

# Report

## Levelling Data Representations

Salvatore Florio, Carlo Nicolai, and Chris Partridge

### Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Schemata-as-patterns . . . . .	2
1.2	bClearer . . . . .	3
1.3	Logical methods and formalization . . . . .	3
1.4	Looking ahead . . . . .	3
<b>2</b>	<b>Operations</b>	<b>4</b>
2.1	Simple Predicate Pushdown . . . . .	5
2.2	Levels . . . . .	6
2.3	Arity reduction . . . . .	6
2.4	Pairing . . . . .	8
2.5	Predicate Merging . . . . .	9
<b>3</b>	<b>Unification Cycles</b>	<b>10</b>
<b>4</b>	<b>The translation in detail</b>	<b>12</b>
4.1	Initial Data . . . . .	12
4.2	Simple Predicate Pushdown . . . . .	12
4.3	Levelling . . . . .	13
4.4	Pairing . . . . .	14
4.5	$\eta$ -Merging . . . . .	15
4.6	Monadicisation . . . . .	16
4.7	Iteration . . . . .	17
<b>5</b>	<b>The General Formulation</b>	<b>18</b>

## 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

We introduce the basic operations involved in the unification cycles. They will be classified along the following key dimensions:

to check: predicates, instantiation,

<i>Formal Predicates</i>	What the operation determines for predicates that by-products of the unification process and do not directly correspond with initial data.
<i>Contentful Predicates</i>	What the operation determines for predicates that are directly associated with initial data.
<i>Domain Expansion</i>	Does the operation involves creation of new objects in the first-order domain?
<i>Iterations</i>	Is the operation involved in iteration cycles?
<i>Arity</i>	Does the operation determine an arity reduction for predicates?
<i>Order</i>	Does the operation determine an order reduction for predicates?

To introduce the operations, we will make use of a concrete UML example employed in [dCP16] to describe Boro's overall ontological framework.

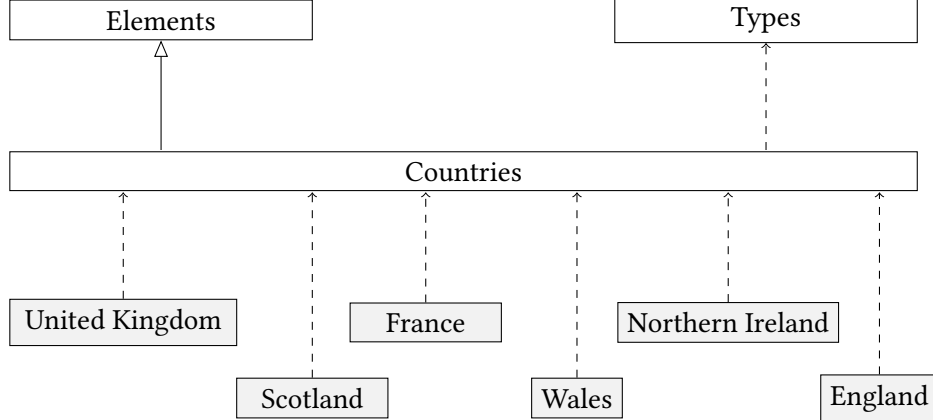


Figure 1: Grounding countries in BORO semantics using UML-style representation

There is straightforward translation of Figure 1 in a higher-order (in fact, second-order) language:

1. individual constants  $u_k, sco, fra, wal, n_i, eng$ .

2.  $\text{Elements}(\_)$  and  $\text{Countries}(\_)$  are predicate symbol of type (0), i.e. the type of classes of individuals. They satisfy

$$\forall x^0 (\text{Countries}(x^0) \rightarrow \text{Elements}(x^0)), \quad (1)$$

$$\text{Countries}(a) \text{ for } a \in \{u\_k, sco, fra, wal, n\_i, eng\}. \quad (2)$$

