigorous and principled approach to distributed interoperable first-order information systems. FOLE has been developed in two forms: a classification form and an interpretation form. Two papers represent FOLE in a classification form corresponding to ideas of the Information Flow Framework ([14],[15],[16]): the first paper [9] provides a foundation that connects elements of the ERA data model [2] with components of the first-order logical environment FOLE; the second paper [10] provides a superstructure that extends FOLE to the formalisms of first-order logic. The formalisms in the classification form of FOLE provide an appropriate framework for developing the relational calculus. Two other papers represent FOLE in an interpretation form: the first paper [11] develops the notion of the FOLE table following the relational model [3]; the second paper [12] discusses the notion of a FOLE relational database. All the operations of the relational algebra have been rigorously developed [13] using the interpretation form of FOLE. The present study demonstrates that the classification form of FOLE is informationally equivalent to the interpretation form of FOLE. In general, the FOLE representation uses a conceptual structures approach, that is completely compatible with the theory of institutions, formal concept analysis and information flow.

Keywords: structures, specifications, sound logics, databases.
# Table of Contents

**FOLE Equivalence** ................................................................. 1  
*Robert E. Kent*  
1 Preface. ................................................................. 3  
2 Introduction. ................................................................. 5  
  2.1 First Order Logical Environment ................................. 5  
  2.2 Overview. ................................................................. 6  
3 Basic Concepts ............................................................... 9  
  3.1 Formalism ................................................................. 9  
  3.2 Interpretation ........................................................... 13  
  3.3 Satisfaction ............................................................. 15  
4 FOLE Structures .............................................................. 18  
  4.1 Structures. ............................................................... 18  
  4.2 Structure Morphisms. .................................................. 20  
  4.3 Transition. ............................................................... 21  
  4.4 Lax Structures. ......................................................... 22  
  4.5 Lax Structure Morphisms. ............................................. 23  
5 FOLE Specifications .......................................................... 26  
  5.1 Specifications. .......................................................... 26  
  5.2 Specification Morphisms. ............................................. 27  
  5.3 Specification Satisfaction .......................................... 28  
6 FOLE Sound Logics ............................................................. 30  
  6.1 Sound Logics. ........................................................... 30  
  6.2 Sound Logic Morphisms. .............................................. 31  
7 FOLE Databases ................................................................. 35  
  7.1 Databases. ............................................................... 35  
  7.2 Database Morphisms. .................................................. 37  
A Appendix ....................................................................... 43  
  A.1 FOLE Components ..................................................... 43  
  A.2 Classifications and Infomorphisms ............................... 45
1 Preface.

The architecture for the first-order logical environment FOLE is displayed in Fig. 1. The left side of Fig. 1 represents FOLE in classification form. The right side of Fig. 1 represents FOLE in interpretation form.

![Fig. 1. FOLE Architecture](image)

Classification form. An ontology defines the primitives with which to model the knowledge resources for a community of discourse (Gruber [6]). These primitives, which consist of classes, relationships and properties, are represented by the entity-relationship-attribute ERA model (Chen [2]). An ontology uses formal axioms to constrain the interpretation of these primitives. In short, an ontology specifies a logical theory.

Two papers provide a rigorous mathematical representation for the ERA data model in particular, and ontologies in general, within the first-order logical environment FOLE. These papers represent the formalism and semantics of (many-sorted) first-order logic in a classification form corresponding to ideas discussed in the Information Flow Framework (IFF [16]). The paper (Kent [9]) develops the notion of a FOLE structure; this provides a foundation that connects elements of the ERA data model with components of the first-order logical environment FOLE. The paper (Kent [10]) develops the notion of a FOLE sound logic; this provides a superstructure that extends FOLE to the formalisms of first-order logic.

FOLE is described in the following papers: “The ERA of FOLE: Foundation” [9], “The ERA of FOLE: Superstructure” [10], “The FOLE Table” [11], “The FOLE Database” [12], “FOLE Equivalence” [this paper], and “Relational Operations in FOLE” [13].
Interpretation form. The relational model (Codd [3]) is an approach for the information management of a “community of discourse” [3] using the semantics and formalism of (many-sorted) first-order predicate logic. The first-order logical environment FOLE (Kent [8]) is a category-theoretic representation for this logic. Hence, the relational model can naturally be represented in FOLE. A series of papers is being developed to provide a rigorous mathematical basis for FOLE by defining an architectural semantics for the relational data model.

Two papers provide a precise mathematical basis for FOLE interpretation. Both of these papers expand on material found in the paper (Kent [7]). The paper (Kent [11]) develops the notion of a FOLE table following the relational model (Codd [3]). The paper (Kent [12]) develops the notion of a FOLE relational database, thus providing the foundation for the semantics of first-order logical/relational database systems.

Equivalence. The current paper “FOLE Equivalence” defines an interpretation of FOLE in terms of the transformational passage, first described in (Kent [8]), from the classification form of first-order logic to an equivalent interpretation form, thereby defining the formalism and semantics of first-order logical/relational database systems. Although the classification form follows the entity-relationship-attribute data model of Chen [2], the interpretation form incorporates the relational data model of Codd [3]. Relational operations are described in the paper (Kent [13]). The relational calculus will be discussed in a future paper.

---

2 Examples include: an academic discipline; a commercial enterprise; the genetics research community, library science; the legal profession; etc.

3 Following the original discussion of FOLE (Kent [8]), we use the term mathematical context for the concept of a category, the term passage for the concept of a functor, and the term bridge for the concept of a natural transformation. A context represents some “species of mathematical structure”. A passage is a “natural construction on structures of one species, yielding structures of another species” (Goguen [5]).
2 Introduction.

This paper defines the equivalence between two forms of the first-order logical environment FOLE, the classification form and the interpretation form. Equivalence sits between the top of both forms, pictured briefly on the right, and more completely in Fig.1 in the preface. The classification form hierarchy (left hand side) consists of “The FOLE Foundation” at the bottom and “The FOLE Superstructure” at the top. The interpretation form hierarchy (right hand side) consists of “The FOLE Table” at the bottom and “The FOLE Database” at the top.

2.1 First Order Logical Environment

This paper relates two forms of the first order logic environment FOLE: the classification form and the interpretation form. These two forms are shown to be “informationally equivalent” to each other. Both forms have their own advantages: the classification form allows formalism to be easily defined, whereas the interpretation form allows relational operations to be easily defined. The classification form of FOLE is realized in the notion of a (lax) sound logic, whereas the interpretation form of FOLE is realized in the notion of a relational database. To demonstrate the equivalence of the classification form with the interpretation form, (lax) sound logics and relational databases are shown to be in a reflective relationship (Fig. 2). Although a FOLE relational database can be taken as a whole, a FOLE sound logic resolves into the two parts of structure and specification, plus the relationship of satisfaction.

In more detail (Tbl. 1), the classification form of FOLE is represented by a (lax) sound logic consisting of two components, a (lax) structure and an abstract specification, which are connected by a satisfaction relation. The (lax) structure consists of a (lax) entity classification, an attribute classification (typed domain), a schema, and one of the two equivalent descriptions: either a tabular map from predicates to tables; or a bridge consisting of an indexed collection of table tuple functions. The abstract specification is the same as the database schema. Satisfaction is defined by table interpretation using constraints. The interpretation form of FOLE is represented by a relational database with constant type domain, a tabular interpretation diagram, whose projective components consist of a database schema, a key diagram, and a tuple bridge. The tabular interpretation diagram is equivalent to the satisfaction relation between the (lax) structure and the abstract specification.

\textsuperscript{4} See Prop. 12 in § A.1 of the paper “The FOLE Table”.
\textsuperscript{5} See the paper “The ERA of FOLE: Superstructure”.
\textsuperscript{6} See the paper “Relational Operations in FOLE”.
\textsuperscript{7} The satisfaction relation corresponds to the “truth classification” in Barwise and Seligman, where the conceptual intent $M^S$ corresponds to the “theory of $M$”.
\textsuperscript{8} See (Key) Prop. 3 in § 5.3.
2.2 Overview.

\[ \text{Snd}\xrightarrow{lax}\text{Snd}\xrightarrow{im}\text{Db} \]

\[ \text{Snd}\xleftarrow{inc}\text{Db} \]

Fig. 2. FOLE Equivalence

Briefly, §3 defines some basic concepts of FOLE (formula, interpretation and satisfaction); §4 defines structures and converts them to a lax variety; §5 defines formal and abstract specifications, and satisfaction of specifications by structures; §6 defines (lax) sound logics and converts these to relational databases; and §7 defines relational databases and converts these to (lax) sound logics. In overview, Table 2 lists the figures and tables used in this paper.

Basic Concepts. §3 reviews some basic concepts (Tbl.3) of the FOLE logical environment from §2 of the paper [10] “Superstructure”. This covers the concepts of formalism, tabular interpretation, and satisfaction. §3.1 recapitulates the definition of formalism in [10]. Formalism is defined in terms of the logical Boolean operations and quantifiers using syntactic flow (Tbl.4). §3.2 extends to tables the definition of logical interpretation in [10] using Boolean operations within fibers and quantifier flow between fibers (Tbl.5). It represents the syntactic flow operators of Tbl.4 by their associated semantic flow operators (Tbl.5). §3.3 reviews the notion of satisfaction in [10], which links formalism and semantics via satisfaction. Satisfaction is defined for both sequents and constraints. Satisfaction is reflective (Fig.3) between relations and tables.

Structures. §4 reviews and extends the basic concepts of FOLE structure and structure morphism from §4.3 of the paper [9] “Foundation”. In §4.4 Def.10 defines a FOLE structure with both a classification and interpretation form. Fig.4
illustrates this idea. Note the global key sets used in the entity classification of this definition. In §4.2 Def.2 defines an extended notion of FOLE structure morphism by adding internal bridges. Fig.5 illustrates this idea. Note the global key function in the entity infomorphism of this definition. §4.3 defines a transition from strict to lax structures and structure morphisms. This is accomplished by changing the entity classification and entity infomorphism to lax versions, thus changing the global key sets and key function to local versions. Note1 Prop.1 (Key) and Cor.1 define a step-by-step process for this transition. Fig.6 illustrates the steps of this transition. §4.4 and §4.5 introduce the idea of lax structures and lax structure morphisms. We generalize to lax structures and lax structure morphisms by eliminating the global key sets and key functions. Def.3 defines a lax FOLE structure in terms of either a tabular interpretation function or a tuple bridge. Prop.2 shows that a lax structure is the same as the constraint-free aspect of a database. Def.4 defines a (lax) structure morphism. Cor.2 shows that a (lax) structure morphism defines a tabular interpretation bridge function. Fig.7 illustrates a (lax) structure morphism.

Specifications. §5 reviews and extends the notion of a FOLE specification. Formal FOLE specifications are covered in §3.1 of the paper [10] “Superstructure”. Here we extend to abstract FOLE specifications. §5.1 defines a formal specification in Def.5 and an abstract specification in Def.6. Every abstract specification is closely linked to a companion formal specification. §5.2 defines a abstract specification morphism in Def.7 and illustrates this in Fig.8. Abstract specifications and morphisms are the same as database schemas and morphisms. §5.3 defines specification satisfaction. Def.8 defines satisfaction for formal specifications, whereas Def.9 defines satisfaction for abstract specifications in terms of satisfaction for their formal companion. Assuming satisfaction holds, Def.10 defines the abstract table passage as the composition of the object-identical companion passage, the inclusion into the structure conceptual intent, and the relation interpretation passage of Lem.2 in §3.3. Prop.3 is key: it proves that satisfaction is equivalent to tabular interpretation. This is central, since it binds together the two parts of a sound logic, its structure and its specification. It underpins the core idea that sound logics are equivalent to relational databases. Fig.9 illustrates and compares specification passages, both in general and with satisfaction. Tbl.7 illustrates and compares specifications with the sound logics defined in §6.

Sound Logics. §6 reviews the notion of a FOLE sound logic. §6.1 discusses sound logics. Def.11 defines (lax) sound logics in terms of satisfaction between its two components: structure and abstract specification. Prop.4 proves that any (lax) sound logic defines a relational database. Tbl.8 lists and illustrates the various components of a (lax) sound logic. §6.2 discusses sound logic morphisms. Def.12 defines the notion of a (lax) sound logic morphism. Prop.5 shows how (lax) sound logic morphisms preserve and link satisfaction between their source and target logics. The large figure Fig.10 expands in detail the naturality used in this proposition. Prop.6 proves that a (lax) sound logic morphism defines a database
morphism. Thm. 1 proves existence of a passage from the context of (lax) \( \text{FOLE} \) sound logics to the context of \( \text{FOLE} \) relational databases. Tbl. 9 illustrates the components of a (lax) sound logic morphism.

\textit{Relational Databases.} §7 reviews the notion of a \( \text{FOLE} \) relational database. §7.1 discusses \( \text{FOLE} \) relational databases. Def. 13 defines a relational database as an interpretation diagram from predicates to tables for a fixed type domain. Fig. 11 illustrates this definition. Def. 14 defines relational databases using table projection passages. Tbl. 10 lists the components of a \( \text{FOLE} \) relational database. Prop. 7 shows that the constraint-free aspect of a database is the same as a (lax) structure. Prop. 10 proves that a \( \text{FOLE} \) database defines a (lax) \( \text{FOLE} \) sound logic. §7.2 discusses \( \text{FOLE} \) relational database morphisms. Def. 15 defines a database morphism to consist of a tabular passage between source/target predicate contexts, an infomorphism between source/target type domains, and a bridge connecting the source/target tabular interpretations. Fig. 12 illustrates this definition. Def. 16 defines relational database morphisms using table projection passages. Tbl. 11 lists and Fig. 13 illustrates the components of a \( \text{FOLE} \) relational database. Prop. 11 shows that the constraint-free aspect of a \( \text{FOLE} \) database morphism is the same as a (lax) \( \text{FOLE} \) structure morphism. Prop. 12 proves that a \( \text{FOLE} \) database morphism defines a (lax) \( \text{FOLE} \) sound logic morphism. Thm. 2 proves existence of a passage from the context of \( \text{FOLE} \) relational databases to the context of (lax) \( \text{FOLE} \) sound logics. Thm. 3 proves that the contexts of \( \text{FOLE} \) relational databases and \( \text{FOLE} \) (lax) sound logics are “informationally equivalent” by way of a reflection.

\textit{Appendix.} §A.1 lists the \( \text{FOLE} \) components used in this paper. Tbl. 12 in §A.1 lists the equivalent and adjoint versions of \( \text{FOLE} \) bridges. Tbl. 13 in §A.1 lists the \( \text{FOLE} \) Morphisms. §A.2 reviews the concepts of classifications and infomorphisms. Both are extended to lax versions.
3 Basic Concepts

The basic concepts of FOLE are listed in Tbl.3. Except for the formula interpretation in Tbl.5, we can refer all the formal material to the paper “The ERA of FOLE: Superstructure” [10].

| §2.1  | Formalism (formulas, sequents, constraints)  |
|-------|---------------------------------------------|
|       | Axioms (Tbl. 3)                            |
| §2.2  | Semantics (formula interpretation)         |
|       | Formal/Semantics Reflection (Tbl. 6)       |
| §2.3  | Satisfaction (sequents, constraints)       |

Table 3. Basic Concepts

3.1 Formalism

Formulas. Let $S = (R, \sigma, X)$ be a fixed schema with a set of entity types $R$, a set of sorts (attribute types) $X$ and a signature function $R \rightarrow \text{List}(X)$. The set of entity types $R$ is partitioned $R = \bigcup_{(I,s) \in \text{List}(X)} R(I,s)$ into fibers, where

9 Satisfaction for (abstract) specifications is defined in Def.9 of §5.3.
$R(I, s) \subseteq R$ is the fiber (subset) of all entity types with signature $(I, s)$. These are called $(I, s)$-ary entity types. Here, we follow the tuple, domain, and relation calculi from database theory, using logical operations to extend the set of basic entity types $R$ to a set of defined entity types $\hat{R}$ called formulas or queries.

