Report
Levelling Data Representations

Salvatore Florio, Carlo Nicolai, and Chris Partridge

4 June 2025

Contents

1 Introduction

This is an initial report on the work of the commercialisation project Levelling Data Representations (AHRC Grant AH/Z506515/1). The project successfully provided a logical foundation for the data engineering unification process developed in bCLEARer by BORO Solutions. The paper provides a logical picture of the micro-transformations that compose the unification process. This has enabled a much clearer picture of the transformation space. From this, one can build a vision of possible improvements to the bCLEARer process. We plan to exploit this at the next stage.

1.1 Schemata-as-patterns

A common style for data architectures uses syntax to capture schemata as repeatable patterns in a given data language. This typically involves a syntax that segregates the representational domain into schemata objects and schemata-bound objects, where a hard-coded schemata framework is used to encode the repeatable patterns, and to impose them upon schemata-bound objects through an instantiation relation.

At its simplest, the use of schemata involves a two-level structure. Two good examples are the relational paradigm (implemented in SQL), widely used in computing, and table-based structures. In the relational paradigm, the schemata are explicitly called (data) schema. Here the schema syntax of database tables and column headings is used to capture the repeatable patterns, and the harmonised data (instance) syntax of rows and cells enable the schema-bound data objects to have the schema level structure imposed on them.

The schemata pattern can recur. In other words, when the schemata objects themselves have repeatable patterns, this can be encoded in a higher level schemata structure where the schemata objects play a schemata-bound role. This results in a domain that is divided into a (linear) chain of ranked layers. A good example of this is Object Management Group's Meta-Object Facility (MOF), with its layers, typically four: M0, M1, M2, and M3. This has schema structures linking the adjacent pairs of connected layers (see table below), where each pair has its instantiation relation to bind the schema to its objects.

schema structure	schema objects layer	schema-bound objects layer
M0-M1 schema structure	M1	M0
M1-M2 schema structure	M2	M1
M2-M3 schema structure	M3	M2

The relational paradigm (a two-level schema-based architecture) emerged in the 1970s and became commonplace in the 1980s. While it was broadly successful, there were a number of areas where it was found to be an inappropriate approach. An ubiquitous problem was integrating heterogeneous datasets: datasets with no obvious common schema structure, with new datasets possibly introducing unexpected schema structures. In this context, it made no sense to invest in an architecture that expected the schema structure to be finalised upfront.

1.2 bClearer

bCLEARer is an ontology-driven data platform framework that uses pipelines to transform, integrate, and exchange information from heterogeneous datasets. Over the past three decades, bCLEARer has evolved a micro-transformation architectural style for its pipelines that helps make the fine-grained transformation structure inspectable.

This visibility has revealed recurring motifs, including a family of refactoring micro-transformations that build into a macro-transformation that we call unification (the unification processed referred to above). Unification has been found to help streamlining integration as well as supporting the sophisticated general data structures needed for foundational ontologies.

1.3 Logical methods and formalization

One can see a close analogy between standard predicate logic (including higher-order logic) and the schema-separation architectural style. In both cases, the domains are separated into a linear hierarchy of layers, which are used to capture repeatable patterns. As described earlier, the syntactic predication/instantiation relation is used as a link between layers.

Some elements of this data unification architectural style have been developed within formal logic, in particular in the context of techniques employed to transform multiple-domain (many-sorted) structures modelling higher-order logic into single-domain structures suitable for first-order logic. A clear example is *Quine's unification of universes* (1956). This suggested to us that we may be able to represent in predicate logic the bCLEARer micro-transformations for migrating from schema-based datasets to unified datasets.

This is what has been achieved in the project. Because the instantiation/predication relation at the heart of formal logic is both simpler and better studied than its data-engineering counterparts, examining its unification micro-transformations has provided a rigorous foundation to a mostly pragmatic computerised process. The formalization opens the way to (i) review the existing pragmatic process embedded in bCLEARer and (ii) uncover new opportunities to improve such processes. The formalization of the transformations are described in the body of the report.

1.4 Looking ahead

At a first stage, the project has developed a good picture of the logical counterparts of the data engineering micro-transformations. These are described in the body of the report. From this, a vision of the transformation space is beginning to emerge, one that exposes how schemata based representations can be migrated into a more tractable and expressive graph-based representation. Also, the nature of the increased expressivity afforded by the unification process is becoming clearer. So, even at this early stage, we can see multiple opportunities for commercial exploitation.