3.  $\text{Types}(\_)$  is a predicate of type ((0)) satisfying:

$$\text{Types}(\text{Countries}). \quad (3)$$

In employing Figure 1, it is sufficient to mostly focus on the behaviour of only two individual constants, *eng* and *sco*. To increase generality, we will also add binary relations  $R$  of type (0, 0) and  $\mathcal{Q}$  of type (1, 0), respectively, satisfying

$$R(eng, sco), \quad (4)$$

$$\mathcal{Q}(\text{Countries}, sco). \quad (5)$$

## 2.1 Simple Predicate Pushdown

The operation replaces syntactic predication with an explicit instantiation predicate. In our example:

$$\begin{aligned} \text{Countries}(eng) &\mapsto \eta_{01}(eng, cou) \\ R(eng, sco) &\mapsto \eta_{001}(eng, sco, r) \\ \text{Types}(\text{Countries}) &\mapsto \eta_{12}(cou, typ) \\ \eta_{01}(eng, cou) &\mapsto \eta_{(01)(01)}(eng, cou, e_{(01)}) \end{aligned}$$

The indices of the  $\eta$ -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.

Consider inserting the ‘rewriting operation’, our  $\delta$ .

**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.

Check instantiation as formal

**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  $\eta_{01}$  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’<sup>a</sup> predicates expressing syntactic types of the previous language.<sup>b</sup>

$$\begin{aligned} fra, eng, sco \dots &\mapsto L_0(fra), L_0(eng), L_0(sco) \dots \\ \text{Countries, Types, } X, \mathfrak{X} \dots &\mapsto L_1(cou), L_2(typ), L_1(x), L_2(\mathfrak{x}) \dots \\ \eta(\langle eng, cou \rangle) &\mapsto L_1^*(e), L_0(\langle eng, cou \rangle) \end{aligned}$$

<sup>a</sup>This could even be understood in terms of permutation invariance.

<sup>b</sup>In the example, fraktur alphabet is used for third-order variables.

**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 Arity reduction

Arity reduction can be realized in different ways. For our purposes, we focus on two operations. The first, *monadisation*, simply takes an  $n$ -ary predicate and replaces it with a unary one. The second, *discriminate arity reduction*, only reduces arity to a given positive number, by leaving untouched a given subset of the arguments of the relation. Again, for our specific purposes, we will only be interested in ‘freezing’ one argument and reduce the arity to 2; in particular,

we are interested in freezing the argument corresponding to the new objects—properties and relations—that have emerged from (Simple) Predicate Pushdown.

### 2.3.1 Independent arity reduction $n \mapsto 1$

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

$$\begin{aligned} R(eng, sco) &\mapsto R^*(\langle eng, sco \rangle) \\ \eta_{\langle 01 \rangle \langle \langle 01 \rangle \rangle}(\langle eng, cou \rangle, e) &\mapsto \eta_{\langle 01 \rangle \langle \langle 01 \rangle \rangle}^*(\langle \langle eng, cou \rangle, e \rangle) \end{aligned}$$

**Formal Predicates.** The operation rests on (but not formally introduces per se) formal predicates for pairing and associated projections.<sup>1</sup>

**Contentful Predicates.** The operation replaces contentful predicates with unary counterparts.

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

**Iteration.** The procedure may be applied to iterations of predicate pushdown but not itself iterated.

**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.

### 2.3.2 Pushdown-dependent arity reduction

The operation employs pairing to turn relations that are *recognized as* instantiation relations with arity  $\geq 2$  into instantiation relations of arity 2: given an  $n$ -ary instantiation relation, its  $n^{\text{th}}$  argument is untouched, all others are turned into a tuple.

$$\begin{aligned} \eta_{01}(eng, cou) &\mapsto \eta_{01}(eng, cou) \\ \eta_{001}(eng, cou, r) &\mapsto \eta_{\langle 00 \rangle 1}(\langle eng, cou \rangle, r) \\ \eta_{102}(cou, sco, q) &\mapsto \eta_{\langle 10 \rangle 2}(\langle cou, sco \rangle, q). \end{aligned}$$

Add reference/dependency to predicate pushdown, and mention the more general operation of taking arbitrary subsets?