Formulas, which are defined entity types corresponding to queries, are constructed by using logical connectives within a fiber and logical flow along signature morphisms between fibers (Tbl. 3). Logical connectives on formulas express intuitive notions of natural language operations on the interpretation (extent, view) of formulas. These connectives include: conjunction, disjunction, negation, implication, etc. For any signature $(I, s)$, let $\hat{R}(I, s) \subseteq \hat{R}$ denote the set of all formulas with this signature. There are called $(I, s)$-ary formulas.

The set of $S$-formulas is partitioned as $\hat{R} = \bigcup_{(I, s) \in \text{List}(X)} \hat{R}(I, s)$.

**fiber:** Let $(I, s)$ be any signature. Any $(I, s)$-ary entity type (relation symbol) is an $(I, s)$-ary formula; that is, $R(I, s) \subseteq \hat{R}(I, s)$. For a pair of $(I, s)$-ary formulas $\varphi$ and $\psi$, there are the following $(I, s)$-ary formulas: meet ($\varphi \land \psi$), join ($\varphi \lor \psi$), implication ($\varphi \rightarrow \psi$) and difference ($\varphi \setminus \psi$). For $(I, s)$-ary formula $\varphi$, there is an $(I, s)$-ary negation formula $\neg \varphi$. There are top/bottom $(I, s)$-ary formulas $\top(I, s)$ and $\bot(I, s)$.

**flow:** Let $(I', s') \xrightarrow{h} (I, s)$ be any signature morphism. For $(I, s)$-ary formula $\varphi$, there are $(I', s')$-ary existentially/universally quantified formulas $\Sigma h(\varphi)$ and $\Pi h(\varphi)$. For an $(I', s')$-ary formula $\varphi'$, there is an $(I, s)$-ary substitution formula $h^*(\varphi') = \varphi'(t)$.

---

*a* For any index $i \in I$, quantification for the complement inclusion signature function $\langle I \setminus \{i\}, s' \rangle \xrightarrow{\text{inc}} (I, s)$ gives the traditional syntactic quantifiers $\forall_i \varphi, \exists_i \varphi$.

\[
\begin{align*}
\langle I', s' \rangle & \xrightarrow{h} \langle I, s \rangle \\
\hat{R}(I', s') & \xrightarrow{\Sigma h} \hat{R}(I, s) \\
\hat{R}(I', s') & \xrightarrow{\Pi h} \hat{R}(I, s)
\end{align*}
\]

**Table 4. Syntactic Flow**

In general, we regard formulas to be constructed entities or queries (defining views and interpretations; i.e., relations/tables), not assertions. Contrast this with the use of “asserted formulas” below. For example, in a corporation data model the conjunction ($\text{Salaried} \land \text{Married}$) is not an assertion, but a constructed entity type or query that defines the view “salaried employees that

---

10 This is a slight misnomer, since the signature of $r$ is $\sigma(R) = (I, s)$, whereas the arity of $r$ is $\alpha(R) = I$.

11 An $S$-signature morphism $(I', s') \xrightarrow{h} (I, s)$ in $\text{List}(X)$ is an arity function $I' \xrightarrow{h} I$ that preserves signature $s' = h \cdot s$.

12 The full version of FOLE (Kent [8]) defines syntactic flow along term vectors.
are married”. Formulas form a schema \( \hat{S} = \langle R, \hat{\sigma}, X \rangle \) that extends \( S \) with \( S \)-formulas as entity types: with the inductive definitions above, the set of entity types is extended to a set of logical formulas \( R \xrightarrow{inc_S} \hat{R} \), and the entity type signature function is extended to a formula signature function \( \hat{R} \xrightarrow{\hat{\sigma}} \text{List}(X) \) with \( \sigma = inc_S \cdot \hat{\sigma} \).

**Sequents.** To make an assertion about things, we use a sequent. Let \( \mathcal{S} = (R, \sigma, X) \) be a schema. A (binary) \( \mathcal{S} \)-sequent \(^{13}\) is a pair of formulas \( (\varphi, \psi) \in \hat{R} \times \hat{R} \) with the same signature \( \hat{\sigma}(\varphi) = (I, s) = \hat{\sigma}(\psi) \). To be explicit that we are making an assertion, we use the turnstile notation \( \varphi \vdash \psi \) for a sequent. Then, we are claiming that a specialization-generalization relationship exists between the formulas \( \varphi \) and \( \psi \). An asserted sequent \( \varphi \vdash \psi \) expresses interpretation widening, with the interpretation (view) of \( \varphi \) required to be within the interpretation (view) of \( \psi \). An asserted formula \( \varphi \in \hat{R} \) can be identified with the sequent \( \top \vdash \varphi \) in \( \hat{R} \times \hat{R} \), which asserts the universal view “all entities” of signature \( \sigma(\varphi) = (I, s) \). Hence, from an entailment viewpoint we can say that “formulas are sequents”. In the opposite direction, there is an enfolding map \( \hat{R} \times \hat{R} \rightarrow \hat{R} \) that maps \( \mathcal{S} \)-sequents to \( \mathcal{S} \)-formulas \( (\varphi \vdash \psi) \rightarrow (\varphi \rightarrow \psi) \). The axioms in Tbl. 3 of the paper \(^{10}\) make sequents into an order. Let \( \text{Con}_\mathcal{S}(I, s) = \langle \mathcal{R}(I, s), \vdash \rangle \) denote the fiber preorder of \( \mathcal{S} \)-formulas.

**Constraints.** Sequents only connect formulas within a particular fiber; an \( \mathcal{S} \)-sequent \( \varphi \vdash \psi \) is between two formulas with the same signature \( \hat{\sigma}(\varphi) = (I, s) = \hat{\sigma}(\psi) \), and hence between elements in the same fiber \( \varphi, \psi \in \mathcal{R}(I, s) \). We now define a useful notion that connects formulas across fibers. An \( \mathcal{S} \)-constraint \( \varphi' \xrightarrow{h} \varphi \) consists of a signature morphism \( \hat{\sigma}(\varphi') = (I', s') \xrightarrow{h} (I, s) = \hat{\sigma}(\varphi) \) and a binary sequent \( \sum_h(\varphi) \vdash \varphi' \) in \( \text{Con}_\mathcal{S}(I', s') \), or equivalently by the axioms in Tbl. 3 of the paper \(^{10}\), a binary sequent \( \varphi \vdash h^*(\varphi') \) in \( \text{Con}_\mathcal{S}(I, s) \). Hence, a constraint requires that the interpretation of the \( h^\text{th} \)-projection of \( \varphi \) be within the interpretation of \( \varphi' \), or equivalently, that the interpretation of \( \varphi' \) be within the interpretation of the \( h^\text{th} \)-substitution of \( \varphi' \). \(^{14}\)

Given any schema \( S \), an \( S \)-constraint \( \varphi' \xrightarrow{h} \varphi \) has source formula \( \varphi' \) and target formula \( \varphi \). Constraints are closed under composition: \( \varphi'' \xrightarrow{h'} \varphi' \xrightarrow{h} \varphi \). Let \( \text{Con}(S) \) \(^{15}\) denote the mathematical context, whose set of objects are \( S \)-formulas and whose set of morphisms are \( S \)-constraints. This context is fibered over the projection passage \( \text{Cons}(S) \rightarrow \text{List}(X) \):

\(^{13}\) Since FOLE formulas are not just types, but are constructed using, inter alia, conjunction and disjunction operations, we can restrict attention to binary sequents.

\(^{14}\) In some sense, this formula/constraint approach to formalism turns the tuple calculus upside down, with atoms in the tuple calculus becoming constraints here.

\(^{15}\) \( \text{Con}(S) \), which stands for “\( S \)-constraints”, was defined in the paper \(^{10}\).
\[(\varphi' \xrightarrow[h]{\ } \varphi) \mapsto ((I', s') \xrightarrow[h]{\ } (I, s)).\] Formula formation is idempotent: \(\widehat{R} = \widehat{R}\) and 
\[\text{Cons}(\widehat{S}) = \text{Cons}(S).\]

Sequents are special cases of constraints: a sequent \(\varphi' \vdash \varphi\) asserts a constraint \(\varphi \xrightarrow[s]{\ } \varphi'\) that uses an identity signature morphism. Since an asserted formula \(\varphi\) can be identified with the sequent \(\top \vdash \varphi\), it can also be identified with the constraint \(\varphi \xrightarrow[s]{\ } \top\). Thus, from an entailment viewpoint we can say that “formulas are sequents are constraints”. In the opposite direction, there are enfolding maps that map \(S\)-constraints to \(S\)-formulas: either \((\varphi' \xrightarrow[h]{\ } \varphi) \mapsto (\sum_h(\varphi) \rightarrow \varphi')\) with signature \(\langle I', s' \rangle\), or \((\varphi' \xrightarrow[h]{\ } \varphi) \mapsto (\varphi \rightarrow h^*(\varphi'))\) with signature \(\langle I, s \rangle\).

\[\text{Formulas form a schema } \widehat{S} = (\widehat{R}, \widehat{\sigma}, X) \text{ that extends } S \text{ with } S\text{-formulas as entity types (relation names): by induction, the set of entity types is extended to a set of logical formulas } R \xleftarrow[incS]{\ } \widehat{R}, \text{ and the entity type signature function is extended to a formula signature function } \widehat{R} \xrightarrow[\sigma]{\ } \mathbf{List}(X) \text{ with } \sigma = incS \cdot \widehat{\sigma}.\]

\[\text{A constraint in the fiber } \text{Con}_S(I, s) \text{ uses an identity signature morphism } \varphi \xrightarrow[s]{\ } \varphi', \text{ and hence is a sequent } \varphi' \vdash \varphi.\]
3.2 Interpretation

The logical semantics of a structure $\mathcal{M}$ resides in its core, which is defined by its formula interpretation function

$$T_{\mathcal{M}} : \hat{R} \to \text{Tbl}(A).$$  \hspace{1cm} (1)

The formula interpretation function, which extends the traditional interpretation function $T_{\mathcal{M}} : R \to \text{Tbl}(A)$ of a structure $\mathcal{M}$ (Def. 1 in §4), is defined in Tbl. 5 by induction within fibers and flow between fibers. To respect the logical semantics, the formal flow operators $\langle \sum_h \Pi_h, h^* \rangle$ for existential/universal quantification and substitution reflect the semantic flow operators $\langle \sum_h \Pi_h, h^* \rangle$ via interpretation (Tbl. 6).

**fiber:** For each signature $\langle I, s \rangle \in \text{List}(X)$, the fiber function $\hat{R}(I, s)$  \hspace{1cm} (bottom Tbl. 6) is defined in the top part of Tbl. 6 by induction on formulas. At the base step (first line in the top of Tbl. 6), it defines the formula interpretation of an entity type $r \in \hat{R}$ as the traditional interpretation of that type: the $A$-table $T_{\mathcal{M}}(r) = \langle \sigma(r), K(r), \tau_r \rangle \in \text{Tbl}(A)$ (Fig. 4 of §4). At the induction step (remaining lines in the top of Tbl. 6), it represents the logical operations by their associated boolean operations: meet of interpretations for conjunction, join of interpretations for disjunction, etc.; see the boolean operations defined in §3.2 of the paper “Relational Operations in FOLE” [13].

**flow:** (bottom Tbl. 6), It represents the syntactic flow operators in Tbl. 4 by their associated semantic flow operators; see the adjoint flow for fixed type domain $A$ defined in §3.3.1 of the paper “Relational Operations in FOLE” [13]. The function $\hat{R} \xrightarrow{T_{\mathcal{M}}} \text{Tbl}(A)$ is the parallel combination of its fiber functions

$$\left\{ \hat{R}(I, s) \xrightarrow{T_{\mathcal{M}}(I,s)} \text{Tbl}_A(I, s) \mid \langle I, s \rangle \in \text{List}(X) \right\}$$ defined above. The function $T_{\mathcal{M}}$ is defined in terms of these fiber functions and the flow operators.

---

18 The morphic aspect of Tbl. 6 anticipates the definition of satisfaction in §3.3 for sequents within the fibers $T_A(I, s)$ and for constraints using flow between fibers; $\langle \sum_h \Pi_h^* \rangle : T_A(I', s') \Rightarrow T_A(I, s)$.

19 The paper [13] used only two of the three operators in the fiber adjunction of tables $T_A(I', s') \xleftarrow{\{ h_A^* \}} T_A(I, s)$ defined by composition/pullback: the left adjoint existential operation (called projection) and the right adjoint substitution or inverse image operation (called inflation).
fiber: signature \( \langle I, s \rangle \) with extent (tuple set) \( \text{tup}_A(I, s) = \prod_{i \in I} A_i \)

| operator | definition |
|----------|------------|
| \( \text{tup}_M(\varphi) \in \text{Tbl}_A(I, s) = (\text{Set} \downarrow \text{tup}_A(I, s)) \) |
| \( T_M(r) = T_M(\langle \sigma(r), K(r), \tau_r \rangle \, r \in R(I, s) \subseteq \hat{R}(I, s) \) |
| meet | \( T_M(\varphi \land \psi) = T_M(\varphi) \land T_M(\psi) \) |
| \( \varphi, \psi \in \hat{R}(I, s) \) |
| join | \( T_M(\varphi \lor \psi) = T_M(\varphi) \lor T_M(\psi) \) |
| top | \( T_M(\top_{(I, s)}) = \langle \text{tup}_A(I, s), 1 \rangle \) |
| terminal table |
| bottom | \( T_M(\bot_{(I, s)}) = (\emptyset, 0) \) |
| initial table |
| negation | \( T_M(\lnot \varphi) = T_M(\top_{(I, s)}) \setminus \varphi = T_M(\top_{(I, s)}) - T_M(\varphi) \) |
| implication | \( T_M(\varphi \rightarrow \psi) = T_M(\varphi) \rightarrow T_M(\psi) = (\lnot T_M(\varphi)) \lor T_M(\psi) \) |
| difference | \( T_M(\varphi \setminus \psi) = T_M(\varphi) - T_M(\psi) = T_M(\varphi) \land (\lnot T_M(\psi)) \) |

flow: signature morphism \( \langle I', s' \rangle \xrightarrow{h} \langle I, s \rangle \)

with tuple map \( \text{tup}_A(I', s') \xrightarrow{\text{tup}_A(h)} \text{tup}_A(I, s) \)

| operator | definition |
|----------|------------|
| \( T_M(\Sigma_h(\varphi)) = \Sigma_h(T_M(\varphi)) \) |
| \( \varphi \in \hat{R}(I, s) \) |
| \( T_M(\Pi_h(\varphi)) = \Pi_h(T_M(\varphi)) \) |
| \( h^{\ast}(\varphi') = h^{\ast}(I_M(\varphi')) \) |
| \( \varphi' \in \hat{R}(I', s') \) |

Table 5. Formula Interpretation

\[
\begin{array}{ccc}
\langle \hat{R}(I', s'), h^{\ast} \rangle & \xrightarrow{\Sigma_h} & \langle \hat{R}(I, s), h \rangle \\
\Pi_h & \downarrow & \Pi_h \\
T_M(I', s') & \xrightarrow{T_M} & T_M(I, s) \\
\Sigma_h & \downarrow & \Sigma_h \\
\text{Tbl}_A(I', s') & \xrightarrow{h^{\ast}} & \text{Tbl}_A(I, s) \\
\Pi_h & \downarrow & \Pi_h \\
\end{array}
\]

\[
\begin{array}{ccc}
(\langle I', s' \rangle \xrightarrow{h} \langle I, s \rangle) & \xrightarrow{\text{tup}_A} & \text{tup}_A(I', s') \xleftarrow{\text{tup}_A(h)} \text{tup}_A(I, s) \\
\Sigma_h \downarrow h^{\ast} \downarrow \Pi_h \\
\Sigma_h \circ T_M(I', s') = T_M(I', s') \circ \Sigma_h \\
h^{\ast} \circ T_M(I, s) = T_M(I, s') \circ h^{\ast} \\
\Pi_h \circ T_M(I', s') = T_M(I, s') \circ \Pi_h \\
\end{array}
\]

Table 6. Formal/Semantics Reflection
3.3 Satisfaction.