2 Operations

2.1 Simple Predicate Pushdown

The operation replaces syntactic predication with an explicit instantiation predicate. Examples:

$$
\begin{aligned}
P(a) &\mapsto \eta_{01}(a, p) \\
R(a, b) &\mapsto \eta_{001}(a, b, r) \\
Q(P) &\mapsto \eta_{12}(q, p) \\
\eta_{01}(a, p) &\mapsto \eta_{(01(01))}(a, p, e_{(01)})
\end{aligned}
$$

The indices of the η-predicates track the types relevant to the predication replaced. Successive applications of the operation retain this "historical" information in the form shown by the last line.

Formal Predicates. 'Structural' predicates, such as instantiation and pairing, are considered formal. The operation of Simple Predicate Pushdown introduces new formal predicates, namely an instantiation predicate for each form of predication in the original language.

Contentful Predicates. The operation replaces contentful predicates, which correspond to data schemes, with individual constants.

Domain Expansion. The domain expands with new objects, the denotations of the new individual constants obtained from contentful predicates as well as from instantiation relations obtained via iterations of simple pushdown.

Iterations. Simple predicate pushdown can be iterated. Starting with the second iteration, only 'structural' predicates and constants (constants for instantiation relations) are added: in the boxed examples, the relation η_{01} is replaced by the constant $e_{(01)}$.

Arity. The predicates in the new language increase by one the arities of those in the previous language.

Order. This operation translates the vocabulary of the previous language, regardless of its order, into first-order vocabulary.

2.2 Levels

The operation adds 'formal'[a] predicates expressing syntactic types of the previous language.[b]

$$a, b, c \ldots \mapsto L_0(a), L_0(b), L_0(c)\ldots$$
$$P, \mathfrak{P}, X, \mathfrak{X}\ldots \mapsto L_1(p), L_2(\mathfrak{p}), L_1(x), L_2(\mathfrak{x})\ldots$$
$$\eta(\langle a, p\rangle) \mapsto L_1^*(e), L_0(\langle a, p\rangle)$$

[a] This could even be understood in terms of permutation invariance.
[b] In the example, fraktur alphabet is used for third-order predicates and constants.

Formal Predicates. The operation adds new structural/formal predicates. In the case of levels, there's a case to be made that levels can be seen to be 'logical' in the sense of permutation invariance.

Contentful Predicates. Assuming that the operation only adds formal levels, the operation of levels does not add new contentful predicates.

Domain Expansion. By itself, the operation does not add new objects in the domain.

Iteration. The procedure is not per se iterated, but takes place at every iteration. New level predicates are added, and the old ones are *not* translated.

Arity. The new level predicates are unary, even in iterations. The old level predicates' arity is preserved.

Order. The operation only adds new first-order predicates.

2.3 Monadisation

The operation employs pairing to turn (first-order) predicates with arity > 1 into unary predicates.

$$R(a, b) \mapsto R^*(\langle a, b\rangle)$$
$$\eta(\langle a, p\rangle, e) \mapsto \eta^*(\langle\langle a, p\rangle, e\rangle)$$

Formal Predicates. The operation rests on (but not formally introduces per se) formal predicates for pairing and associated projections.[1]

Contentful Predicates. Assuming that the operation only adds formal pairing predicate and projections, the operation of monadisation does not add new contentful predicates.

[1] The notion of formality is semi-formal, and cannot be explained in terms of permutation invariance as in the case of formal predicates levels.

Domain Expansion. The operation demands adding new pairs in the domain whose components are the new domain entities resulting from predicate pushdown.

Iteration. The procedure is iterated, and results in a domain expansion at each iteration, but no new formal predicates for pairing and projection are added once they are introduced (see pairing).

Arity. The operation reduces predicates' arity, 'turning' all of predicates into unary ones.

Order. Monadisation always operates at the first-order level. The 'tuples' involved in the operation are considered to be individuals in the relevant domain.

2.4 Pairing

The operation introduces ordered pairs of objects recognized in the previous language. This relies on a pairing predicate (Pair) and two definable projection predicates (Left and Right). Example:

$$a, b \ldots \mapsto \exists x \textsc{Pair}(x, a, b)$$

The operation is accompanied by axioms and definitions characterizing the order pairs, such as:

$$(\textsc{Pair}(x_1, y_1, z_1) \wedge \textsc{Pair}(x_2, y_2, z_2)) \rightarrow (y_1 = y_2 \wedge z_1 = z_2 \leftrightarrow x_1 = x_2) \qquad \text{(Pair-Identity)}$$

$$\exists z \textsc{Pair}(x, y, z) \leftrightarrow : \textsc{Left}(x, y) \qquad \text{(Pair-Left)}$$

$$\exists y \textsc{Pair}(x, y, z) \leftrightarrow : \textsc{Right}(x, z) \qquad \text{(Pair-Right)}$$

Formal Predicates. The operation introduces a formal predicate, namely Pair, governed by appropriate axioms. Successive applications do not introduce additional pairing predicates.