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

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

**Contentful Predicates.** The operation replaces contentful instantiation relations with binary counterparts.

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

**Iteration.** The procedure may be applied to iterations of predicate pushdown but not itself iterated.

**Arity.** The operation reduces predicates' arity, 'turning' instantiation predicates into binary ones.

**Order.** The operation always operates at the first-order level, specifically to first-order instantiation relations. 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:

$$eng, sco \dots \mapsto \exists x \text{PAIR}(x, eng, sco)$$

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

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

$$\exists z \text{PAIR}(x, y, z) \leftrightarrow : \text{LEFT}(x, y) \quad (\text{Pair-Left})$$

$$\exists y \text{PAIR}(x, y, z) \leftrightarrow : \text{RIGHT}(x, z) \quad (\text{Pair-Right})$$

**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 Predicate Merging

The operation merges a given collection of different predicates of the same arity with a single predicate. These predicates may or may not be instantiation relations. In our application we apply the operation to predicates already recognized to be instantiation predicates:

$$\begin{aligned}\eta_{01}^*(eng, cou) &\mapsto \eta^\top(eng, cou) \\ \eta_{\langle 00 \rangle 1}^*(\langle eng, sco \rangle, r) &\mapsto \eta^\top(\langle eng, sco \rangle, r) \\ \eta_{\langle 10 \rangle 2}^*(\langle cou, sco \rangle, q) &\mapsto \eta^\top(\langle cou, sco \rangle, q)\end{aligned}$$

**Formal Predicates.** The operation does not *directly* make use of formal predicates.

**Contentful Predicates.** The operation replaces a collection of different contentful predicates, which correspond to data schemes, with a single predicate of the same arity.

**Domain Expansion.** The operation does not involve domain expansion.

**Iterations.** The operation is not directly iterated, but can be applied instantiation relations resulting from previous predicate merging.

**Arity.** The operation leaves the arity of the predicates untouched.

**Order.** The operation takes place already at the first-order level.

### 3 Unification Cycles

The operations introduced above can be arranged in suitable application cycles. In order to capture some essential features of the Boro methodology ??, we suggest two natural *initial* cycles, one giving priority to *order* reduction rather than *arity* reduction (Table 1), the other prioritizing *arity* over *order* (Table 2). In addition, there is a final *iteration* cycle.

As it shall be clear in later section when the full formalization will be presented, each step in the unification cycles preserve consistency.

Table of functions to be called in the boxes.