Satisfaction is a fundamental classification between formalism and semantics. The atom of formalism used in satisfaction is the FOLE constraint, whereas the atom of semantics used in satisfaction is the FOLE structure. Satisfaction is defined in terms of the table formula interpretation function \( T_M : \hat{R} \rightarrow Tbl(A) \) of §3.2 and the associated the relation formula interpretation function \( R_M : \hat{R} \rightarrow Rel(A) \). Assume that we are given a structure schema \( S = \langle R, \sigma, X \rangle \) with a set of predicate symbols \( R \) and a signature (header) map \( \sigma : \hat{R} \rightarrow \text{List}(X) \).

Sequent Satisfaction. A (lax) \( S \)-structure \( M = \langle E, \sigma, \tau, A \rangle \in \text{Struc}(S) \) satisfies a formal \( S \)-sequent \( \varphi \vdash \varphi' \), for a pair of formulas \( (\varphi, \varphi') \in \hat{R} \times \hat{R} \) with the same signature \( \hat{\sigma}(\varphi) = \langle I, s \rangle = \hat{\sigma}(\varphi') \), when the interpretation widening of views asserted by the sequent actually holds in \( M \):

\[
R_M(\varphi) \subseteq R_M(\varphi') \quad \text{reflectively}^{21} \quad T_M(\varphi) \xrightarrow{k} T_M(\varphi').
\]

Satisfaction is symbolized either by \( M \models_S (\varphi \vdash \varphi') \) or by \( \varphi \vdash_M \varphi' \). For each signature \( \langle I, s \rangle \in \text{List}(X) \), satisfaction defines the fiber order \( \langle \hat{R}(I, s), \leq_M \rangle \), where \( \varphi \leq_M \varphi' \) when \( \varphi \vdash_M \varphi' \) for any two formulas \( \varphi, \varphi' \in \hat{R}(I, s) \).

\[\text{Fig. 3. Table-Relation Reflection}\]

---

20 Denoted by \( I_M : \hat{R} \rightarrow \text{Rel}(A) \) in § 2.2.1 of the paper “The ERA of FOLE: Superstructure” [10].

21 See the formal/semantics reflection defined in § A.1 of the paper “The FOLE Table” [11] and in § 3.1 of the paper “Relational Operations in FOLE” [13].
Constraint Satisfaction. A (lax) $S$-structure $M \in \text{Struc}(S)$ satisfies a formal $S$-constraint $\varphi' \xrightarrow{h} \varphi$, for a pair of formulas $(\varphi', \varphi) \in R \times \tilde{R}$ connected by a signature morphism $\tilde{\sigma}(\varphi') = (I', s') \xrightarrow{h} (I, s) = \tilde{\sigma}(\varphi)$, when the following equivalence holds

$$R_M(\varphi') \supseteq \exists_h(R_M(\varphi)) \iff h^{-1}(R_M(\varphi')) \supseteq R_M(\varphi);$$

reflectively (visualized in Fig.3)

$$R_M(\varphi') \leftarrow R_M(h) \rightarrow R_M(\varphi)$$

reflectively (visualized in Fig.3)

$$T_M(\varphi') \leftarrow k \sum_h(T_M(\varphi)) \iff h^*(T_M(\varphi')) \leftarrow k' \sum_h(T_M(\varphi)).$$

Satisfaction is symbolized by $M \models_S (\varphi \xrightarrow{h} \varphi)$.

Lemma 1. A (lax) $S$-structure $M$ determines a mathematical context $M^S \subseteq \text{Cons}(S)$, called the conceptual intent of $M$, whose objects are $S$-formulas and whose morphisms are $S$-constraints satisfied by $M$. The $(I, s)^{th}$ fiber is the order $M^S(I, s) = \langle \tilde{R}(I, s), \geq_M \rangle$.

Proof. $M^S$ is closed under constraint identities and constraint composition.

1: $M \models_S (\varphi \xrightarrow{h} \varphi)$, and 2: if $M \models_S (\varphi'' \xrightarrow{h'} \varphi')$ and $M \models_S (\varphi \xrightarrow{h} \varphi)$, then $M \models_S (\varphi'' \xrightarrow{h \cdot h'} \varphi')$, since $\varphi'' \supseteq_M \sum_h(\varphi')$ and $\varphi' \supseteq_M \sum_h(\varphi)$ implies $\varphi'' \supseteq_M \sum_h(\varphi') = \sum_h(\varphi)$.

Lemma 2. For any structure $M$, there is a relation interpretation passage and a table interpretation graph morphism, reflectively

$$M^S_{op} \xrightarrow{R_M} \text{Rel}(A) \xleftarrow{\text{inc}_A} \text{Tbl}(A)$$

and

$$|M^S|_{op} \xrightarrow{T_M} |	ext{Tbl}(A)||,$$

which extend the formula interpretation function $T_M : \tilde{R} \rightarrow \text{Tbl}(A)$ of §1.2 and hence the structure interpretation function $T_M : \tilde{R} \rightarrow \text{Tbl}(A)$ of §1.2.

Proof. An $S$-formula $\varphi \in M^S(I, s) = \langle \tilde{R}(I, s), \leq_M \rangle$ with $S$-signature $\langle I, s \rangle$ is mapped to an $A$-table $T_M(\varphi) = \langle I, s, K, t \rangle$ defined by induction in the top part of Tbl. 4 in in §3.2. Reflectively, the formula $\varphi$ is mapped to an $A$-relation $R_M(\varphi) = \langle I, s, \varphi(t(K)) \rangle$.

---

22 The satisfaction relation corresponds to the “truth classification” in Barwise and Seligman 11, where the conceptual intent $M^S$ corresponds to the “theory of $M$”.

23 Relational interpretation is closed under composition; tabular interpretation is closed under composition only up to key equivalence.
morphisms: An $S$-constraint $\varphi' \xrightarrow{h} \varphi$ satisfied by the $S$-structure $\mathcal{M} \in \textbf{Struc}(S)$, $\mathcal{M} \models_S (\varphi' \xrightarrow{h} \varphi)$, is mapped by constraint satisfaction to the $\textbf{Tbl}(A)$-morphism $T_\mathcal{M}(\varphi') = T' = \langle I', s', K(\varphi'), t_{\varphi'} \rangle \xleftarrow{(h,k)} \langle I, s, K(\varphi), t_{\varphi} \rangle = T = T_\mathcal{M}(\varphi)$ with indexing $X$-sorted signature morphism $\langle I', s' \rangle \xrightarrow{h} \langle I, s \rangle$ and a key function $K(\varphi') \xrightarrow{k} K(\varphi)$ [24] satisfying either of the adjoint fiber morphisms in Disp[4] leading to the naturality condition $k \cdot t_{\varphi'} = t_{\varphi} \cdot \text{tup}_A(h)$ and visualized in Fig.3. Reflectively, the constraint $\varphi' \xrightarrow{h} \varphi$ is mapped to an $A$-relation morphism $R_\mathcal{M}(\varphi') = R' = \langle I', s', \varphi t_{\varphi'}(K(\varphi')) \rangle \xleftarrow{(h,r)} \langle I, s, \varphi t_{\varphi}(K(\varphi)) \rangle = R = R_\mathcal{M}(\varphi)$ satisfying either of the adjoint fiber orderings in Disp[3] leading to the naturality condition $r \cdot i_{\varphi'} = i_{\varphi} \cdot \text{tup}_A(h)$ and visualized in Fig.3 □

[24] Defined by choice. However, the choice for a composite signature morphism is not identical to the composition of the component choices; only equivalent.
4 **FOLE Structures.**

We define two forms of structures and structure morphisms: a classification (strict) form and an interpretation (lax) form.

### 4.1 Structures.

Assume that we are given a schema \( S = \langle R, \sigma, X \rangle \) with a set of predicate symbols \( R \) and a signature (header) map \( R \xrightarrow{\sigma} \text{List}(X) \).

**Definition 1.** An \( S \)-structure \( M = \langle E, \sigma, \tau, A \rangle \) (LHS Fig. 4) consists of an entity classification \( E = \langle R, K, \vert = E \rangle \) of predicate symbols \( R \) and keys \( K \), an attribute classification (typed domain) \( A = \langle X, Y, \vert = A \rangle \) of sorts \( X \) and data values \( Y \), and a list designation \( \langle \sigma, \tau \rangle: E \models \text{List}(A) \) with signature map \( R \xrightarrow{\sigma} \text{List}(X) \), tuple map \( K \xrightarrow{\tau} \text{List}(Y) \) and defining condition \( k \models \varepsilon \) implies \( \tau(k) \models \text{List}(A) \sigma(r) \).

Projections define the type hypergraph (schema) \( S = \langle R, \sigma, X \rangle \), and an instance hypergraph (universe) \( U = \langle K, \tau, Y \rangle \).

![Figure 4. FOLE Structure](image)

By dropping the global key set \( K \), the entity classification \( E = \langle R, K, \models \varepsilon \rangle \) can be regarded (RHS Fig. 4) as a set-valued function \( K = R \xrightarrow{\text{ext}} \varphi K \subseteq \text{Set} \); that is, as an indexed collection \( E = \langle R, K \rangle \) of subsets of keys \( \{ \text{ext}_E(r) = K(r) \subseteq \text{Set} \mid r \in R \} \). The defining condition of the list designation demonstrates that a structure can be regarded (MID Fig. 4) as a table-valued function \( R \xrightarrow{T_M} \text{Tbl}(A) \); that is, as an \( R \)-indexed collection of \( A \)-tables \( T_M(r) = \langle \sigma(r), K(r), \tau_r \rangle \), whose tuple maps \( \{ K(r) \xrightarrow{\text{tup}_A} \text{tup}_A(\sigma(r)) \mid r \in R \} \) are restrictions of the tuple map \( K \xrightarrow{\text{ext}} \text{List}(Y) \). Here, an entity type (predicate symbol) \( r \in R \) is interpreted as the \( A \)-table \( T_M(r) = \langle \sigma(r), K(r), \tau_r \rangle \in \text{Tbl}(A) \), consisting of the \( X \)-signature \( \sigma(r) \in \text{List}(X) \), the key set \( K(r) = \text{ext}_E(r) \subseteq K \), and the data value \( t_n \) is of sort \( s_n \).

25 In more detail, if entity \( k \in K \) is of type \( r \in R \), then the description tuple \( \tau(k) = \langle J, t \rangle \) is the same “size” \( (J = I) \) as the signature \( \sigma(r) = \langle I, s \rangle \) and for each index \( n \in I \) the data value \( t_n \) is of sort \( s_n \).
tuple function $K(r) \xrightarrow{\tau} \text{tup}_A(\sigma(r))$, which is a restriction of the tuple function $K \xrightarrow{\tau} \text{List}(Y)$. The list classification $\text{List}(A) = \langle \text{List}(X), \text{List}(Y), \models_{\text{List}(A)} \rangle$ can alternatively be regarded (RHS Fig.4) (Fig.4 of the paper [9]) as a set-valued function $\text{List}(X) \xrightarrow{\text{ext}_{\text{List}(A)}} \wp \text{List}(Y)$; that is, as an indexed collection $\text{List}(A) = \langle \text{List}(X), \text{tup}_A, \text{List}(Y) \rangle$ of subsets of tuples $\{ \text{ext}_{\text{List}(A)}(I, s) = \text{tup}_A(I, s) \subseteq \text{List}(Y) \mid (I, s) \in \text{List}(X) \}$. The bridge $K \xrightarrow{\tau} \wp \tau \circ \text{inc}$ provides “restrictions” of the bridge $K \xrightarrow{\tau} \text{List}(Y)$ to subsets of $K$. 

4.2 Structure Morphisms.
Assume that we are given a schema morphism (Tbl. A.1 in §A.1)
\[ S_2 = \langle R_2, \sigma_2, X_2 \rangle \xrightarrow{(r, \varphi, f)} \langle R_1, \sigma_1, X_1 \rangle = S_1 \] (6)
consisting of a function on relation symbols \( R_2 \xrightarrow{r} R_1 \), a sort function \( X_2 \xrightarrow{f} X_1 \),
and a schema bridge \( \sigma_2 \cdot \Sigma_f \xrightarrow{\varphi} r \cdot \sigma_1 \), whose \( r^{th} \)-component is the morphism
\[ \Sigma_f(\sigma_2(r_2')) \xrightarrow{\varphi_{r_2'}} \sigma_1(r(r_2')) \text{ in } \text{List}(X_1). \]

**Definition 2.** A structure morphism \( \mathcal{M}_2 \xrightarrow{(r, \varphi, \beta, f, g)} \mathcal{M}_1 \) from source structure \( \mathcal{M}_2 = (\mathcal{E}_2, (\sigma_2, \tau_2), \mathcal{A}_2) \) to target structure \( \mathcal{M}_1 = (\mathcal{E}_1, (\sigma_1, \tau_1), \mathcal{A}_1) \) along (top Fig. 5) a schema morphism \( \text{Disp}(\phi) \) is a list designation morphism consisting of (back Fig. 5) an entity infomorphism \( \mathcal{E}_2 \xrightarrow{(r, k)} \mathcal{E}_1 \), and (front Fig. 5) an attribute infomorphism \( \mathcal{A}_2 \xrightarrow{(f, g)} \mathcal{A}_1 \). These are bridged by the above schema morphism and (bottom Fig. 5) a universe morphism \( \text{Univ}(\phi) \) with tuple bridge \( k \cdot \tau_2 \xrightarrow{\beta} \tau_1 \cdot \Sigma_g \), whose \( k^{th} \)-component is the morphism \( \tau_2(k(k_1)) \xrightarrow{\beta_{k_1}} \Sigma_g(\tau_1(k_1)) \) in \( \text{List}(Y_2) \), with equal arity
\[ \varphi_{r_2} \circ \text{arity}_{X_1} = \beta_{k_1} \circ \text{arity}_{Y_2} \]
when \( k(k_1) \models \epsilon_2(r_2') \text{ iff } k_1 \models \epsilon_1(r(r_2)). \)

![Fig. 5. Structure Morphism](image-url)

Let \( \text{Struc} \) denote the mathematical context of structures and their morphisms.
4.3 Transition.

In this section (§4.3), we define two forms of structures and structure morphisms: a classification (strict) form and an interpretation (lax) form. Fig. 5 illustrates the classification (strict) form; whereas Fig. 7 illustrates the interpretation (lax) form. Here we define a transition from the strict form to the lax form.