Contentful Predicates. The operation does not introduce new contentful predicates.

Domain Expansion. The domain expands with ordered pairs of objects recognized in the previous language. The size of the expansion can be controlled by axioms. A minimalistic approach assumes only pairs of specific objects previously recognized. A more liberal approach assumes all pairs that can be formed on the basis of a given pool of objects, including pairs of pairs, pairs of pairs of pairs, and so on.

Iterations. Pairing can be iterated. Iterations add new pairs, without relying on new predicates.

Arity. The operation introduces a ternary predicate and two definable binary predicates. It leaves the arity of other predicates unchanged.

Order. Pairing does not affect the order of the language.

2.5 Monadicized Predicate Pushdown

The operation replaces syntactic predication with an explicit, unique binary instantiation predicate. A pairing operation is used to reduce arities > 1 to a binary form. Examples:

$$\begin{aligned} P(a) &\mapsto \eta(a, p) \\ R(a, b) &\mapsto \eta(\langle a, b\rangle, r) \\ Q(P) &\mapsto \eta(q, p) \\ S(P, a) &\mapsto \eta(\langle p, a\rangle, s) \\ \eta(a, p) &\mapsto \eta'(\langle a, p\rangle, e) \end{aligned}$$

Successive applications of the operation introduce new ηs (η', η'',…), and the previous ηs appear as entities in the domain (e, e',…).

Formal Predicates. Each application of Monadicized Predicate Pushdown introduces one new formal instantiation predicate, which might be accompanied by other structural predicates, such as pairing.

Contentful Predicates. The operation replaces contentful predicates, which correspond to data schemes, with individual constants.

Domain Expansion. The domain expands with new objects, the denotations of the new individual constants obtained from contentful predicates as well as from instantiation relations obtained via iterated applications. The domain also expands with ordered pairs corresponding to arguments of relations in the previous language.

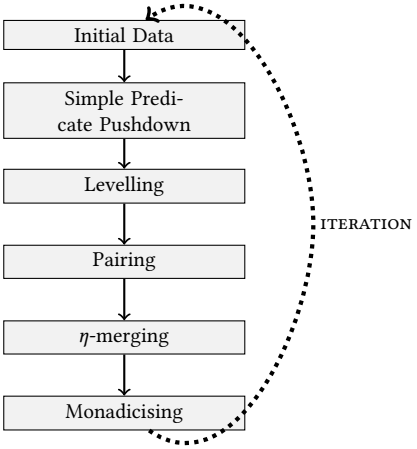
Iterations. The operation can be iterated, as also shown in the last example. Starting with the second iteration, only 'structural' predicates and constants (constants for instantiation relations) are added. That is, we successively introduce new ηs (η', η'',…), and the previous ηs appear as entities in the domain (e, e',…) .

Arity. Each application of the operation yields a language with a single binary predicate for instantiation. So, starting after the first applicaiton, the arity of predicates in the language remains fixed.

Order. This operation translates the vocabulary of the previous language, regardless of its order, into first-order vocabulary. So the order is fixed after the first application.

3 The Unification Algorithm

We apply the operations introduced to realize our unification 'algorithm': the procedure is intended to start with (a formalization of) raw data and proceeds with applications of the operations above arranged in a specific sequence; once the first cycle ends, the procedure can be applied again in the first iteration cycle, and so on until a *stability point* is reached.



Remark 1. It's worth remarking that we are presenting a natural sequence starting with initial data that, in the intended application, is provided in some higher-order setting. The operations introduced below may be applied individually – by allowing vacuous application when the basic notions required by the operation are not available.[2] It is also possible that data presents itself already at some stage of the natural sequence, in which case one can simply continue with the subsequent operations in the sequence.

3.1 Initial Data

We start with an arbitrary *higher-order, relational language* $\mathcal{L}$ employed to formalize a (possibly empty) set of nonlogical assumptions. Suitable logical information (higher-order logic at the appropriate level) is assumed.