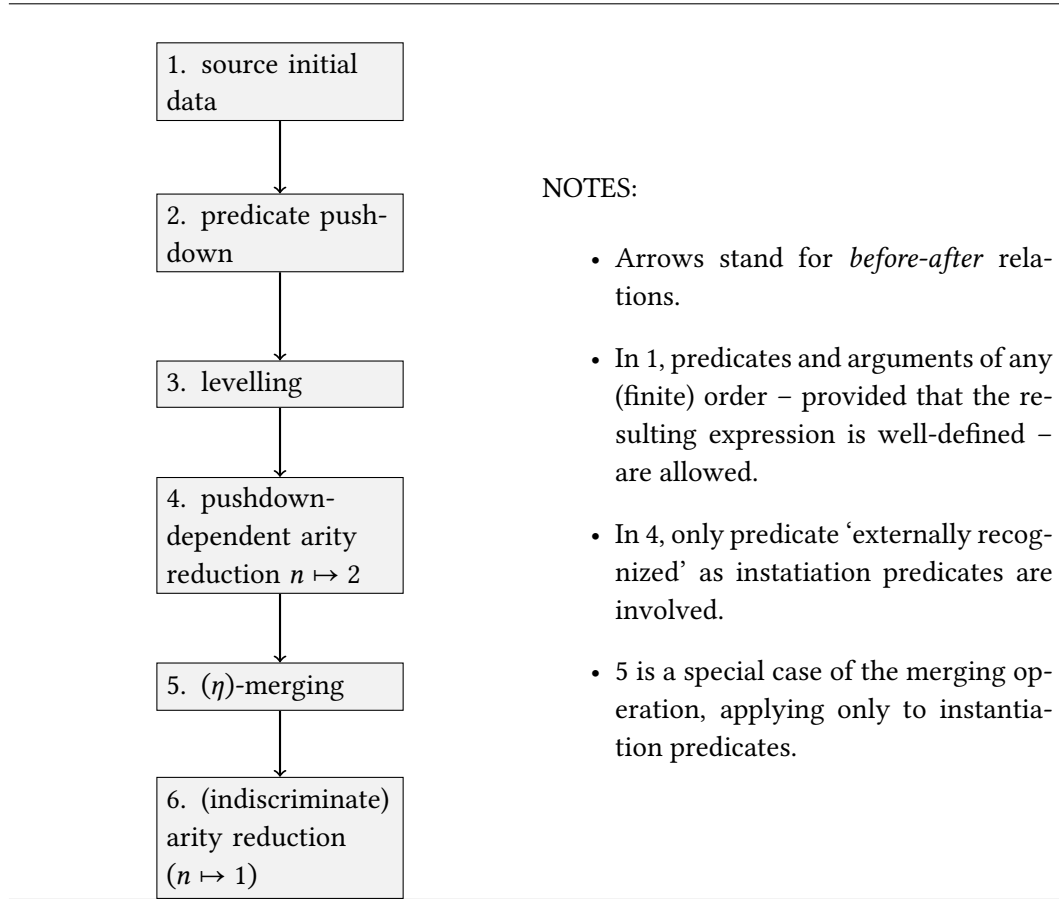


Table 1: Order-arity reduction cycle.

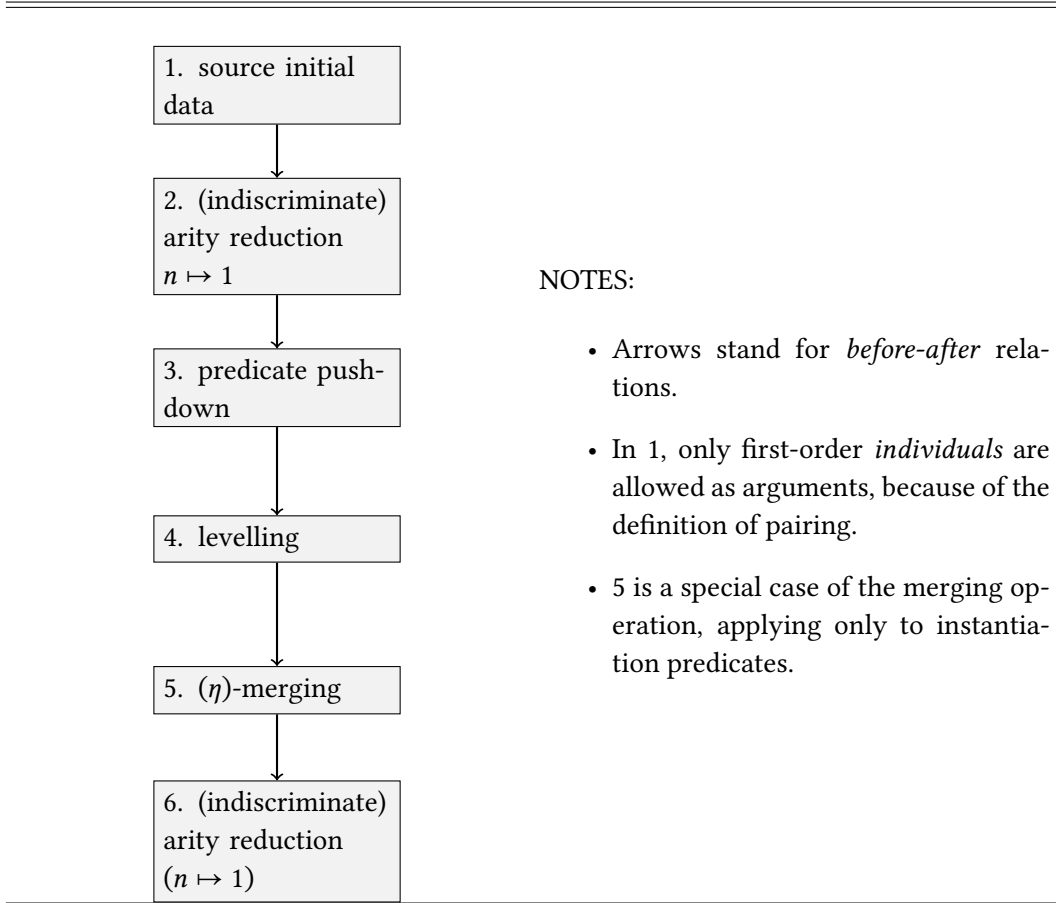


Table 2: Arity-order reduction cycle.