Note 1. By forgetting the full key sets and the full key function, the entity infomorphism becomes a (lax) entity infomorphism \( K_2 \xrightarrow{\kappa_2} K_1 \) consisting of an \( R_2 \)-indexed collection of key functions

\[
\{ \text{ext}_{E_2}(r_2) \xrightarrow{\kappa_{ext}_2} \text{ext}_{E_1}(r_2) \mid r_2 \in R_2 \}.
\]

Map \( k_1 \in \text{ext}_{E_2}(r(r_2)) \) with \( k_1 \models \epsilon_1, r(r_2) \) to \( \kappa_{ext}_2(k_1) = k(k_1) \in \text{ext}_{E_2}(r_2) \) with \( k(k_1) \models \epsilon_2, r_2 \). The collection of \( \text{List}(X_1) \)-signature morphisms \( \{ \Sigma_f(\sigma_2(r_2)) \xrightarrow{\phi_{r_2}} \Sigma_f(\sigma_1(r(r_2))) \mid r_2 \in R_2 \} \) defines the \( \text{List}(Y_1) \)-tuple functions

\[
\{ \text{tup}_{A_1}(\Sigma_f(\sigma_2(r_2))) \xleftarrow{\text{tup}_{A_1}(\Sigma_f(\sigma_1(r(r_2)))))} \mid r_2 \in R_2 \}.
\]

For any source predicate \( r_2 \in R_2 \), there is a bridge \( \kappa_{r_2} \cdot \bar{\tau}_{2,r_2} \xrightarrow{\beta_{r_2}} \bar{\tau}_{1,r(r_2)} \cdot \Sigma_g \)

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{transition_diagram.png}
\caption{Transition}
\end{figure}

that is a restriction of the bridge \( k \cdot \tau_2 \xrightarrow{\beta} \tau_1 \cdot \Sigma_g \) to the subset \( K_1(r(r_2)) \subseteq K_1 \), where \( \bar{\tau}_{2,r_2} = \tau_{2,r_2} \cdot \text{inc} \) and \( \bar{\tau}_{1,r(r_2)} = \tau_{1,r(r_2)} \cdot \text{inc} \). For each key \( k_1 \in K_1(r(r_2)) \), the \( k_1 \)-component of the bridge \( \beta_{r_2} \) is the \( \text{List}(Y_2) \)-morphism

\[
\bar{\tau}_{2,r_2}(\kappa_2(k_1)) \xrightarrow{\beta_{r_2}} \bar{\tau}_{1,r(r_2)}(k_1) \]

\text{Note 2.} Note that we have used the equal arity condition for a structure morphism (Def 2): \( \phi_{r_2} \circ \text{arity}_{X_1} = \beta_{k_1} \circ \text{arity}_{Y_2} \) when \( k(k_1) \models \epsilon_2, r_2 \iff k_1 \models \epsilon_1, r(r_2) \).
Proposition 1 (Key). Any structure morphism $\mathcal{M}_2 \xrightarrow{(r,k,\phi,\beta,f,g)} \mathcal{M}_1$ (Def\ref{def:structure-morphism}) satisfies the following condition: for any source predicate $r_2 \in R_2$,

$$\kappa_{r_2} \cdot \tau_{r_2} = \tau_{1,r(r_2)} \cdot \text{tup}_{\mathcal{A}_1}(\hat{\phi}_{r_2}) \cdot \hat{\tau}_{(f,g)}(\sigma_2(r_2)) \quad (\ast)$$

Proof. For suppose that $k_1 \in \text{ext}_{\mathcal{E}_1}(r(r_2))$ with $k_2 = \kappa_{r_2}(k_1) = k(k_1) \in \text{ext}_{\mathcal{E}_2}(r_2)$. Define $(I_1,t_1) = \tau_{1,r(r_2)}(k_1) = \tau_1(k_1) \in \text{tup}_{\mathcal{A}_1}(\sigma_1(r(r_2)))$ and $(I_2,t_2) = \tau_{2,r_2}(\kappa_{r_2}(k_1)) = \tau_2(k_2) \in \text{tup}_{\mathcal{A}_2}(\sigma_2(r_2))$. We have the List($Y_2$) morphism $(I_2,t_2) = \tau_2(k(k_1)) \xrightarrow{\beta_{k_1}} \sum_g(\tau_1(k_1)) = \sum_g(I_1,t_1)$.

Thus, $t_2 = h \cdot t_1 \cdot g$. This means that $(I_2,t_2) = \text{tup}(h,f,g)(I_1,t_1) = \hat{\tau}_{(f,g)}(\sigma_2(r_2))(\text{tup}_{\mathcal{A}_1}(h)(I_1,t_1)) = \hat{\tau}_{(f,g)}(\sigma_2(r_2))(\text{tup}_{\mathcal{A}_1}(\hat{\phi}_{r_2})(I_1,t_1))$. Hence, $\tau_{2,r_2}(\kappa_{r_2}(k_1)) = \hat{\tau}_{(f,g)}(\sigma_2(r_2))(\text{tup}_{\mathcal{A}_1}(\hat{\phi}_{r_2})(\tau_{1,r(r_2)}(k_1))) = \text{tup}(\hat{\phi}_{r_2},f,g)(\tau_{1,r(r_2)}(k_1))$, for all $k_1 \in \text{ext}_{\mathcal{E}_1}(r(r_2))$. \pagebreak

Corollary 1. For $r_2 \in R_2$, each bridge $\kappa_{r_2} \cdot \tau_{2,r_2} = \tau_{1,r(r_2)} \cdot \sum_g$ reduces to the composite $\beta_{r_2} = \tau_{1,r(r_2)} \cdot \tau_{r_2} \circ \sum_g$ for the bridge $\text{tup}_{\mathcal{A}_1}(\hat{\phi}_{r_2}) \circ \text{inc}$ associated with the function $\text{tup}_{\mathcal{A}_1}(\sum_f(\sigma_2(r_2)))$.

Proof. Straightforward. \pagebreak

27 The tuple bridge $\text{tup}_{\mathcal{A}_2} \xrightarrow{\beta(\sigma_2)} (\sum_f)^0 \text{tup}_{\mathcal{A}_1}$ is defined and used in the paper “The FOLE Table”\cite{fole-table}.

28 For tuple $(I_1,t_1) \in \text{tup}_{\mathcal{A}_1}(\sigma_1(r(r_2)))$, the $(I_1,t_1)^{th}$ component of $\tau_{r_2}$ is the List($Y_1$)-morphism $\text{tup}_{\mathcal{A}_1}(\hat{\phi}_{r_2})(I_1,t_1) = (I_1,h \cdot t_1) \xrightarrow{\tau_{r_2}(I_1,t_1)} (I_1,t_1)$.
4.4 Lax Structures.

To show equivalence between sound logics and databases, we need to use lax structures. \(^{29}\) Again, assume that we are given a schema \(S = \langle R, \sigma, X \rangle\) with a set of predicate symbols \(R\) and a signature (header) map \(R \sigma \rightarrow \text{List}(X)\).

**Definition 3.** A (lax) \(S\)-structure \(M = \langle E, \sigma, \tau, A \rangle\) consists of a lax entity classification \(E = \langle R, K \rangle\), an attribute classification (typed domain) \(A = \langle X, Y, |=_A \rangle\), the schema \(S = \langle R, \sigma, X \rangle\), and one of the two equivalent descriptions:

- either (MID Fig.\[4\]) a function \(R \xrightarrow{T_M} \text{Tbl}(A)\) consisting of an \(R\)-indexed collection of \(A\)-tables \(T_M(r) = \langle \sigma(r), K(r), \tau_r \rangle\);
- or (RHS Fig.\[4\]) a tuple bridge \(K \xrightarrow{\tau} \sigma \circ \text{tup}_A\) consisting of an indexed collection of tuple functions \(\{ K(r) \xrightarrow{\tau_r} \text{tup}_A(\sigma(r)) \mid r \in R \}\).

Hence, we can think of a lax structure \(M = \langle E, \sigma, \tau, A \rangle\) as extending the FOLE schema \(R \sigma \rightarrow \text{List}(X)\) to the tabular interpretation function \(R \xrightarrow{T_M} \text{Tbl}(A)\).

**Proposition 2.** A (lax) \(S\)-structure \(M = \langle E, \sigma, \tau, A \rangle\) defines, is the same as, the constraint-free aspect of a database \(R = \langle R, \sigma, A, K, \tau \rangle\) consisting of a set of predicate types \(R\), a key function \(R \xrightarrow{K} \text{Set}\), a schema \(S = \langle R, \sigma, X \rangle\) with a signature function \(R \sigma \rightarrow \text{List}(X)\), and a (constraint-free) tuple bridge \(K \xrightarrow{\tau} S \circ \text{tup}_A\) consisting of an \(R\)-indexed collection of tuple maps \(\{ K(r) \xrightarrow{\tau_r} \text{tup}_A(\sigma(r)) \mid r \in R \}\).

**Proof.** See the previous discussion.

4.5 Lax Structure Morphisms.

In order to make sound logic morphisms equivalent to database morphisms, we need to eliminate the global key function \(K_2 \leftarrow K_1\). To do this we define a lax version of Def.\[2\]. Note\[1\] eliminated the need for the global key function in the entity infomorphism; Prop.\[1\] and Cor.\[1\] eliminated the need for the universe morphism with its global key function. However, we do need to include the condition of Prop.\[1\].

\(^{29}\) A structure becomes lax when we forget the global set of keys \(K\) and use only a lax entity classification \(E = \langle R, K \rangle\). We can retrieve a “crisp” entity classification by defining the disjoint union of key subsets \(\bigsqcup_{r \in R} K(r)\). But, using the extent form of a structure (laxness), we have lost a certain coordination of tuple functions \(\{ \text{ext}_E(r) \xrightarrow{\tau_r} \text{tup}_A(\sigma(r)) \mid r \in R \}\) here, which may or may not be important.
Definition 4. For any two (lax) structures \( \mathcal{M}_2 = (\mathcal{E}_2, \sigma_2, \tau_2, \mathcal{A}_2) \) and \( \mathcal{M}_1 = (\mathcal{E}_1, \sigma_1, \tau_1, \mathcal{A}_1) \), a (lax) structure morphism (Tbl. 13 in §4.2) (Fig. 7)

\[
\mathcal{M}_2 = (\mathcal{E}_2, \sigma_2, \tau_2, \mathcal{A}_2) \xrightarrow{(r, \kappa, \phi, f, g)} (\mathcal{E}_1, \sigma_1, \tau_1, \mathcal{A}_1) = \mathcal{M}_1
\]

consists of (RHS Fig. 7) a schemed domain morphism

\[
\mathcal{D}_2 = (R_2, \sigma_2, \mathcal{A}_2) \xrightarrow{(r, \phi, f, g)} (R_1, \sigma_1, \mathcal{A}_1) = \mathcal{D}_1
\]

consisting of a typed domain morphism \( \mathcal{A}_2 \xrightarrow{(f, g)} \mathcal{A}_1 \) and a schema morphism \( \mathcal{S}_2 = (R_2, \sigma_2, X_2) \xrightarrow{(r, \phi, f)} (R_1, \sigma_1, X_1) = \mathcal{S}_1 \) (Disp 1 in §4.2) with common type function \( X_2 \xrightarrow{f} X_1 \), and (LHS Fig. 7) a lax entity infomorphism

\[
\mathcal{E}_2 = (R_2, \mathcal{K}_2) \xrightarrow{(r, \kappa)} (R_1, \mathcal{K}_1) = \mathcal{E}_1
\]

with a predicate function \( R_2 \xrightarrow{\xi} R_1 \) and a key bridge \( \mathcal{K}_2 \xleftarrow{\kappa} \mathcal{K}_1 \) consisting of the key functions \( \{ \mathcal{K}_2(r_2) \xleftarrow{\kappa} \mathcal{K}_1(r(r_2)) \mid r_2 \in R_2 \} \), which satisfy the condition

\[
\left\{ \kappa_{r_2} \cdot \tau_{r_2, r_2} = \tau_{1, r(r_2)} \cdot \text{tup}_{\mathcal{A}_1}(\hat{r}_{r_2}) \cdot \hat{\tau}_{(f, g)}(\sigma_2(r_2)) \mid r_2 \in R_2 \right\} \tag{7}
\]

See the (Key) Prop. 1

Corollary 2. A (lax) structure morphism \( \mathcal{M}_2 \xrightarrow{(r, \kappa, \phi, f, g)} \mathcal{M}_1 \) defines a tabular interpretation bridge function \( R_2 \xrightarrow{\xi} \text{mor}(\text{Tbl}) \), which maps a predicate symbol \( r_2 \in R_2 \) with image \( r(r_2) \in R_1 \) to the table morphism

\[
\begin{array}{c}
T_2(r_2) \\
\downarrow
\end{array} \xleftarrow{\kappa_{r_2}} \xrightarrow{\tau_{r_2}} \\
\begin{array}{c}
\text{tup}_{\mathcal{A}_2}(\sigma_2(r_2)) \xrightarrow{\uparrow_{\phi_{r_2}}} \text{tup}_{\mathcal{A}_1}(\sigma_1(\tau_{(r_2, r_2)})) \\
\mathcal{K}_2(r_2) \\
\downarrow
\end{array} \xleftarrow{\tau_{r', 1}} \xrightarrow{\kappa_{r, 1}} \\
\begin{array}{c}
\text{tup}_{\mathcal{A}_2}(\sigma_2(r_2)) \xrightarrow{\uparrow_{\phi_{r_2}}} \text{tup}_{\mathcal{A}_1}(\sigma_1(\tau_{(r_2, r_2)})) \\
\mathcal{K}_1(r(r_2)) \\
\downarrow
\end{array} \xrightarrow{\tau_{r_2}} \xrightarrow{T_1(r(r_2))} \xrightarrow{\tau_{r_2}} \xrightarrow{T_2(r_2)} \\
\text{source table} \xrightarrow{\text{tup}_{\mathcal{A}_1}(\sigma_2(r_2)) \xrightarrow{\uparrow_{\phi_{r_2}}}} \text{target table} \xrightarrow{\text{tup}_{\mathcal{A}_1}(\sigma_1(\tau_{(r_2, r_2)})) \xrightarrow{\uparrow_{\phi_{r_2}}}}
\]

where

\[
\kappa \cdot \tau_2 = (R^{\phi_0} \cdot \tau_1) \cdot (\hat{\tau}_{(f, g)} \cdot \text{tup}_{\mathcal{A}_1}) \cdot (S^{\phi_0}_2 \circ \hat{\tau}_{(f, g)}).
\]

See Def. 13 in §4.2

30 This is the constraint-free aspect of the database morphism condition
• $T_2(r_2) = (\sigma_2(r_2), A_2, K_2(r_2), \tau_2, r_2)$ is a table defined by the tabular interpretation function $R_2 \xrightarrow{T_2} \text{Tbl}(A_2)$ of structure $M_2$, and

• $T_1(r(r_2)) = (\sigma_1(r(r_2)), A_1, K_1(r(r_2)), \tau_1, r(r_2))$ is the table defined by the tabular interpretation function $R_1 \xrightarrow{T_1} \text{Tbl}(A_1)$ of structure $M_1$.

Proof. See Disp. 7 above. ■

Let $\text{Struc}$ denote the mathematical context of (lax) structures and their morphisms. Since any structure is a lax structure and any structure morphism is a lax structure morphism, there is an passage $\text{Struc} \xrightarrow{\text{lax}} \text{Struc}$.

Fig. 7. Lax Structure Morphism
5 FOLE Specifications.

A FOLE specification $\mathcal{R} = \langle R, S \rangle$ consists of a context $R$ of predicates linked by constraints and a diagram $S : R \rightarrow \text{Set}$ of lists. A specification morphism is a diagram morphism $\langle R, \zeta \rangle : \langle R_2, S_2 \rangle \Rightarrow \langle R_1, S_1 \rangle$ consisting of a shape-changing passage $R_2 \xrightarrow{R} R_1$ and a bridge $\zeta : S_2 \Rightarrow R \circ S_1$. Composition is component-wise. The mathematical context of FOLE specifications is denoted by $\text{SPEC} = \text{List}^u = (\_ \downarrow \text{List})$. The subcontext of FOLE specifications (with fixed sort sets and fixed sort functions) is denoted by $\text{Spec} \subseteq \text{SPEC}$.

5.1 Specifications.

Assume that we are given a schema $S = \langle R, \sigma, X \rangle$ with a set of predicate symbols $R$ and a signature (header) map $R \xrightarrow{\sigma} \text{List}(X)$.

**Definition 5.** A (formal) $S$-specification is a subgraph $R \sqsubseteq \text{Cons}(S)$, whose nodes are $S$-formulas and whose edges are $S$-constraints.

Let $\text{Spec}(S) = \wp\text{Cons}(S)$ denote the set of all $S$-specifications.