Example. $\mathcal{L}$ is a second-order language. We add to second-order logical axioms and rules the non-logical principles:

The Initial Theory $\mathcal{I}$

$$P(a) \qquad (P)$$
$$R(a,b) \qquad (R)$$
$$\mathscr{Q}(P,b). \qquad (\mathscr{Q})$$

[2] E.g. one can vacuously apply Predicate Pushdown if there are no predicates.

3.2 Simple Predicate Pushdown

Simple predicate pushdown can be formalized as a translation τ_1 of $\mathcal{L}$ into a first-order language in which syntactic predications are replaced by *suitable instantiations relations respecting arity and order.* [...]

Example. Continuing with the example above, we first assume a function δ whose domain is the set of variables and constants of the language considered, which behaves as follows:

$$\begin{array}{lll} \delta(x_i) := x_{2i+1} & \delta(X_i) := x_{2i} & \\ \delta(a) := a & \delta(b) := b & \\ \delta(P) := p & \delta(R) := r & \delta(\mathscr{Q}) := q \end{array}$$

We can then employ δ to define the full translation of the higher-order language. I omit reference to δ when notationally convenient.

The translation τ_1

$$\begin{aligned} \tau_1(s = t) &:\leftrightarrow \delta(s) = \delta(t) \\ \tau_1(X_i\vec{s}) &:\leftrightarrow \eta_{\vec{01}}(\delta(s), \delta(\vec{X}_i)) \\ \tau_1(P(s)) &:\leftrightarrow \eta_{01}(\delta(s), p) \\ \tau_1(R(s,t)) &:\leftrightarrow \eta_{001}(\delta(s), \delta(t), r) \\ \tau_1(\mathscr{Q}(T,s) &:\leftrightarrow \eta_{102}(\delta(T), \delta(s), q) \\ \tau_1(\neg\phi) &:\leftrightarrow \neg(\tau_1(\phi)) \\ \tau_1(\phi \wedge \psi) &:\leftrightarrow \tau_1(\phi) \wedge \tau_1(\psi) \\ \tau_1(\forall x_i \phi) &:\leftrightarrow \forall x_{2i+1} \tau_1(\phi) \\ \tau_1(\forall X_i \phi) &:\leftrightarrow \forall x_{2i} \tau_1(\phi) \end{aligned}$$

Finally, the translation τ_1 behaves like a *relative interpretation* once the translations of the non-logical axioms of $\mathcal{I}$ are added to first-order logic with identity. Specifically:

The theory $\mathcal{I}^1$

$\mathcal{I}^1$ extends FOL with identity with the following:

$$\begin{aligned} &\eta_{01}(a, p) && (P^1) \\ &\eta_{001}(a, b, r) && (R^1) \\ &\eta_{102}(p, b, q) && (\mathscr{Q}^1) \end{aligned}$$

Proposition

$\mathcal{I} \vdash \phi(\vec{X}, \vec{x}) \;\Rightarrow\; \mathcal{I}^1 \vdash \tau_1(\phi)(\delta(\vec{X}), \delta(\vec{x}))$.

3.3 Levelling

The translation τ_1 formalizing Simple predicate pushdown can be further relativized to suitable *levels* corresponding to the ontological/ideological levels of the initial data. This can be formalized as a new translation from the initial language $\mathcal{L}$ to an expansion of the language obtained via Simple Predicate Pushdown with such levels.

Example. Levelling takes the language of the theory $\mathcal{I}$ and adds predicates for levels L_0, L_1, L_3 corresponding to the syntactic categories of $\mathcal{L}$, as well as all new constants required by Simple Predicate Pushdown; suitable 'axioms' are added to assign the right level to constants. This amounts to a translation τ_2 from the language of $\mathcal{I}$ into this augmented language which is just like τ_1 but suitably relativizes quantifiers.

The Levelling Translation τ_2

τ_2 behaves just like τ_1 on atomic formulae and propositional connectives (including δ). For quantifiers:

$$\tau_2(\forall x_i \phi) :\leftrightarrow \forall x_{2i+1}(L_0(x_{2i+1}) \to \tau_2(\phi))$$
$$\tau_2(\forall X_i \phi) :\leftrightarrow \forall x_{2i}(L_1(x_{2i}) \to \tau_2(\phi))$$

τ_2 only takes care of quantifiers (and variables). Levelling requires suitable axioms assigning *nonlogical constants* the right level. Such axioms form the theory $\mathcal{I}^2$.