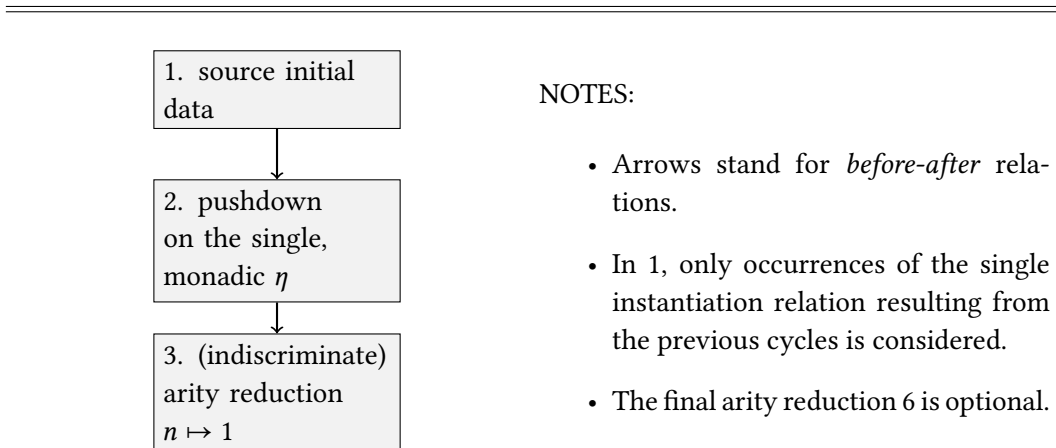


Table 3: Iteration cycle

## 4 The translation in detail

### 4.1 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 $\mathcal{I}$		
	Countries( <i>eng</i> )	(Countries)
	$R(\textit{eng}, \textit{sco})$	( $R$ )
	$\mathcal{Q}(\textit{Countries}, \textit{sco})$ .	( $\mathcal{Q}$ )

### 4.2 Simple Predicate Pushdown

Simple predicate pushdown can be formalized as a translation  $\tau_1$  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  $\delta$  whose domain is the set of variables and constants of the language considered, which behaves as follows:

$\delta(x_i) := x_{2i+1}$	$\delta(X_i) := x_{2i}$	
$\delta(\textit{eng}) := \textit{eng}$	$\delta(\textit{sco}) := \textit{sco}$	
$\delta(\textit{Countries}) := \textit{cou}$	$\delta(R) := r$	$\delta(\mathcal{Q}) := q$

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

The translation  $\tau_1$

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

Finally, the translation  $\tau_1$  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:

$$\begin{aligned}
 \eta_{01}(\text{eng}, \text{cou}) & \quad (\text{Countries}^1) \\
 \eta_{001}(\text{eng}, \text{sco}, r) & \quad (R^1) \\
 \eta_{102}(\text{cou}, \text{sco}, q) & \quad (\mathcal{Q}^1)
 \end{aligned}$$

Proposition

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

### 4.3 Levelling

The translation  $\tau_1$  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  $\tau_2$  from the language of  $\mathcal{I}$  into this augmented language which is just like  $\tau_1$  but suitably relativizes quantifiers.

#### The Levelling Translation $\tau_2$

$\tau_2$  behaves just like  $\tau_1$  on atomic formulae and propositional connectives (including  $\delta$ ). For quantifiers:

$$\begin{aligned}\tau_2(\forall x_i \phi) &: \leftrightarrow \forall x_{2i+1} (L_0(x_{2i+1}) \rightarrow \tau_2(\phi)) \\ \tau_2(\forall X_i \phi) &: \leftrightarrow \forall x_{2i} (L_1(x_{2i}) \rightarrow \tau_2(\phi))\end{aligned}$$

$\tau_2$  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

$$\begin{aligned}L_0(eng) \wedge L_0(sco) & \quad (L_0) \\ L_1(cou) \wedge L_1(r) & \quad (L_1) \\ L_2(q) & \quad (L_2) \\ (L_0(x) \vee L_1(x) \vee L_2(x)) \wedge \neg(L_0(x) \wedge L_1(x)) \wedge \neg(L_1(x) \wedge L_2(x)) \wedge \neg(L_0(x) \wedge L_2(x)) & \quad (LED)\end{aligned}$$

Finally, the translation  $\tau_2$  indeed preserves provability in  $\mathcal{I}$ .