**Definition 6.** An (abstract) $S$-specification $T = \langle R, S, X \rangle$ consists of: a context $R$, whose objects $r \in R$ are predicate symbols and whose arrows $r' \xrightarrow{p} r$ are called abstract $S$-constraints, and a passage $R \xrightarrow{S} \text{List}(X)$, which extends the signature map $R \xrightarrow{\sigma} \text{List}(X)$ by mapping an abstract constraint $r' \xrightarrow{p} r$ to an $X$-signature morphism $S(r') = \sigma(r') = \langle I', s' \rangle = \langle I, s \rangle = \sigma(r) = S(r)$.

An abstract $S$-specification $T = \langle R, S, X \rangle$ has a companion formal $S$-specification $\hat{T} = \langle \hat{R}, \hat{S}, X \rangle$, whose graph $\hat{R} \sqsubseteq \text{Cons}(S)$ is the set of all formal constraints $\{ r' \xrightarrow{\sigma(p)}_h r \mid r' \xrightarrow{p} r \in R \}$. Since $\hat{R}$ is closed under composition and contains all identities, $\hat{T}$ is an abstract $S$-specification with signature passage $\hat{R} \xrightarrow{\hat{S}} \text{List}(X)$.

...
5.2 Specification Morphisms.

A FOLE (abstract) specification morphism in \( Spec \), with constant sort function \( f : X_2 \rightarrow X_1 \), is a diagram morphism \( \langle R, \zeta \rangle : \langle R_2, S_2 \rangle \rightarrow \langle R_1, S_1 \rangle \) in \( List \) consisting of a relation passage \( R : R_2 \rightarrow R_1 \) and a list interpretation bridge \( S_2 \Rightarrow R \circ S_1 \) that factors (Fig. 8)

\[
\zeta = (\hat{\varphi} \circ inc_{X_1}) \bullet (S_2 \circ i_f)
\]

(9)

through the fiber adjunction \( \text{List}(X_2) \xleftarrow{\text{List}_f} \text{List}(X_1) \) [35] in terms of

- some bridge \( \hat{\varphi} : S_2 \circ \Sigma_f \Rightarrow R^{op} \circ S_1 \) and
- the inclusion bridge \( i_f : \text{inc}_{X_2} \Rightarrow \Sigma_f \circ \text{inc}_{X_1} \).

We normally just use the bridge restriction \( \hat{\varphi} \) for the specification morphism. The original definition can be computed with the factorization in Disp. 9.

Definition 7. An (abstract) specification morphology (Tbl 13 in §4.1) [36] [37]

\[
T_2 = \langle R_2, S_2, X_2 \rangle \xrightarrow{(\hat{\varphi}, f)} \langle R_1, S_1, X_1 \rangle = T_1,
\]

along a schema morphism (Disp. 6 in §4.2) consists of a relation passage \( R_2 \xrightarrow{R} R_1 \) extending the predicate function \( R_2 \xrightarrow{f} R_1 \) of the schema morphism to constraints, the sort function \( X_2 \xrightarrow{f} X_1 \) of the schema morphism, and a bridge \( S_2 \circ \Sigma_f \Rightarrow R \circ S_1 \) extending the schema bridge to naturality. [38] [39]

As noted before, \( Spec \subseteq \text{SPEC} \) denotes the mathematical context of abstract FOLE specifications (with fixed sort sets). This context is the same as the context of database schemas and their morphisms.

There is an (abstract) S-specification morphism (LHS Fig. 9 in §5.3)

\[
T = \langle R, S, X \rangle \xrightarrow{R} \langle \overline{R}, \overline{S}, X \rangle = \overline{T}
\]

with an object-identical passage \( R \xrightarrow{R} \overline{R} \) that preserves signature

\[
R \xrightarrow{R} \overline{R} \xrightarrow{\overline{S}} \text{List}(X) = R \xrightarrow{S} \text{List}(X).
\]

[35] Fibered by signature over the adjunction \( \text{List}(X_2) \xleftarrow{\text{List}_f} \text{List}(X_1) \) (Kent [11]) representing list flow along a sort function \( f : X_2 \rightarrow X_1 \).

[36] Visualized on the right side Fig. 8.

[37] This is the same as a database schema morphism of §7.2.

[38] Naturality means that for any source constraint \( r_2 \xrightarrow{R(p_2)} r_2 \) with target constraint \( R(p_2) = r(r_2) \xrightarrow{p_1} r_1 = R(r_1) \), if their signature morphisms are \( \langle I'_2, s'_2 \rangle \xrightarrow{h_2} \langle I'_1, s'_1 \rangle \) and \( \langle I_2, s_2 \rangle \xrightarrow{h_1} \langle I_1, s_1 \rangle \), we have the naturality diagram \( \Sigma_f(h_2) \circ \hat{\varphi}_{r_2} = \hat{\varphi}_{r'_2} \circ h_1 \).

[39] An example of a non-trivial bridge is projection: assume that the arity functions \( I'_2 \xrightarrow{\varphi_{r'_2}} I'_1 \) and \( I_2 \xrightarrow{\varphi_{r_2}} I_1 \) are inclusion, thus defining signature projection, and assume that \( I'_2 \xrightarrow{h_2} I_2 \) is a restriction of \( I'_1 \xrightarrow{h_1} I_1 \).
5.3 Specification Satisfaction.

Assume that we are given a schema $\mathcal{S} = \langle R, \sigma, X \rangle$ with a set of predicate symbols $R$ and a signature (header) map $R \xrightarrow{\sigma} \text{List}(X)$.

**Definition 8.** (formal satisfaction) An $\mathcal{S}$-structure $M \in \text{Struc}(\mathcal{S})$ satisfies a (formal) $\mathcal{S}$-specification $T = \langle R, S, X \rangle$, symbolized $M \models_{\mathcal{S}} T$, when it satisfies every constraint in the specification: $M \models_{\mathcal{S}} T \iff M \models \sigma \circ \delta \subseteq R$. Hence, $M^S$ is the largest and most specialized (formal) $\mathcal{S}$-specification satisfied by $M$. \[40\]

**Definition 9.** (abstract satisfaction) An $\mathcal{S}$-structure $M \in \text{Struc}(\mathcal{S})$ satisfies an abstract $\mathcal{S}$-constraint $r' \xrightarrow{\sigma(p)} r$ in $R$, symbolized by $M \models_{\mathcal{S}} (r' \xrightarrow{\sigma(p)} r)$, when it satisfies the associated formal constraint $r' \xrightarrow{\sigma(p)} r$ in $\hat{R}$. An $\mathcal{S}$-structure $M \in \text{Struc}(\mathcal{S})$ satisfies (is a model of) an abstract $\mathcal{S}$-specification $T = \langle R, S, X \rangle$ when it satisfies the companion formal specification $\hat{T} = \langle \hat{R}, \hat{S}, X \rangle$: $M \models_{\mathcal{S}} T \iff M \models \sigma \circ \delta \subseteq R$.

**Definition 10.** When $M \models_{\mathcal{S}} T$ holds, the abstract table passage $R_{\text{op}} \xrightarrow{T} \text{Rel}(A)$ is defined to be the composition (RHS Fig. [4]) of the object-identical passage $R \xrightarrow{R} \hat{R}$ with the “inclusion” $\hat{R} \subseteq M^S$ and the relation interpretation passage $M^S_{\text{op}} \xrightarrow{R_{\text{op}}} \text{Rel}(A)$ of Lem. [2] in §3.3. \[41\]

Note that the the abstract table passage $R_{\text{op}} \xrightarrow{T} \text{Rel}(A)$ extends the table-valued function $R \xrightarrow{T_M} \text{Tbl}(A)$ of the structure $M$ to constraints.

**Proposition 3 (Key).** Satisfaction is equivalent to tabular interpretation.

**Proof.** On the one hand, if $M \models_{\mathcal{S}} T$, then the tabular interpretation passage $R_{\text{op}} \xrightarrow{T} \text{Rel}(A) \xrightarrow{\text{in}_{A}} \text{Rel}(A)$ maps a constraint $r' \xrightarrow{\sigma(p)} r$ in $R$ to the $A$-relation $\mathcal{A}$. So, $M^S$ is not just a mathematical context, but also an $\mathcal{S}$-specification.

\[40\] The passage $M^S_{\text{op}} \xrightarrow{R_{\text{op}}} \text{Rel}(A)$ was defined in Lemma [2] of §3.3 on satisfaction.
The morphism

\[ T(r') = \langle S(r'), \phi t_{r'}(K(r')) \rangle \xrightarrow{T(p)} \langle S(r), \phi t_r(K(r)) \rangle = T(r) \]

as pictured in Fig. 3 of §3.3.

On the other hand, if there is a tabular interpretation passage \( R^{op} \xrightarrow{T} \text{Tbl}(A) \), then the adjoint flow of \( A \)-tables\(^{43}\) demonstrates that the \( S \)-structure \( M \) satisfies each abstract \( S \)-constraint \( r' \xrightarrow{T} r \) in \( R \) (see Disp.4 in §3.3). □

\[ \text{specification passages:} \]

\[ \begin{array}{c}
R & \xrightarrow{R} \tilde{R} & \xleftarrow{\text{inc}} \text{Cons}(S) \\
S && S \\
\text{List}(X) && \text{List}(X)
\end{array} \]

\[ \text{with satisfaction} \]

\[ \begin{array}{c}
R^{op} & \xrightarrow{R^{op}} \tilde{R} & \xleftarrow{\text{inc}} M^{S^{op}} \\
\text{Rel}(A) && \text{Tbl}(A)
\end{array} \]

\[ \text{general} \]

**Table 7. Comparisons**

\[ \begin{array}{l}
\text{(formal) } S\text{-specification – subgraph:} & R \subseteq \text{Cons}(S) \\
\text{signature passage:} & \text{Cons}(S) \rightarrow \text{List}(X) \\
\text{(abstract) } S\text{-specification – passage:} & R \xrightarrow{S} \text{List}(X)
\end{array} \]

\[ \begin{array}{l}
\text{(formal) } S\text{-specification – subgraph:} & R \subseteq M^S \\
\text{table passage:} & M^{S^{op}} \xrightarrow{R_M} \text{Rel}(A) \xleftarrow{\text{inc}} \text{Tbl}(A) \\
\text{(abstract) } S\text{-specification – table passage:} & R^{op} \xrightarrow{T} \text{Rel}(A) \xleftarrow{\text{inc}} \text{Tbl}(A)
\end{array} \]

\[ ^{42} \text{We use relations here rather than tables, since (Lem.} 2 \text{of } \S3.3): \text{relational interpretation is closed under composition; but tabular interpretation is closed under composition only up to key equivalence.} \]

\[ ^{43} \text{See the discussion about type domain indexing in } \S3.4.1 \text{ of the paper} \[11]. \]
6 FOLE Sound Logics.

6.1 Sound Logics.

**Definition 11.** A (lax) sound logic $L = \langle S, M, T \rangle$ with a schema $S = \langle R, \sigma, X \rangle$, consists of a (lax) $S$-structure $M = \langle E, \sigma, \tau, A \rangle$ and an (abstract) $S$-specification $T = \langle R, S, X \rangle$, where the structure $M$ satisfies the specification $T$: $M \models_S T$.

**Proposition 4.** Any (lax) FOLE sound logic $L = \langle S, M, T \rangle$ defines a FOLE relational database with constant type domain $R = \langle R, T, A \rangle$.

**Proof.** By the definition of satisfaction (Def. 9 and Def. 11 in §5.3), the tabular interpretation $R^T \rightarrow \text{Tbl}(A)$ (Def. 3 of §4.4) extends to a table passage $R^{op} \rightarrow \text{Rel}(A) \subseteq \text{Tbl}(A)$, which forms a relational database $R = \langle R, T, A \rangle$.

| L = ⟨S, M, T⟩ | schema $S = \langle R, \sigma, X \rangle$ |
|---------------|-----------------------------------------------|
| signature function $R \sigma \rightarrow \text{List}(X)$ |
| (lax) $S$-structure $M = \langle E, \sigma, \tau, A \rangle$ |
| (lax) entity classification $E = \langle R, K \rangle$ |
| attribute classification $A = \langle X, Y, \models_A \rangle$ |
| tuple bridge $K \xRightarrow{T} \sigma \circ \text{tup}_A$ |
| table function $R^T \rightarrow \text{Tbl}(A)$ |
| $S$-specification $T = \langle R, S, X \rangle$ |
| signature passage $R^S \rightarrow \text{List}(X)$ |
| satisfaction $M \models_S T$ |
| table passage $R^{op} \rightarrow \text{Rel}(A) \xleftarrow{\text{inc}_A} \text{Tbl}(A)$ |

**Table 8. Sound Logic**

---

44 A (lax) FGLE sound logic is described in terms of its components in Table 8. In §7.1, a FOLE relational database is defined in Def. 13 and is described in terms of its components in Table 10.
6.2 Sound Logic Morphisms.

Definition 12. For any two sound logics $L_2 = \langle S_2, M_2, T_2 \rangle$ and $L_1 = \langle S_1, M_1, T_1 \rangle$, a (lax) sound logic morphism (Tab. 1 in §A.1.3) 

$$L_2 = \langle S_2, M_2, T_2 \rangle \xrightarrow{(R, \kappa, \phi, f, g)} \langle S_1, M_1, T_1 \rangle = L_1$$

consists of a (lax) structure morphism 

$$M_2 = \langle \varepsilon_2, \langle \sigma_2, \tau_2 \rangle, A_2 \rangle \xrightarrow{(r, \kappa, \phi, f, g)} \langle \varepsilon_1, \langle \sigma_1, \tau_1 \rangle, A_1 \rangle = M_1$$

and an (abstract) specification morphism 

$$T_2 = \langle R_2, S_2, X_2 \rangle \xrightarrow{(R, \phi, f)} \langle R_1, S_1, X_1 \rangle = T_1$$

along a common schema morphism 

$$S_2 = \langle R_2, \sigma_2, X_2 \rangle \xrightarrow{(r, \phi, f)} \langle R_1, \sigma_1, X_1 \rangle = S_1.$$ 

Let $\text{Snd}$ denote the context of (lax) sound logics and their morphisms.

Note 2. At this point we know that a (lax) sound logic morphism 

$$L_2 = \langle S_2, M_2, T_2 \rangle \xrightarrow{(R, \kappa, \phi, f, g)} \langle S_1, M_1, T_1 \rangle = L_1$$

has tabular interpretation passages $R_2 \xrightarrow{\text{T}_2} \text{Rel}(A_2) \subseteq \text{Tbl}(A_2)$ and $R_1 \xrightarrow{\text{T}_1} \text{Rel}(A_1) \subseteq \text{Tbl}(A_1)$ at the source and target sound logics and a predicate passage $R_2 \xrightarrow{R} R_1$. We complete the picture in Prop. 5 by proving that sound logic morphisms preserve satisfaction in some way.

Proposition 5. There is a (tabular interpretation) bridge

$$\xi = (\psi \circ \text{inc}_{A_1}) \bullet (\text{T}_2 \circ \chi_{(f,g)}) : T_2 \leftarrow R_{0}^{\text{op}} \text{T}_1$$

that extends the collection of tabular interpretation bridge functions

$$\{ T_2(r_2) \xleftarrow{\xi_{r_2}} T_1(r(\text{r}_2)) \mid r_2 \in R_2 \}$$

(see Cor. 2 of §4.3) to constraints: for any source constraint $r_1' \xrightarrow{p_2} r_2$ in $R_2$ with $R$-image target constraint $r_1' \xrightarrow{p_1} r_1$ in $R_1$, we have the following naturality diagram, which represents “preservation or linkage of satisfaction”.45

![Naturality Diagram](image)

45 The sound logic morphism $L_2 \xrightarrow{(R, \kappa, \phi, f, g)} L_1$ maps the table morphism

$$T_1(r_1') \xrightarrow{T_1(p_1)} T_1(r_1)$$

representing the satisfaction $M_1 \models S_1 (r_1' \xrightarrow{p_1} r_1)$ to the table morphism $T_2(r_1') \xrightarrow{T_2(p_2)} T_2(r_2)$ representing the satisfaction $M_2 \models S_2 (r_1'^p_2 \xrightarrow{p_2} r_2)$. 
Proof. The above diagram is expanded into more detail in Fig. 10. To understand this, we discuss each part (facet) separately. In short: the front/back is due to Cor. 2 in § 5.3 for (lax) structure morphisms; the left/right hold by satisfaction of source/target sound logics; the bottom-right is due to the (abstract) specification morphism; the bottom-left is due to naturality of $\text{tup}_A$; and the top holds for relations. We now discuss each facet in more detail.