The Theory $\mathcal{I}^2$

$\mathcal{I}^2$ features all axioms of $\mathcal{I}^1$ together with

$L_0(a) \wedge L_0(b)$ (L_0)

$L_1(p) \wedge L_1(r)$ (L_1)

$L_2(q)$ (L_2)

$\big(L_0(x) \vee L_1(x) \vee L_2(x)\big) \wedge \neg(L_0(x) \wedge L_1(x)) \wedge \neg(L_1(x) \wedge L_2(x)) \wedge \neg(L_0(x) \wedge L_2(x))$ (LED)

Finally, the translation τ_2 indeed preserves provability in I.

Proposition

$\mathcal{I} \vdash \phi(\vec{X}, \vec{x}) \;\Rightarrow\; \mathcal{I}^2 \vdash L_1(\delta(\vec{X})) \wedge L_0(\delta(\vec{x})) \rightarrow \tau_2(\phi)(\delta(\vec{X}), \delta(\vec{x}))$.

3.4 Pairing

The operation of Pairing can be formalized as the extension of the theory resulting from the previous operations with the axioms for Pair, Left, Right described in section 2.4. Since the operation is intended to apply to a theory to which Levelling has been applied, the new axioms raise questions concerning the levels to which pairs belong. We stipulate that pairs belong to a new level disjoint from all others. n-tuples are generated by iterated pairing.

Example. The formalization of Pairing requires only a new predicate Pair. There's no need for a predicate for the level to which pairs belong, as this can be defined. The theory does not specify the existence of pairs of any objects in the domain, but specifies which pairs exist. We require that pairs/tuples corresponding to atomic facts in $\mathcal{I}$ exist.

The Theory $\mathcal{I}^3$

The language of $\mathcal{I}^3$ expands the language of $\mathcal{I}_2$ with the additional ternary predicate Pair. Its axioms are the axioms of $\mathcal{I}^2$, (Pair-Left), (Pair-Right) from section 2.4, and the following:[a]

$$(\textsc{Pair}(x, y_1, z_1) \wedge \textsc{Pair}(x, y_2, z_2)) \rightarrow (y_1 = y_2 \wedge z_1 = z_2) \wedge (\neg L_0(x) \wedge L_1(x) \wedge \neg L_2(x)) \qquad \text{(Pair-Identity-L)}$$

$$\exists x\, \textsc{Pair}(x, a, p) \qquad \text{(Pair-E1)}$$

$$\exists x\, \textsc{Pair}(x, a, b) \qquad \text{(Pair-E2)}$$

$$\exists x\, \textsc{Pair}(x, a, \langle a, b\rangle, r) \qquad \text{(Pair-E3)}$$

$$\exists x\, \textsc{Pair}(x, p, b) \qquad \text{(Pair-E4)}$$

$$\exists x\, \textsc{Pair}(x, \langle p, b\rangle, q) \qquad \text{(Pair-E5)}$$

$\vdots$

[a] we often employ functional notation for pairs given their uniqueness.

The vertical dots stand for additional pairing existence axioms that one may want to add to the theory. The last conjunct of (Pair-Identity-L) provides a contextual definition of the 'pair level'

$$L_P(x) :\leftrightarrow \neg L_0(x) \wedge L_1(x) \wedge \neg L_2(x).$$

3.5 η-Merging

The operation of η-merging operates on the theory in which different instantiation predicates are present – corresponding to the different arities and order of the initial data or iterations thereof –, and employs the pairing resources just given to merge the different instantiation relations into one. The formalization of η-merging amounts to a translation which replaces the multiple ηs with a single one.

Example. In the example, the translation applies to the language of $\mathcal{I}^3$ and replaces each of $\eta_{01}, \eta_{001}, \eta_{102}$ with a single η:

The Translation τ_4

τ_4 leaves complex formulae unchanged, and operates at the level of atomic formulae (leaving identity unchanged) as follows:

$$\begin{aligned}
\tau(\eta_{01}(x_i, x_j)) &:\leftrightarrow \eta(x_i, x_j) \\
\tau(\eta_{001}(x_i, x_j, x_k)) &:\leftrightarrow \exists w(\textsc{Pair}(w, x_i, x_j) \wedge \eta(w, x_k)) \\
\tau(\eta_{102}(x_i, x_j, x_k)) &:\leftrightarrow \exists w(\textsc{Pair}(w, x_i, x_j) \wedge \eta(w, x_k))
\end{aligned}$$

We can then formulate a theory $\mathcal{I}^4$ which, essentially, is the result of reformulating the axioms of $\mathcal{I}^3$ in terms of the new single η.