#### Proposition

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

## 4.4 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:<sup>a</sup>

$$(\text{PAIR}(x, y_1, z_1) \wedge \text{PAIR}(x, y_2, z_2)) \rightarrow (y_1 = y_2 \wedge z_1 = z_2) \wedge (\neg L_0(x) \wedge L_1(x) \wedge \neg L_2(x))$$

(Pair-Identity-L)

$$\exists x \text{PAIR}(x, \text{eng}, \text{cou}) \quad \text{(Pair-E1)}$$

$$\exists x \text{PAIR}(x, \text{eng}, \text{sco}) \quad \text{(Pair-E2)}$$

$$\exists x \text{PAIR}(x, \langle \text{eng}, \text{sco} \rangle, r) \quad \text{(Pair-E3)}$$

$$\exists x \text{PAIR}(x, \text{cou}, \text{sco}) \quad \text{(Pair-E4)}$$

$$\exists x \text{PAIR}(x, \langle \text{cou}, \text{sco} \rangle, q) \quad \text{(Pair-E5)}$$

⋮

---

<sup>a</sup>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’

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

## 4.5 $\eta$ -Merging

The operation of  $\eta$ -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  $\eta$ -merging amounts to a translation which replaces the multiple  $\eta$ s 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  $\eta$ :

#### The Translation $\tau_4$

$\tau_4$  leaves complex formulae unchanged, and operates at the level of atomic formulae (leaving identity unchanged) as follows:

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

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  $\eta$ .

#### $\mathcal{I}^4$

The language of  $\mathcal{I}^4$  is obtained from the language of  $\mathcal{I}^3$  by replacing the three instantiation relations  $\eta_{01}$ ,  $\eta_{001}$ , and  $\eta_{102}$  with a single instantiation relation  $\eta$ . Its axioms are the axioms of  $\mathcal{I}^3$ , except that  $P^1$ ,  $R^1$ , and  $\mathcal{Q}^1$  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  $\mathcal{Q}^1$ , we can formulate the new axioms as:

$$\eta(\text{eng}, \text{cou}) \tag{6}$$

$$\eta(\langle \text{eng}, \text{sco} \rangle, r) \tag{7}$$

$$\eta(\langle \text{cou}, \text{sco} \rangle, q). \tag{8}$$

Again, one can show that the translation  $\tau_4$  preserves provability.

#### Proposition

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

## 4.6 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  $\eta$  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  $\eta$  in the language of  $\mathcal{I}^4$  with a *unary*  $\eta^\top$ .



$\tau^5$

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

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

To ensure that  $\tau_5$  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  $\mathcal{I}^4$ ):

$$\exists w(\text{PAIR}(w, \langle \text{eng}, \text{sco} \rangle, r) \tag{9}$$

$$\exists w(\text{PAIR}(w, \langle \text{cou}, \text{sco} \rangle, q). \tag{10}$$

Moreover, the axioms (6)-(8) of  $\mathcal{I}^4$  are replaced with<sup>a</sup>

$$\eta^\top(\langle \text{eng}, \text{cou} \rangle) \tag{11}$$

$$\eta^\top(\langle \langle \text{eng}, \text{sco} \rangle, r \rangle) \tag{12}$$

$$\eta^\top(\langle \langle \text{cou}, \text{sco} \rangle, q \rangle) \tag{13}$$

<sup>a</sup>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

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

## 4.7 Iteration

In the iteration cycle 3, 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  $\delta$  on terms introduced on page 13 needs to apply to instantiation relations as well. As a general rule, we set, for  $\sigma$  a sequence of numbers,

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

where  $e_\sigma$  is an individual constant of the target language.

## 5 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  $(n_1, \dots, n_k)$  is encoded as

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

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  $i_1, \dots, i_l$  are types, then  $(i_1, \dots, i_l)$  is the type of an  $l$ -ary relation over entities of types  $i_1, \dots, i_l$ , 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  $C_{j_1}^{i_1}, \dots, C_{j_l}^{i_l}$  of respective types  $i_1, \dots, i_l$ . 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:

$$C_{j_{k+1}}^{(i_1, \dots, i_k)}(C_{j_1}^{i_1}, \dots, C_{j_k}^{i_k}) \quad \text{with } k < l. \quad (14)$$

**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  $\delta$  defined below,
- a finite stock of ‘domain’ predicates  $L_0, \dots, L_{n-1}$ ,
- a denumerable list of instantiation relations  $\eta_{(i_1, \dots, i_m)}$ .