**front/back:** For each source predicate $r_2 \in R_2$, the (lax) structure morphism $M_2 \xrightarrow{(r, \hat{\varphi}, f, g)} M_1$ defines (Cor. 2 of § 5.3) the table morphism

$$\langle \sigma(r_2), A_2, \text{ext}_{\xi_2}(r_2), \tau_{r_2} \rangle \xleftarrow{\hat{\varphi}_{r_2}: f, g, r_2} \langle \sigma(r_2), A_1, \text{ext}_{\xi_1}(r_1), \tau_{r_1} \rangle$$

(same for $r_2' \in R_2$).

**left/right (sat):** Satisfaction for source/target sound logics (§ 5.3) define table morphisms

$$\langle \sigma(r'_1), \text{ext}_{\xi}(r'_1), \tau_{r'_1} \rangle \xleftarrow{\langle h_{p_1}, k_{p_1} \rangle \xrightarrow{T_i(p_i)} \tau_{r_i}} \langle \sigma(r_i), \text{ext}_{\xi}(r_i), \tau_{r_i} \rangle,$$

which are equivalent to naturality of the bridges $K_i \xrightarrow{T_i \circ \tau_{A_1}} S_i \circ \text{tup}_A$.

**bottom right:** The structure and specification morphisms have the same underlying schema morphism $S_2 = \langle R_2, \sigma_2, X_2 \rangle \xrightarrow{(r, \hat{\varphi})} \langle R_1, \sigma_1, X_1 \rangle = S_1$. Apply the tuple function $\text{tup}_A$ to the naturality of the bridge $S_2 \circ \sum_f \xrightarrow{\hat{\varphi}} R \circ S_1$ for any abstract $S$-constraint $\langle r_2' \xrightarrow{p_2} r_2 \rangle$ in $R_2$: for any constraint $\langle r_2' \xrightarrow{p_2} r_2 \rangle$ in $R_2$, the following diagram commutes.
FOLE Equivalence

\[ \sum_f (\sigma (r')) \xrightarrow{\phi_{r'}} \sigma_1 (r') \]

\[ \sum_f (h_{p'_2}) = h_{p'_2} \xrightarrow{\phi \text{ naturality}} h_{p_1} \]

\[ \sum_f (\sigma (r)) \xrightarrow{\phi_r} \sigma_1 (r') \]

**bottom left:** The tuple bridge \( \text{tup}_{A_2} \xrightarrow{\tau(f,g)} \sum_f \circ \text{tup}_{A_1} \) of the type domain morphism \( A_2 \xrightarrow{(f,g)} A_1 \) (see footnote 27 in §4.2) satisfies naturality: for any constraint \( r'_2 \xrightarrow{p_2} r_2 \) in \( R_2 \), the following diagram commutes.

\[ \text{tup}_{A_2} (\sigma (r')) \xrightarrow{\tau(f,g)(\sigma (r'))} \text{tup}_{A_1} (\sum_f (\sigma (r'))) \]

\[ \text{tup}_{A_2} (\sigma (r)) \xrightarrow{(-) \cdot g \text{ naturality}} \text{tup}_{A_1} (\sum_f (\sigma (r'))) \]

**top:** By the other naturality conditions just proven, for any constraint \( r'_2 \xrightarrow{p_2} r_2 \) in \( R_2 \) with \( R \)-image \( r'_1 \xrightarrow{p_1} r_1 \) in \( R_1 \), we know that \( k_{p_1} \cdot \kappa_{r'_2} \cdot \tau_{r'_2} = \kappa_{r_2} \cdot k_{p_2} \cdot \tau_{r_2} \).

If the tuple map \( \text{ext}_{\mathcal{E}_2} (r'_2) \xrightarrow{\tau_{r'_2}} \text{tup}_{A_2} (\sigma (r'_2)) \) were injective, the bridge \( \text{ext}_{\mathcal{E}_2} \xrightarrow{\kappa \circ \text{ext}_{\mathcal{E}_1}} \text{ext}_{\mathcal{E}_1} \) would satisfy “extent naturality”:

\[ \text{ext}_{\mathcal{E}_2} (r'_2) \xrightarrow{k_{p_2}} \text{ext}_{\mathcal{E}_1} (r'_1) \]

\[ \text{ext}_{\mathcal{E}_2} (r_2) \xrightarrow{k} \text{ext}_{\mathcal{E}_1} (r_1) \]

Here, we use the image part of the table-relation reflection \( \langle \text{im} \dashv \text{inc} \rangle : \text{Tbl} \rightleftarrows \text{Rel} \) Then, we use diagonal fill-in Fig. 10. Hence, extent naturality holds for the relational interpretation into \( \text{Rel} \).

---

46 The reflection \( \langle \text{im} \dashv \text{inc} \rangle : \text{Tbl} \rightleftarrows \text{Rel} \) (“The FOLE Table”[11]) of the context of relations into the context of tables embodies the notion of “informational equivalence”.
Proposition 6. A (lax) sound logic morphism
\[
\mathcal{L}_2 = \langle S_2, M_2, \mathcal{R}_2 \rangle \xrightarrow{(R, \kappa, \hat{\varphi}, f, g)} \langle S_1, M_1, \mathcal{R}_1 \rangle = \mathcal{L}_1
\]
defines a database morphism
\[
\mathcal{R}_2 = \langle R_2, S_2, A_2, K_2, \tau_2 \rangle \xrightarrow{(R, \kappa, \hat{\varphi}, f, g)} \langle R_1, S_1, A_1, K_1, \tau_1 \rangle = \mathcal{R}_1
\]
whose tabular interpretation bridge
\[
\xi = (\hat{\psi} \circ \text{inc}_{A_1}) \cdot (T_2 \circ \hat{\chi}_{f,g})
\]
through the fiber adjunction \(\text{Tbl}(A_2) \xleftarrow{\text{tbl}_{f,g}} \text{Tbl}(A_1)\).

Proof. By Prop. 4 the sound logics \(\mathcal{L}_2\) and \(\mathcal{L}_1\) define two databases \(\mathcal{R}_2\) and \(\mathcal{R}_1\). By Prop. 5 the (lax) sound logic morphism \(\mathcal{L}_2 \xrightarrow{(R, \kappa, \hat{\varphi}, f, g)} \mathcal{L}_1\) defines a database morphism \(\langle R_2, T_2 \rangle \xleftarrow{(R, \kappa)} \langle R_1, T_1 \rangle\), whose tabular interpretation bridge \(T_2 \xleftarrow{\xi} R \circ T_1\) factors as above.

Theorem 1. There is a passage \(\text{Snd} \xrightarrow{\text{db}} \text{Db}\).

Proof. A sound logic \(\mathcal{L} = \langle S, M, T \rangle\) is mapped to its associated database \(\mathcal{R} = \langle R, K, S, \tau, A \rangle\) by Prop. 4. A sound logic morphism \(\mathcal{L}_2 \xrightarrow{(R, \kappa, \hat{\varphi}, f, g)} \mathcal{L}_1\) is mapped to its associated database morphism \(\mathcal{R}_2 \xrightarrow{(R, \kappa, \hat{\varphi}, f, g)} \mathcal{R}_1\) by Prop. 5.

| sound logic morphism | \(\mathcal{L}_2 = \langle S_2, M_2, \mathcal{R}_2 \rangle \xrightarrow{(R, \kappa, \hat{\varphi}, f, g)} \langle S_1, M_1, \mathcal{R}_1 \rangle = \mathcal{L}_1\) |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------|
| • schema morphism    | \(S_2 = \langle R_2, \sigma_2, X_2 \rangle \xrightarrow{(r, \phi, f)} \langle R_1, \sigma_1, X_1 \rangle = S_1\) |
| • (lax) struc morph  | \(M_2 = \langle E_2, \mathcal{E}_2 \rangle \xrightarrow{(r, \kappa, \hat{\varphi}, f, g)} \langle (R_1, K_1), \sigma_1, \tau_1, A_1 \rangle = M_1\) |
| ○ type domain morph  | \(A_2 = \langle X_2, Y_2, \models_{A_2} \rangle \xrightarrow{(f, \varphi)} \langle X_1, Y_1, \models_{A_1} \rangle = A_1\) |
| ○ (lax) entity infomorphism | \(K_2 = \langle R_2, K_2 \rangle \xrightarrow{(R, \kappa)} \langle R_1, K_1 \rangle = K_1\) |
| which satisfy the constraint | \(\kappa \cdot \tau_2 = (R^0, \tau_2) \cdot (\hat{\varphi}^0 \circ \text{tup}_{A_1}) \cdot (S^0 \circ \varphi)\) |
| • specification morphism | \(T_2 = \langle R_2, S_2, X_2 \rangle \xrightarrow{(R, \varphi, f)} \langle R_1, S_1, X_1 \rangle = T_1\) |

Table 9. Sound Logic Morphism

---

47 See Def. 16 and Tbl. 11 in §7.2
7 FOLE Databases.

A FOLE relational database \( R = \langle R, T \rangle \) consists of a context \( R \) of predicates linked by constraints and an interpretation diagram \( T : R^{\text{op}} \to \text{Tbl} \) of tables. A relational database morphism \( \langle R, \xi \rangle : \langle R_2, T_2 \rangle \leftarrow \langle R_1, T_1 \rangle \) (Tbl. 13 in §A.1) is a diagram morphism (LHS Fig. 12 in §7.2) consisting of a shape-changing passage \( R_2 \xrightarrow{R} R_1 \) and a bridge \( \xi : T_2 \leftarrow R^{\text{op}} \circ T_1 \). Composition is component-wise. The mathematical context of FOLE relational databases is denoted by \( \text{DB} = \text{Tbl}^u = ((-)^{\text{op}} \downarrow \text{Tbl}) \).

7.1 Databases.

**Definition 13.** A FOLE database \( R = \langle R, T, A \rangle \), with constant type domain \( A \), is a database with an interpretation diagram \( T : R^{\text{op}} \to \text{Tbl}(A) \hookrightarrow \text{Tbl} \) that factors through the context of \( A \)-tables.

**Fig. 11.** Database: Type Domain

**Definition 14.** Using \( A \)-table projection passages \(^{49}\) (Fig. 11), a database \( R = \langle R, K, S, \tau, A \rangle \), with constant type domain \( A \), consists of

- a context \( R \) of predicates,
- a relational database schema \( S = \langle R, S, X \rangle \) with signature diagram \( S = T^{\text{op}} \circ \text{sign}_A : R \to \text{List}(X) \),
- a key diagram \( K = T \circ \text{key}_A : R^{\text{op}} \to \text{Set} \), and
- a tuple bridge \( \tau = T \circ \tau_A : K \Rightarrow S^{\text{op}} \circ \text{tup}_A \).

\(^{48}\) See §4.2.1 in the paper “The FOLE Database” [12].

\(^{49}\) See §3.4.1 of the paper “The FOLE Table” [11].
Proposition 7. The constraint-free aspect of a FOLE database $\mathcal{R} = (\mathbb{R}, T, \mathcal{A})$, with constant type domain $\mathcal{A}$, is the same as a (lax) FOLE structure $\mathcal{M} = (\mathcal{E}, \sigma, \tau, \mathcal{A})$ in Struct.

Proof. We define the various components of the (lax) structure $\mathcal{M} = (\mathcal{E}, \sigma, \tau, \mathcal{A})$ listed in Def. 3 of §4.4 and pictured in Fig. 4 of §3.1.

$\mathcal{A}$: The attribute classification (typed domain) $\mathcal{A} = (X, Y, \vdash \mathcal{A})$ is given.

$\sigma$: The schema (type hypergraph) $\mathcal{S} = (R, \sigma, X)$ consists of the set $R$ of relation symbols (predicates) and the signature map $\sigma : R \rightarrow \text{List}(X) : r \mapsto \sigma(r) = \mathcal{S}(r)$. This is the constraint-free aspect of the database schema $\mathcal{S} = (R, S, X)$ with signature diagram $\mathcal{S} : R \rightarrow \text{List}(X)$.

$\mathcal{E}$: The (lax) entity classification $\mathcal{E} = (R, K)$ consists of the set $R$ of relation symbols (predicates) and the key function $R \xrightarrow{K} \text{Set}$. This is the constraint-free aspect of the key diagram $\mathcal{R}^{\text{op}} \xrightarrow{K} \text{Set}$.

$\tau$: The tuple bridge $\tau : K \Rightarrow \sigma \circ \text{tup}_{\mathcal{A}}$ is the constraint-free aspect of the tuple bridge $\mathcal{R}^{\text{op}} \xrightarrow{\tau} \mathcal{S}^{\text{op}} \circ \text{tup}_{\mathcal{A}}$.

Proposition 8. Any FOLE database $\mathcal{R} = (\mathbb{R}, T, \mathcal{A}) = (\mathbb{R}, K, S, \tau, \mathcal{A})$ in $\mathrm{Db}(A)$ defines a (lax) FOLE sound logic $\mathcal{L} = (\mathcal{S}, \mathcal{M}, T)$ in $\mathrm{Snd}$.

Proof. The schema $\mathcal{S} = (R, S, X)$ is the structure schema (mentioned above), the constraint-free aspect of the database schema $(R, S, X)$. The $\mathcal{S}$-structure $\mathcal{M} = (\mathcal{E}, \sigma, \tau, \mathcal{A})$ is defined in Prop. 7 above. The (abstract) $\mathcal{S}$-specification $\mathcal{T} = (R, S, X)$ is the same as the database schema, as discussed in §5.4. By Prop 8 in §5.3, the tabular interpretation passage $\mathcal{R}^{\text{op}} \xrightarrow{T} \text{Tbl}(A)$ demonstrates satisfaction $\mathcal{M} \models_{\mathcal{S}} \mathcal{T}$.
7.2 Database Morphisms.

A FOLE database morphism, with constant type domain morphism \( (f, g) : A_2 \rightarrow A_1 \), is a FOLE database morphism \( (R, \xi) : (R_2, T_2) \leftarrow (R_1, T_1) \), whose tabular interpretation bridge \( T_2 \leftarrow R \circ T_1 \) factors (Fig.12) through the fiber adjunction \( Tbl(A_2) \leftrightarrow \text{tbl}(f,g) \ Tbl(A_1) \) in terms of

- some bridge \( \hat{\psi} : T_2 \circ \text{tbl}(f,g) \leftarrow R^{op} \circ T_1 \) and
- the inclusion bridge \( \hat{\chi}(f,g) : \text{inc}_{A_2} \leftarrow \text{tbl}(f,g) \circ \text{inc}_{A_1} \).

We normally just use the bridge restriction \( \hat{\psi} \) for the database morphism. The original definition can be computed with the factorization in Disp.10.

![Fig. 12. Database Morphism](image)

The inclusion bridges for the table fiber adjunction \( Tbl(A_2) \leftrightarrow \text{tbl}(f,g) \ Tbl(A_1) \)

---

50 There is a table fiber adjunction \( Tbl(A_2) \leftrightarrow \text{tbl}(f,g) \ Tbl(A_1) \) representing tabular flow along a type domain morphism \( A_2 \leftrightarrow A_1 \). See §3.4.2 of Kent [11].

The fiber passage \( Tbl(A_2) \leftrightarrow \text{tbl}(f,g) \ Tbl(A_1) \) is defined in terms of the tuple bridge \( f^{op} \circ \text{tup}_{A_2} \leftarrow \text{tup}_{A_1} \) and the substitution function \( \text{List}(X_2) \leftrightarrow \text{List}(X_1) \).