$\mathcal{I}^4$

The language of $\mathcal{I}^4$ is obtained from the language of $\mathcal{I}^3$ by replacing the three instantiation relations η_{01}, η_{001}, and η_{102} with a single instantiation relation η. Its axioms are the axioms of $\mathcal{I}^3$, except that P^1, R^1, and $\mathscr{Q}^1$ from $\mathcal{I}^1$ – which have been carried over to $\mathcal{I}^3$. Given the existence of the relevant pairs involved in R^1, and $\mathscr{Q}^1$, we can formulate the new axioms as:

$$\eta(a, p) \tag{1}$$
$$\eta(\langle a, b\rangle, r) \tag{2}$$
$$\eta(\langle p, b\rangle, q). \tag{3}$$

Again, one can show that the translation τ_4 preserves provability.

Proposition

$\mathcal{I}^3 \vdash \phi(\vec{x}) \;\Rightarrow\; \mathcal{I}^4 \vdash \phi^{\tau_4}(\vec{x})$.

3.6 Monadicisation

The information contained in the theory obtained so far via successive applications of the relevant operations – $\mathcal{I}^4$ in the example – can be roughly divided into two camps: *formal* content information concerns identity, levels, pairs; *contentual* information concerns instantiation. The final operation of monadicisation operates on contentful information and replaces the binary η with a *unary* one. Again, the formalization of the process can be carried out in terms of a provability-preserving translation.

Example. We first introduce a simple translation replacing the binary η in the language of $\mathcal{I}^4$ with a *unary* $\eta^\top$.

τ^5

τ^5 leaves all vocabulary of the language of $\mathcal{I}^4$ unchanged, except for the clause

$$\tau^5(\eta(x,y)) :\leftrightarrow \exists w(\textsc{Pair}(w,x,y) \wedge \eta^\top(w)).$$

To ensure that τ_5 indeed preserves provability, we need to stipulate the existence of the required pairs as well as the truth of the required instantiation claims.

$\mathcal{I}^5$

The axioms of $\mathcal{I}^5$ are the axioms of $\mathcal{I}^4$ including structural content only, with the addition of (if not already present in $\mathcal{I}^4$):

$$\exists w(\textsc{Pair}(w,\langle a,b\rangle,r) \tag{4}$$
$$\exists w(\textsc{Pair}(w,\langle p,b\rangle,q). \tag{5}$$

Moreover, the axioms (1)-(3) of $\mathcal{I}^4$ are replaced with[a]

$$\eta^\top(\langle a,p\rangle) \tag{6}$$
$$\eta^\top(\langle\langle a,b\rangle,r\rangle) \tag{7}$$
$$\eta^\top(\langle\langle p,b\rangle,q\rangle) \tag{8}$$

[a] Again the existence and uniqueness of the relevant pairs licenses a more suggestive notation.

Finally, $\mathcal{I}^5$ is able to interpret $\mathcal{I}^4$:

Proposition

$\mathcal{I}^4 \vdash \phi(\vec{x}) \;\Rightarrow\; \mathcal{I}^5 \vdash \phi^{\tau_5}(\vec{x})$.

3.7 Iteration

The natural sequence provided in this section is of course iterable. With reference to the diagram on page 7, the "Initial Data" can even correspond to a theory such as $\boldsymbol{\mathcal{I}}^5$. The operations applied to such theories will only introduce new instantiation relations, which can then be monadicised. Arguably, the point in which only iteration relations are introduced via predicate push-down can be seen as a natural stopping point for the procedure.

Specifically, this means that the function δ on terms introduced on page 9 needs to apply to instantiation relations as well. As a general rule, we set, for σ a sequence of numbers,

$$\delta(\eta_\sigma) = e_\sigma,$$

where e_σ is an individual constant of the target language.

4 The General Formulation

We assume a sequence encoding method (e.g. the familiar one based on prime decomposition) to express types and indices by natural numbers. For the sake of definiteness, we take the sequence $(n_1, \ldots, n_k)$ is encoded as

$$p_0^{n_1+1} \times \ldots \times p_k^{n_k+1},$$

where p_i is the i^{th} prime. This function is easily seen to be primitive recursive, and so are the projection function.

Definition 1 (Language)**.** We start with the language of *relational* n^{th}-order logic $\mathcal{L}^n$. Types are defined as follows:

- 0 is the type of individuals;
- If $i_1, \ldots, i_l$ are types, then $(i_1, \ldots, i_l)$ is the type of an l-ary relation over entities of types $i_1, \ldots, i_l$, respectively.