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

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

Then we set:

$$\begin{aligned} \tau_2(s = t) &:= \delta(s) = \delta(t); \\ \tau(X_j^{(i_1, \dots, i_k)}(S_{l_1}^{i_1}, \dots, S_{l_k}^{i_k})) &:= \eta_{(i_1 \dots i_k, (i_1, \dots, i_k))}(\delta(S_{l_1}^{i_1}), \dots, \delta(S_{l_k}^{i_k}), \delta(X_j^{(i_1 \dots i_k)})) \quad S_{l_1}^{i_1} \dots S_{l_k}^{i_k} \text{ arbitrary terms,} \end{aligned}$$

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

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

1. The translation of of all finitely many nonlogical axioms, including the translation of the atomic facts of form (14):

$$\eta_{(i_1, \dots, i_k, (i_1, \dots, i_k))}(c_{(i_1, j_1)}, \dots, c_{(i_k, j_k)}, c_{((i_1, \dots, i_k), j_{k+1})}); \quad (15)$$

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

**Proposition 1** (Interpretability).

$$\mathcal{I} \vdash \phi(X_{j_1}^{i_1}, \dots, X_{j_k}^{i_k}) \Rightarrow \mathcal{I}^2 \vdash L_{i_1}(\delta(X_{j_1}^{i_1})) \wedge \dots \wedge L_{i_k}(\delta(X_{j_k}^{i_k})) \rightarrow \tau_2(\phi)(\delta(X_{j_1}^{i_1}), \dots, \delta(X_{j_k}^{i_k})).$$

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

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

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

as well as axioms for the existence of pairs of entities corresponding to the finitely many atomic facts (14).

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

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

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

$$\begin{aligned}\tau_4(\eta_{(i_1, \dots, i_{k+2}, (i_1, \dots, i_{k+2}))}(c_{(i_1, j_1)}, \dots, c_{(i_{k+2}, j_{k+2})}, c_{((i_1, \dots, i_{k+2}), j_{k+3})})) \\ : \leftrightarrow \exists w \text{PAIR}^k(w, c_{(i_1, j_1)}, \dots, c_{(i_{k+2}, j_{k+2})}) \wedge \eta(w, c_{((i_1, \dots, i_{k+2}), j_{k+3})}).\end{aligned}$$

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

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

$$\eta(\langle c_{(i_1, j_1)}, \dots, c_{(i_k, j_k)} \rangle, c_{((i_1, \dots, i_k), j_{k+1})}). \quad (16)$$

**Proposition 2.**

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

The final operation we consider is a general version of Monadicisation introduced above. We apply it to the theory  $\mathcal{I}^4$ , but it can of course be applied to any theory including pairing. The language of  $\mathcal{I}^4$  is modified only in that the binary instantiation predicate  $\eta$  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** ( $\mathcal{I}^5$ ). The theory features all axioms of  $\mathcal{I}^4$ , except that the axioms (16) are replaced by

$$\eta^\top(\langle \langle c_{(i_1, j_1)}, \dots, c_{(i_k, j_k)} \rangle, c_{((i_1, \dots, i_k), j_{k+1})} \rangle). \quad (17)$$

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

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

$$\tau^5(\eta(\langle c_{(i_1, j_1)}, \dots, c_{(i_k, j_k)} \rangle, c_{((i_1, \dots, i_k), j_{k+1})})) : \leftrightarrow \exists w \text{PAIR}(w, \langle c_{(i_1, j_1)}, \dots, c_{(i_k, j_k)} \rangle, c_{((i_1, \dots, i_k), j_{k+1})}) \wedge \eta^\top(w). \quad (18)$$

The translation  $\tau^5$  also preserves provability.

**Proposition 3.**

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

Finally, the considerations from subsection 4.7 carry over without modifications to the general setting.

## References

- [dCP16] Sergio de Cesare and Chris Partridge. Boro as a foundation to enterprise ontology. *Journal of Information Systems*, 30(2):83–112, 2016.