The adjoint fiber passage \( Tbl(A_2) \rightarrow \text{tbl}(f,g) \ Tbl(A_1) \) is defined in terms of the adjoints, the tuple bridge \( \Sigma f^{op} \circ \text{tup}_{A_2} \leftarrow \text{tup}_{A_1} \) and the existential quantifier function \( \text{List}(X_2) \rightarrow \Sigma f \text{List}(X_1) \).

51 Equivalently, in terms of their \text{levo} bridge adjoints in Tbl.12 of §3.1.

52 Notation from § 3.4.2.
Fig. 13. Inclusion Bridge: Tables

are defined by

\[
\begin{align*}
\chi_{(f,g)} : \textsf{tbl}_{(f,g)} \circ \textsf{inc}_{A_2} &\simeq \textsf{inc}_{A_1} \\
\hat{\chi}_{(f,g)} &= (\eta_{(f,g)} \circ \textsf{inc}_{A_1}) \ast (\textsf{tbl}_{(f,g)} \circ \hat{\chi}_{(f,g)}) \\
\hat{\chi}_{(f,g)} &= (\textsf{tbl}_{(f,g)} \circ \hat{\chi}_{(f,g)}) \ast (\varepsilon_{(f,g)} \circ \textsf{inc}_{A_2})
\end{align*}
\]

\[
\begin{array}{c}
\xymatrix{
\text{Tbl}(A_2) \ar[rr]^{\text{tbl}_{(f,g)}} \ar[d]_{\text{inc}_{A_2}} & & \text{Tbl}(A_1) \ar[d]^{\text{inc}_{A_1}} \\
\text{Tbl} & & \text{Tbl}
}
\end{array}
\]
Definition 15. For any two databases \( R_2 = \langle R_2, T_2, A_2 \rangle \) and \( R_1 = \langle R_1, T_1, A_1 \rangle \), a database morphism, with constant type domain morphism \( A \xrightarrow{(f,g)} A_1 \), (Fig. 12)

\[
R_2 = \langle R_2, T_2, A_2 \rangle \xleftarrow{\langle R, \psi, f, g \rangle} \langle R_1, T_1, A_1 \rangle = R_1
\]

(Tbl. 13 in §A.1) consists of a shape-changing relation passage \( R : R_2 \rightarrow R_1 \) and a bridge \( \psi : T_2 \circ \text{tbl}(f,g) \xleftarrow{R_{\text{op}} \circ T_1} \).

The subcontext of FOLE relational databases (with constant type domains and constant type domain morphisms) is denoted by \( \text{Db} \subseteq \text{DB} \).

\[53\] See §4.2.2 in the paper “The FOLE Database” [12].
Definition 16. Using projections (Fig. 14), a FOLE database morphism, with constant type domain morphism $\mathcal{A}_2 \xrightarrow{(f,g)} \mathcal{A}_1$,

$$\mathcal{R}_2 = \langle \mathcal{R}_2, \mathcal{S}_2, \mathcal{K}_2, \tau_2 \rangle \xleftarrow{(R,\kappa,\dot{f},\dot{g})} \langle \mathcal{R}_1, \mathcal{S}_1, \mathcal{K}_1, \tau_1 \rangle = \mathcal{R}_1$$

(Tbl. 13 in §A.1) consists of a relation passage $\mathcal{R}_2 \xrightarrow{R} \mathcal{R}_1$ as above, and

- a schema bridge $\dot{\varphi} = \psi'^{\text{op}} \circ \text{sign} : \mathcal{S}_2 \circ \Sigma_f \Rightarrow \mathcal{R} \circ \mathcal{S}_1$,
- a key bridge $\kappa = \psi \circ \text{key} : \mathcal{K}_2 \leftarrow \mathcal{K}_1$ consisting of an $\mathcal{R}_2$-indexed collection $\{ \mathcal{K}_2(\mathcal{r}_2) \xleftarrow{\kappa(\mathcal{r}_2)} \mathcal{K}_1(\mathcal{r}(\mathcal{r}_2)) \mid \mathcal{r}_2 \in \mathcal{R}_2 \}$ of key functions.

These components satisfy the condition

$$\kappa \cdot \tau_2 = (R'^{\text{op}} \circ \tau_1) \bullet (\varphi'^{\text{op}} \circ \text{tup}_{\mathcal{A}_1}) \bullet (S_2'^{\text{op}} \circ \dot{\tau}(f,g)), \quad (11)$$

as pictured in the following bridge diagram.

$\begin{array}{c}
\xymatrix{
\mathcal{R}^{op} \circ \mathcal{K}_1 \ar[r]^{R'^{op} \circ \tau_1} & \mathcal{R}^{op} \circ \mathcal{S}_1^{op} \circ \text{tup}_{\mathcal{A}_1} \ar[d]_{\Sigma^{op} \circ \text{tup}_{\mathcal{A}_1}} \\
\mathcal{K}_2 \ar[r]_{\tau_2} \ar[u]^{\varphi'^{op} \circ \text{tup}_{\mathcal{A}_1}} \ar[d]^\kappa & \mathcal{S}_2^{op} \circ \text{tup}_{\mathcal{A}_2} \ar[r]_{\Sigma^{op} \circ \text{tup}_{\mathcal{A}_2}} \ar[u]_{\dot{\tau}(f,g)} & \\
\text{lax entity infomorphism} & \text{tuple attribute infomorphism} & \\
}\
\end{array}$

Table 11. Database Morphism

---

54 This defines a database schema morphism $\mathcal{T}_2 = \langle \mathcal{R}_2, \mathcal{S}_2, X_2 \rangle \xleftarrow{(R,\dot{f},\dot{g})} \langle \mathcal{R}_1, \mathcal{S}_1, X_1 \rangle = \mathcal{T}_1$, which is the same as the (abstract) specification morphism in Def. 7 of §5.2.

55 This defines a lax entity infomorphism $\mathcal{K}_2 \leftarrow \mathcal{K}_1$ consisting of an $\mathcal{R}_2$-indexed collection of key functions $\{ \mathcal{K}_2(\mathcal{r}_2) \xleftarrow{\kappa(\mathcal{r}_2)} \mathcal{K}_1(\mathcal{r}(\mathcal{r}_2)) \mid \mathcal{r}_2 \in \mathcal{R}_2 \}$.
Proposition 9. The constraint-free aspect of a \textsc{fole} database morphism

\[ R_2 = (R_2, S_2, A_2, K_2, \tau_2) \xleftarrow{(r, \varphi, f, g)} (R_1, S_1, A_1, K_1, \tau_1) = R_1, \]

in \textit{Db} defines, is the same as, a (lax) \textsc{fole} structure morphism

\[ M_2 = (E_2, \sigma_2, \tau_2, A_2) \xrightarrow{(r, \varphi, f, g)} (E_1, \sigma_1, \tau_1, A_1) = M_1 \]

in \textit{Struc}.

\textit{Proof.} We define the various components of the (lax) structure morphism above, as defined in Def\[\text{4}\] and pictured in Fig.\[\text{4}\] of §\[\text{4.3}\].

\( r \): The predicate function \( R_2 \xrightarrow{R} R_1 \) is the constraint-free aspect of the relation passage \( R_2 \xrightarrow{R} R_1 \).

\( \langle f, g \rangle \): The type domain morphism \( A_2 \xrightarrow{(f, g)} A_1 \) is given.

\( \langle r, \varphi, f \rangle \): The schema morphism \( S_2 = (R_2, \sigma_2, X_2) \xrightarrow{(r, \varphi, f)} (R_1, \sigma_1, X_1) = S_1 \), consisting of the \( R_2 \)-indexed collection of signature morphisms \( \{ \sigma_2(r_2) \xrightarrow{h} \sigma_1(r_2) \mid r_2 \in R_2 \} \), is the constraint-free aspect of the database schema morphism in Def\[\text{16}\].

\( \langle r, \kappa \rangle \): The lax entity infomorphism \( E_2 = (R_2, K_2) \xrightarrow{(r, \kappa)} (R_1, K_1) = E_1 \), consisting of the \( R_2 \)-indexed collection of key functions \( \{ K_2(r_2) \xrightarrow{\kappa_{r_2}} K_1(r_2) \mid r_2 \in R_2 \} \), is the constraint-free aspect of the key bridge \( \kappa : K_2 \xrightarrow{R^{op}} K_1 \). These components satisfy the condition\[\text{[56]}\]

\[ \kappa \cdot \tau_2, \tau_2 = \tau_1, \tau(r_2) \cdot \text{tup}_{A_1}(\hat{\varphi}_{r_2}) \cdot \hat{\tau}_{(f, g)}(\sigma_2(r_2)) \mid r_2 \in R_2 \].

\[\text{[56]}\] This is the constraint-free aspect of the database morphism condition

\[ \kappa \cdot \tau_2 = (R^{op}) \cdot \tau_1 \cdot (\hat{\varphi}^{op} \cdot \text{tup}_{A_1}) \cdot (S_2^{op} \cdot \hat{\tau}_{(f, g)}). \]

See Def\[\text{16}\] above.

\[\text{[57]}\] See Disp\[\text{7}\] in Def\[\text{4}\] of §\[\text{4.3}\].
Proposition 10. Any FOLE database morphism
\[ R_2 = (R_2, S_2, A_2, K_2, \tau_2) \xrightarrow{\langle R, \kappa, \psi, f, g \rangle} (R_1, S_1, A_1, K_1, \tau_1) = R_1 \]
in \( \text{Db} \) defines a (lax) FOLE sound logic morphism
\[ L_2 = (S_2, M_2, T_2) \xrightarrow{\langle R, \kappa, \psi, f, g \rangle} (S_1, M_1, T_1) = L_1 \]
in \( \text{Snd} \).

Proof. The source and target sound logics are defined by Prop. 8 above. The structure morphism is given by Prop. 9 above. The (abstract) specification morphism is the same as the database schema morphism
\[ T_2 = (R_2, S_2, X_2) \xrightarrow{\langle R, \psi, f \rangle} (R_1, S_1, X_1) = T_1, \]
as discussed in §5.1.

Theorem 2. There is a passage \( \text{Db}^{\text{op}} \xrightarrow{\text{snd}} \text{Snd} \).

Proof. A database \( R = (R, T, A) \) in \( \text{Db} \) is mapped to its associated sound logic \( \mathcal{L} = (S, M, T) \) by Prop. 8. A database morphism \( R_2 \xrightarrow{\langle R, \kappa, \psi, f, g \rangle} R_1 \) in \( \text{Db} \) is mapped to its associated sound logic morphism \( L_2 \xrightarrow{\langle R, \kappa, \psi, f, g \rangle} L_1 \) by Prop. 10.

Theorem 3. The contexts of databases and (lax) sound logics form a reflection
\[ \text{Db} \xrightarrow{\text{im}} \text{Snd} \]
so that these two representation of FOLE are “informationally equivalent”.

Proof. The passages in Thm. 1 and Thm. 2 form a reflection. Here, we use the image part \( \text{Tbl} \xrightarrow{\text{im}} \text{Rel} \) of the table-relation reflection. For an alternate proof, first compose with image part of the table-relation reflection and then restrict to fixed type domains.

58 The database-logic reflection \( \text{Db}(A) \xrightarrow{\text{im}(A)} \text{Log}(A) \) generalizes the table-relation reflection \( \text{Tbl}(A) \xrightarrow{\text{im}(A)} \text{Rel}(A) \). These reflections embody the notion of informational equivalence.

59 See Prop. 12 in §A.1 of the paper “The FOLE Table”.

60 The reflection \( (\text{im} \circ \text{inc}) : \text{Tbl} \Rightarrow \text{Rel} \) (”The FOLE Table”) of the context of relations into the context of tables embodies the notion of “informational equivalence”.
A Appendix

A.1 FOLE Components

Bridges. Although adjointly equivalent, no levo bridges are used throughout this paper, except for the levo tuple bridge \( f^* \circ \text{tup}_{A_2} \rightleftharpoons \text{tup}_{A_1} \) discussed in footnote 50 in §7.2.

| levo | dextro |
|------|--------|
| \( \hat{\phi} : S_2 \Rightarrow R \circ S_1 \circ f^* \) | \( \hat{\phi} : S_2 \circ \Sigma_f \Rightarrow R \circ S_1 \) |
| \( \hat{\phi} = (S_2 \circ \eta_f) \bullet (\hat{\phi} \circ f^*) \) | \( \hat{\phi} = (\hat{\phi} \circ \Sigma_f) \bullet (R \circ S_1 \circ \varepsilon_f) \) |
| \( \hat{\phi} = (\hat{\psi} \circ \text{sign}_{A_1}^\text{op})^\text{op} \) | \( \hat{\phi} = (\hat{\psi} \circ \text{sign}_{A_1}^\text{op})^\text{op} \) |
| \( \hat{\psi} : T_2 \leftarrow R^\text{op} \circ T_1 \circ \text{tbl}_{(f,g)} \) | \( \hat{\psi} : T_2 \circ \text{tbl}_{(f,g)} \leftarrow R^\text{op} \circ T_1 \) |
| \( \hat{\psi} = (\hat{\psi} \circ \text{tbl}_{(f,g)}) \bullet (T_2 \circ \varepsilon_{(f,g)}) \) | \( \hat{\psi} = (R^\text{op} \circ T_1) \eta_{(f,g)} \bullet (\hat{\psi} \circ \text{tbl}_{(f,g)}) \) |
| \( \hat{\chi}_{(f,g)} : \text{tbl}_{(f,g)} \circ \text{inc}_{A_2} \leftarrow \text{inc}_{A_1} \) | \( \hat{\chi}_{(f,g)} : \text{inc}_{A_2} \leftarrow \text{tbl}_{(f,g)} \circ \text{inc}_{A_1} \) |
| \( \hat{\chi}_{(f,g)} = (\eta_{(f,g)} \circ \text{inc}_{A_1}) \bullet (\text{tbl}_{(f,g)} \circ \hat{\chi}_{(f,g)}) \) | \( \hat{\chi}_{(f,g)} = (\text{tbl}_{(f,g)} \circ \chi_{(f,g)}) \bullet (\varepsilon_{(f,g)} \circ \text{inc}_{A_2}) \) |
| \( \hat{\tau}_{(f,g)} : (f^*)^\text{op} \circ \text{tup}_{A_2} \leftarrow \text{tup}_{A_1} \) | \( \hat{\tau}_{(f,g)} : \text{tup}_{A_2} \leftarrow \sum_{f}^\text{op} \circ \text{tup}_{A_1} \) |
| \( \hat{\tau}_{(f,g)} = (\varepsilon_{\text{op}} \circ \text{tup}_{A_2}) \bullet ((f^*)^\text{op} \circ \hat{\tau}_{(f,g)}) \) | \( \hat{\tau}_{(f,g)} = (\sum_{f}^\text{op} \circ \hat{\tau}_{(f,g)}) \bullet (\eta_{\text{op}} \circ \text{tup}_{A_2}) \) |

Table 12. FOLE Adjoint Bridges

\( \xi : T_2 \circ \text{inc}_{A_2} \leftarrow R^\text{op} \circ T_1 \circ \text{inc}_{A_1} \)

\( (R^\text{op} \circ T_1 \circ \hat{\chi}_{(f,g)}) \circ (\hat{\psi} \circ \text{inc}_{A_2}) = \xi = (\hat{\psi} \circ \text{inc}_{A_1}) \bullet (T_2 \circ \hat{\chi}_{(f,g)}) \)
Morphisms. The FOLE equivalence is explained and understood principally in terms of its various morphisms (Tbl. 13).

| Morphism Type                      | Formula                                                                 |
|------------------------------------|-------------------------------------------------------------------------|
| Schema morphism                    | $S_2 = \langle R_2, \sigma_2, X_2 \rangle \xrightarrow{({r,\phi, f})} \langle R_1, \sigma_1, X_1 \rangle = S_1$ |
| Schemed domain morphism            | $D_2 = \langle R_2, \sigma_2, A_2 \rangle \xrightarrow{({r,\phi, f, g})} \langle R_1, \sigma_1, A_1 \rangle = D_1$ |
| Lax structure morphism             | $M_2 = \langle E_2, \sigma_2, \tau_2, A_2 \rangle \xrightarrow{({r,\kappa, \phi, f, g})} \langle E_1, \sigma_1, \tau_1, A_1 \rangle = M_1$ |
| Specification morphism             | $T_2 = \langle R_2, S_2, X_2 \rangle \xrightarrow{({R,\phi, f})} \langle R_1, S_1, X_1 \rangle = T_1$ |
| Sound logic morphism               | $L_2 = \langle S_2, M_2, T_2 \rangle \xrightarrow{({R,\kappa, f, g})} \langle S_1, M_1, T_1 \rangle = L_1$ |

| Morphism Type                      | Formula                                                                 |
|------------------------------------|-------------------------------------------------------------------------|
| Database morphism 1               | $\mathcal{R}_2 = \langle R_2, T_2 \rangle \xleftarrow{(R,\xi)} \langle R_1, T_1 \rangle = \mathcal{R}_1$ |
| Database morphism 2               | $\mathcal{R}_2 = \langle R_2, T_2, A_2 \rangle \xleftarrow{(R,\psi, f, g)} \langle R_1, T_1, A_1 \rangle = \mathcal{R}_1$ |
| Database morphism 3               | $\mathcal{R}_2 = \langle R_2, S_2, A_2, K_2, \tau_2 \rangle \xleftarrow{(R,\kappa, \phi, f, g)} \langle R_1, S_1, A_1, K_1, \tau_1 \rangle = \mathcal{R}_1$ |

1 full, 2 fixed type domain, 3 with projections

Table 13. FOLE Morphisms
A.2 Classifications and Infomorphisms

The concept of a “classification” comes from the theory of Information Flow: see the book Information Flow: The Logic of Distributed Systems by Barwise and Seligman [1]. A classification is also important in the theory of Formal Context Analysis, where it is called a “formal context”: see the book Formal Concept Analysis: Mathematical Foundations by Ganter and Wille [4].