The language features a denumerable stock of variables of each type. We assume a finite relational signature with the addition of finitely many non-logical constants $C_{j_1}^{i_1}, \ldots, C_{j_l}^{i_l}$ of respective types $i_1, \ldots, i_l$. Well-formed expressions are defined in the usual way.

Definition 2 (Initial Theory `I`)**.** The Logical axioms of `I` are the axioms of n^{th}-order logic *without* comprehension. Besides the usual logical constants, we assume only *one* primitive identity relation $=$ of type $(0, 0)$. We assume a finite set of non-logical, non-structural axioms including a finite set of atomic facts of form:

$$C_{j_{k+1}}^{(i_1,\ldots,i_k)}(C_{j_1}^{i_1}, \ldots, C_{j_k}^{i_k}) \text{ with } k < l. \tag{9}$$

Definition 3 (The first-order language $\mathcal{L}_\eta^1$)**.** We take language $\mathcal{L}_\eta^1$ is a first-order language featuring finitely many *non-structural* constant symbols and predicate symbols, including:

- the images of the $\mathcal{L}^n$ constants under the mapping δ defined below,
- a finite stock of 'domain' predicates $L_0, \ldots, L_{n-1}$,
- a denumerable list of instantiation relations $\eta_{(i_1,\ldots,i_m)}$.

Definition 4 (Pushdown Operation with Levels)**.** The translation $\tau_2 : \mathcal{L}^n \to \mathcal{L}_\eta^1$ presupposes a function δ dealing with terms (including variables of all order):

$$\delta(X_j^i) := x_{(i,j)}, \qquad \delta(C_j^i) := c_{(i,j)}.$$

Then we set:

$$\tau_2(s = t) :\leftrightarrow \delta(s) = \delta(t);$$
$$\tau(X_j^{(i_1,\ldots,i_k)}(S_{l_1}^{i_1}, \ldots, S_{l_k}^{i_k})) :\leftrightarrow \eta_{(i_1\ldots i_k,(i_1,\ldots,i_k))}(\delta(S_{l_1}^{i_1}), \ldots, \delta(S_{l_k}^{i_k}), \delta(X_j^{(i_1\ldots i_k)})) \quad S_{l_1}^{i_1} \ldots S_{l_k}^{i_k} \text{ arbitrary terms,}$$

$$\begin{aligned}\tau_2(\neg\phi) &:\leftrightarrow \neg(\tau_2(\phi))\\ \tau_2(\phi \wedge \psi) &:\leftrightarrow \tau_2(\phi) \wedge \tau(\psi)\\ \tau_2(\forall X^i_j \phi) &:\leftrightarrow \forall x_{(i,j)}(L_i(x_{(i,j)}) \rightarrow \tau_2(\phi))\end{aligned}$$

The target first-order theory, called $\mathtt{I}^2$, features the following axioms

1. The translation of of all finitely many nonlogical axioms, including the translation of the atomic facts of form (9):

$$\eta_{(i_1,\ldots,i_k,(i_1,\ldots,i_k))}(c_{(i_1,j_1)},\ldots,c_{(i_k,j_k)},c_{((i_1,\ldots,i_k),j_{k+1})}); \tag{10}$$

2. Each non-logical, non-structural constant belong to the appropriate level: $L_i(\delta(C^i_j))$.
3. The levels are exhaustive and mutually exclusive.

Proposition 1 (Interpretability)**.**

$$\mathtt{I} \vdash \phi(X^{i_1}_{j_1},\ldots,X^{i_k}_{j_k}) \;\Rightarrow\; \mathtt{I}^2 \vdash L_{i_1}(\delta(X^{i_1}_{j_1})) \wedge \ldots \wedge L_{i_k}(\delta(X^{i_k}_{j_k})) \rightarrow \tau_2(\phi)(\delta(X^{i_1}_{j_1}),\ldots,\delta(X^{i_k}_{j_k})).$$

In the next step, we are extending the theory $\mathtt{I}^2$ with *pairing*. We extend the language of $\mathtt{I}^2$ with a new primitive ternary relation Pair. The binary predicates Left and Right are defined as in Section 2.4.

Definition 5 ($\mathtt{I}^3$)**.** The axioms of the theory $\mathtt{I}^3$ are

$$\textsc{Pair}(x,y_1,z_1) \wedge \textsc{Pair}(x,y_2,z_2) \rightarrow (y_1 = y_2 \wedge z+1 = z_2) \wedge (\neg L_0(x) \wedge \ldots \wedge \neg L_n(x)) \quad \text{(Pair-Id-}L_n\text{)}$$

as well as axioms for the existence of pairs of entities corresponding to the finitely many atomic facts (9).