Classification. A classification $A = \langle X, Y, \models_A \rangle$ consists of: a set $Y = \text{inst}(A)$ of objects to be classified, called tokens or instances; a set $X = \text{typ}(A)$ of objects that classify the instances, called types; and a binary relation $\models_A \subseteq Y \times X$ between instances and types. The extent of any type $x \in X$ is the subset of instances classified by $x$: $\text{ext}_A(x) = \{ y \in Y \mid y \models_A x \}$. Dually, the intent of any instance $y \in Y$ is the subset of types that classify $y$: $\text{int}_A(y) = \{ x \in X \mid y \models_A x \}$. Two types $x, x' \in X$ are coextensive in $A$ when $\text{ext}_A(x) = \text{ext}_A(x')$. Two instances $y, y' \in Y$ are indistinguishable or indistinguishable in $A$ when $\text{int}_A(y) = \text{int}_A(y')$.

A classification $A$ is separated or intensional when there are no two indistinguishable instances: $y \neq y'$ implies $\text{int}_A(y) \neq \text{int}_A(y')$. A classification $A$ is extensional when all coextensive types are identical: $\text{ext}_A(x) = \text{ext}_A(x')$ implies $x = x'$; equivalently, $x \neq x'$ implies $\text{ext}_A(x) \neq \text{ext}_A(x')$. More strongly, a classification $A$ is disjoint when distinct types have disjoint extents: $x \neq x'$ implies $\text{ext}_A(x) \cap \text{ext}_A(x') = \emptyset$. Even more strongly, a classification $A$ is partitioned when it is disjoint and all instances are classified: $\text{int}_A(y) \neq \emptyset$ equivalently $y \in \text{ext}_A(x)$ some $x \in X$ for all $y \in Y$. A classification $A$ is pseudo-partitioned when it is disjoint and all instances are classified except for one special instance $\cdot \in Y$: $\text{int}_A(\cdot) = \emptyset$.

Infomorphism. An infomorphism $A_2 = \langle X_2, Y_2, \models_{A_2} \rangle \xrightarrow{(f,g)} \langle X_1, Y_1, \models_{A_1} \rangle = A_1$ consists of a type map $X_2 \xrightarrow{f} X_1$ and an instance map $Y_2 \xleftarrow{g} Y_1$, which satisfy the following:

$$g(y_1) \models_{A_2} x_2 \iff y_1 \models_{A_1} x_1 = f(x_2)$$
$$g(y_1) \in \text{ext}_{A_2}(x_2) \iff y_1 \in \text{ext}_{A_1}(f(x_2))$$
$$g^{-1}(\text{ext}_{A_2}(x_2)) = \text{ext}_{A_1}(f(x_2))$$

![Fig. 15. Infomorphism](image-url)
If two distinct source types \( x_1, x_2 \in X \), \( x_1 \neq x_2 \), are mapped by the type map to the same target type \( \phi \mapsto f(x_2) = f(x_2') = x_1 \in X_1 \), then any target instance in the extent \( y_1 \in \text{ext}_{\mathcal{A}_1}(x_1) \) is mapped by the instance map to the extent intersection \( g(y_1) \in \text{ext}_{\mathcal{A}_2}(x_2) \cap \text{ext}_{\mathcal{A}_2}(x_2') \). Hence, if extent sets are disjoint in the source classification, then the type map must be injective. Note however that there is restricted map \( \text{ext}_{\mathcal{A}_2}(x_2) \times \text{ext}_{\mathcal{A}_2}(x_2') \xrightarrow{g} \text{ext}_{\mathcal{A}_1}(x_1) \). Let \( \mathsf{CIs} \) denote the context of classifications and infomorphisms.

**Type Domains.** In the FOLE theory of data-types \cite{9}, a classification \( \mathcal{A} = \langle X, Y, \models_{\mathcal{A}} \rangle \) is known as a type domain. A type domain \( \prod \) is an sort-indexed collection of data types from which a table’s tuples are chosen. It consists of a set of sorts (data types) \( X \), a set of data values (instances) \( Y \), and a binary (classification) relation \( \models_{\mathcal{A}} \) between data values and sorts. The extent of any sort (data type) \( x \in X \) is the subset \( \text{ext}_{\mathcal{A}}(x) = A_x = \{ y \in Y \mid y \models_{\mathcal{A}} x \} \). Hence, a type domain is equivalent to a sort-indexed collection of subsets of data values \( X \xrightarrow{\mathcal{A}} \phi Y : x \mapsto \text{ext}_{\mathcal{A}}(x) = A_x \). The list classification \( \mathsf{List}(\mathcal{A}) = \langle \mathsf{List}(X), \mathsf{List}(Y), \models_{\mathsf{List}(\mathcal{A})} \rangle \) has \( X \)-signatures as types and \( Y \)-tuples as instances, with classification by common arity and universal \( \mathcal{A} \)-classification: a \( Y \)-tuple \( \langle J, t \rangle \) is classified by an \( X \)-signature \( \langle I, s \rangle \) when \( J = I \) and \( f_k \models_{\mathcal{A}} s_k \) for all \( k \in J = I \). In the FOLE theory of data-types, an infomorphism \( \mathcal{A} \xrightarrow{\langle f, g \rangle} \overline{\mathcal{A}} \) is known as a type domain morphism, and consists of a sort function \( X' \xrightarrow{f} X \) and a data value function \( Y' \xrightarrow{g} Y \) that satisfy the condition \( g(y) \models_{\mathcal{A}} x' \iff y \models_{\mathcal{A}} f(x') \) for any source sort \( x' \in X' \) and target data value \( y \in Y \).

**Lax Classification.** From an extensional point-of-view, a classification \( \mathcal{A} = \langle X, Y, \models_{\mathcal{A}} \rangle \) is a set-valued function \( X \xrightarrow{\text{ext}_{\mathcal{A}}} \phi Y \); that is, an indexed collection of (sub)sets
\[
\{ \text{ext}_{\mathcal{A}}(x) = Y(x) \mid x \in X \}. \]In a lax classification we omit the global set of data values \( Y \). Hence, a (lax) classification is just the pair \( \mathcal{A} = \langle X, Y \rangle \) consisting of a set \( X = \mathsf{typ}(\mathcal{A}) \) of objects called types that classify the instances.

\begin{itemize}
  \item Are you producing the two types by doing this? Remember how the sum and product of classifications is defined.
  \item Some examples of data-types useful in databases are as follows. The real numbers might use sort symbol \( \mathbb{R} \) with extent \( \{-\infty, \cdots, 0, \cdots, \infty\} \). The alphabet might use sort symbol \( \mathbb{A} \) with extent \( \{a, b, c, \cdots, x, y, z\} \). Words, as a data-type, would be lists of alphabetic symbols with sort symbol \( \mathbb{A}^\ast \) with extent \( \{a, b, c, \cdots, x, y, z\}^\ast \) being all strings of alphabetic symbols. The periodic table of elements might use sort symbol \( \mathbb{E} \) with extent \( \{H, He, Li, He, \cdots, Hs,Mt\} \). Of course, chemical elements can also be regarded as entity types in a database with various properties such as name, symbol, atomic number, atomic mass, density, melting point, boiling point, etc.
  \item In particular, when \( I = 1 \) is a singleton, an \( X \)-signature \( \langle 1, s \rangle \) is the same as a sort \( s(\cdot) = x \in X \), a \( Y \)-tuple \( \langle 1, t \rangle \) is the same as a data value \( t(\cdot) = y \in Y \), and \( \mathsf{tup}(1, s, \mathcal{A}) = \mathsf{tup}(1, s, \mathcal{A}) = \text{ext}_{\mathsf{List}(\mathcal{A})}(1, s) = A_x \).
  \item We can always define a global set of instances; for example, \( Y = \bigcup_{x \in X} Y(x) \), \( Y = \prod_{x \in X} Y(x) \), etc.
\end{itemize}
and a collection of indexed subsets \( \{ Y(x) \mid x \in X \} \), where \( Y(x) = \text{ext}_A(x) \) is the set of instances (tokens) classified by the type \( x \in X \). More briefly, a \( \text{(lax)} \) classification is a \( \text{Set} \)-valued diagram \( X \xrightarrow{Y} \text{Set} \).

Lax Infomorphism.Tbl. 14 motivates the definition of a lax infomorphism. In a lax infomorphism \( \mathcal{A}_2 = \langle X_2, Y_2 \rangle \xleftarrow{\gamma} \langle X_1, Y_1 \rangle = \mathcal{A}_1 \) the instance function \( Y_2 \xleftarrow{g} Y_1 \) is replaced by the collection of instance functions \( \{ Y_2(x_2) \xleftarrow{\gamma_2} Y_1(f(x_2)) \mid x_2 \in X_2 \} \). Hence, an infomorphism is a \( \text{Set} \)-valued instance bridge \( Y_2 \xleftarrow{f \circ Y_1} \). Let \( \text{Cls} \) denote the context of lax classifications and lax infomorphisms. There is a passage \( \text{Cls} \xrightarrow{\text{lax infomorphism}} \text{Cls} \).

| infomorphism |
|------------------|
| \( \mathcal{A}_2 = \langle X_2, Y_2 \rangle \xleftarrow{\gamma} \langle X_1, Y_1 \rangle = \mathcal{A}_1 \) |
| \( g(y_1) \models_{\mathcal{A}_2} x_2 \iff y_1 \models_{\mathcal{A}_1} f(x_2) \) |
| \( g(y_1) \in \text{ext}_{\mathcal{A}_2}(x_2) \iff y_1 \in \text{ext}_{\mathcal{A}_1}(f(x_2)) \) |
| \( g^{-1}(\text{ext}_{\mathcal{A}_2}(x_2)) = \text{ext}_{\mathcal{A}_1}(f(x_2)) \)

| implies |
|---------------------------------------------------------------|
| \( g^{-1}(\text{ext}_{\mathcal{A}_2}(x_2)) \supseteq \text{ext}_{\mathcal{A}_1}(f(x_2)) \) |
| \( g(x_1) \in \text{ext}_{\mathcal{A}_2}(x_2) \iff x_1 \in \text{ext}_{\mathcal{A}_1}(f(x_2)) \) |
| \( g(y_1) \models_{\mathcal{A}_2} x_2 \iff y_1 \models_{\mathcal{A}_1} f(x_2) \) |
| \( A_2 = \langle X_2, Y_2 \rangle \xleftarrow{(f, \gamma)} \langle X_1, Y_1 \rangle = \mathcal{A}_1 \) |

lax infomorphism

| Table 14. Lax Infomorphisms |

\[
\begin{array}{c}
X' & \xrightarrow{f} & X \\
\nu Y' & \xleftarrow{g^{-1}} & \nu Y \\
\end{array}
\]

\[
\begin{array}{c}
X' & \xrightarrow{f} & X \\
\psi & \xleftarrow{\psi} & \text{Set} \\
\end{array}
\]

strict \hspace{1cm} lax

Fig. 16. Lax Infomorphism
References

1. Barwise, J., and Seligman, J.: *Information Flow: The Logic of Distributed Systems*. Cambridge University Press, Cambridge (1997).
2. Chen, P.: “The Entity-Relationship Model - Toward a Unified View of Data”. ACM Trans. on Database Sys., 1 (1): pp. 9-36. doi:10.1145/320434.320440.
3. Codd, E.F.: *The relational model for database management : version 2*. Addison Wesley, Boston (1990).
4. Ganter, B., and Wille, R.: *Formal Concept Analysis: Mathematical Foundations*. Springer, New York (1999).
5. Goguen, J.: *A categorical manifesto*. Mathematical Structures in Computer Science 1, 49–67 (1991).
6. Gruber, T. “Ontology”. In: Ling Liu and M. Tamer ¨Ozsu (eds.) *The Encyclopedia of Database Systems*. Springer-Verlag (2009). Available online: tomgruber.org/writing/ontology-definition-2007.htm.
7. Kent, R.E.: “Database Semantics”. (2011). Available online: https://arxiv.org/abs/1209.3054
8. Kent, R.E.: “The First-order Logical Environment”. In: Pfeiffer, H.D., Ignatov, D.I., Poelmans, J., and Nagarjuna G. (eds.) *Conceptual Structures in Research and Education*, LNCS vol. 7735, pp. 210-230. Springer, Heidelberg (2013). Available online: https://arxiv.org/abs/1305.5240
9. Kent, R.E.: “The ERA of FOLE: Foundation” (2015). Submitted for publication. Available online: https://arxiv.org/abs/1512.07430
10. Kent, R.E.: “The ERA of FOLE: Superstructure” (2016). Available online: https://arxiv.org/abs/1602.04268
11. Kent, R.E.: “The FOLE Table”. Submitted for publication. Available online: https://arxiv.org/abs/1810.12100
12. Kent, R.E.: “The FOLE Database”. Available online: https://arxiv.org/abs/1810.08236
13. Kent, R.E.: “Relational Operations in FOLE”. Submitted for publication. Available online: https://arxiv.org/abs/1809.05997
14. Kent, R.E.: “Semantic Integration in the IFF”. Presented at the Semantic Integration Workshop of the 2nd International Semantic Web Conference (ISWC2003), Sanibel Island, Florida, October 20, 2003. Available online: https://arxiv.org/abs/1109.0032
15. Kent, R.E.: “Semantic Integration in the Information Flow Framework”. Dagstuhl Seminar Proceedings 04391, Semantic Interoperability and Integration, Schloss Dagstuhl, Leibniz-Zentrum für Informatik GmbH, 2005. Available online: https://arxiv.org/abs/1810.08236
16. *The Information Flow Framework (IFF)*. Start document available from the IEEE P1600.1 Standard Upper Ontology Working Group (SUO WG) Home Page. http://ontolog.cim3.net/file/resource/historic-archives/IEEE-SUO-WG/