In $\mathtt{I}^3$, we can define *tuples* inductively:

$$\begin{aligned}\textsc{Pair}^0(z,x_1,x_2) &:\leftrightarrow \textsc{Pair}(z,x_1,x_2),\\ \textsc{Pair}^{n+1}(z,x_1,\ldots,x_{n+3}) &:\leftrightarrow \textsc{Pair}^n(z,\langle x_1,\ldots,x_{n+2}\rangle,x_{n+3}).\end{aligned}$$

Definition 6 (η-merging translation)**.** The translation τ^4 for η-merging operates on instantiation relations of the language of $\mathtt{I}^3$ only, and leaves everything else unchanged including all structural predicates (levels, pairing). Its defining clause is:

$$\begin{aligned}&\tau_4\left(\eta_{(i_1,\ldots,i_{k+2},(i_1,\ldots,i_{k+2}))}(c_{(i_1,j_1)},\ldots,c_{(i_{k+2},j_{k+2})},c_{((i_1,\ldots,i_{k+2}),j_{k+3})})\right)\\ &:\leftrightarrow \exists w \textsc{Pair}^k(w,c_{(i_1,j_1)},\ldots,c_{(i_{k+2},j_{k+2})}) \wedge \eta(w,c_{((i_1,\ldots,i_{k+2}),j_{k+3})}).\end{aligned}$$

Definition 7 ($\mathtt{I}^4$)**.** The theory $\mathtt{I}^4$ is formulated in a language that is just like the language of $\mathtt{I}^3$ except that it replaces all instantiation predicates η^i_j from $\mathtt{I}^3$ with a single, *binary* instantiation relation η. Its axioms are:

1. First-order logical axioms (with identity) for the new signature;
2. All structural axioms of $\mathtt{I}^3$, i.e. axioms concerning levels and pairing;
3. The result of applying the η-merging translation to the finitely many non-logical axioms of $\mathtt{I}^2$:

$$\eta(\langle c_{(i_1,j_1)}, \dots, c_{(i_k,j_k)}\rangle, c_{((i_1,\dots,i_k),j_{k+1})}). \tag{11}$$

Proposition 2.

$$\mathtt{I}^3 \vdash \phi(x_1 \dots, x_n) \Rightarrow \mathtt{I}^4 \vdash \phi_4^\tau(x_1, \dots, x_n).$$

The final operation we consider is a general version of Monadicisation introduced above. We apply it to the theory $\mathtt{I}^4$, but it can of course be applied to any theory including pairing. The language of $\mathtt{I}^4$ is modified only in that the binary instantiation predicate η is replaced by the unary $\eta^\top$. The theory then needs to be modified accordingly with the existence of the required pairs for Monadicisation (if not already present).

Definition 8 ($\mathtt{I}^5$)**.** The theory features all axioms of $\mathtt{I}^4$, except that the axioms (11) are replaced by

$$\eta^\top(\langle\langle c_{(i_1,j_1)}, \dots, c_{(i_k,j_k)}\rangle, c_{((i_1,\dots,i_k),j_{k+1})}\rangle). \tag{12}$$

To translate $\mathtt{I}^4$ into $\mathtt{I}^5$, we simply translate η into $\eta^\top$ and leave all else unchanged.

Definition 9 (Monadicising translation)**.** The translation τ^5 from the language of $\mathtt{I}^4$ into the language of $\mathtt{I}^5$ behaves like the identity translation on all logical connectives and quantifiers, on structural vocabulary; its defining clause is

$$\tau^5\big(\eta(\langle c_{(i_1,j_1)}, \dots, c_{(i_k,j_k)}\rangle, c_{((i_1,\dots,i_k),j_{k+1})})\big) :\leftrightarrow \exists w\, \textsc{Pair}(w, \langle c_{(i_1,j_1)}, \dots, c_{(i_k,j_k)}\rangle, c_{((i_1,\dots,i_k),j_{k+1})}) \wedge \eta^\top(w). \tag{13}$$

The translation τ^5 also preserves provability.

Proposition 3.

$$\mathtt{I}^4 \vdash \phi(x_1, \dots, x_n) \Rightarrow \mathtt{I}^5 \vdash \phi^5(x_1, \dots, x_5).$$

Finally, the considerations from subsection 3.7 carry over without modifications to the general setting.