Consensus-Free Spreadsheet Integration

Brandon Baylor, James Hansen, Esteban Montero
Chevron

Eric Daimler, Ryan Wisnesky
Conexus AI

September 30, 2022

Abstract

We describe a method for merging multiple spreadsheets into one sheet, and/or exchanging data among the sheets, by expressing each sheet’s formulae as an algebraic (equational) theory and each sheet’s values as a model of its theory, expressing the overlap between the sheets as theory and model morphisms, and then performing “colimit”, “lifting”, and “Kan-extension” constructions from category theory to compute a canonically “universal” integrated theory and model, which can then be expressed as a spreadsheet. Our motivation is to find methods of merging engineering models that do not require consensus (agreement) among the authors of the models being merged, a condition fulfilled by our method because theory and model morphisms are semantics-preserving. We describe a case study of this methodology on a real-world oil and gas calculation at a major energy company, describing the theories and models that arise when integrating two different casing pressure test (MASP) calculation spreadsheets constructed by two non-interacting engineers. We also describe the automated theorem proving burden associated with both verifying the semantics preservation of the overlap mappings as well as verifying the “conservativity”/“consistency” of the resulting integrated sheet. We conclude with thoughts on how to apply the methodology to scale engineering efforts across the enterprise.

1 Introduction

As the energy industry moves to model-based software for performing day-to-day engineering tasks, it becomes more and more important to ensure the semantic consistency of models composed of related models (integrated models). For example, we don’t want errors to propagate from one model to another or to try to integrate models with conflicting requirements (e.g. positive and negative voltage at the same time). Commonly, to ensure semantic consistency of integrated models the human subject matter experts that created the input models communicate informally with each other about their respective understandings of the integrated model and check for consistency of the integrated model with respect to their original models (think NASA mission control with its various ‘functions’). Although using groups of human experts to certify/construct integrated models works for small numbers of input models, this methodology is both costly and not scalable, because in principle, all the humans may need to communicate with each other (not everything can pass through mission control), and moreover, people may disagree about the meaning of the input and/or integrated models. Therefore, to construct/certify engineering models in a scalable way, we must eliminate the need to relate each input model to each other, and we must make sure that when models are merged, the original authors of the models need not be involved.

1.1 Contributions

In this paper, we propose a methodology that does precisely the above for many spreadsheet-based engineering models, based on treating each sheet as both an algebraic (equational) theory [6] and a model of that theory (a so-called “olog” [13]) and using techniques from automated theorem proving [6] and category theory [5] (summarized in [11]) to construct composed ologs (theories and models), which can then be exported as spreadsheets. Relationships between input ologs are captured as theory and model “morphisms” [9] (olog morphisms), for which we generate and solve [11] verification conditions to ensure they are semantics-preserving without recourse to the original spreadsheet authors. We demonstrate the practicality
of our methodology on a maximum anticipated surface pressure (MASP) calculation done in two different spreadsheets by two non-interacting engineers, describing the associated ologs as well as the verification conditions generated by both relating the models and integrating them. Specifically, our research contributions (new results) are:

- a method to represent some spreadsheets as algebraic theories [6] and models (‘ologs’ [13]), and vice versa; and,
- a case study about two independently-developed ologs corresponding to a standard spreadsheet-based MASP calculation, as well as another olog and two olog morphisms [9] capturing the relationship of the first two ologs to each other; and,
- a use case about how to apply the technology demonstrated in the case study to reducing costs in auditing spreadsheet-based models for requirements compliance.

1.2 Outline

In the remainder of the introduction, we provide background on MASP calculations, their applications, and why integrating/exchanging data between them can be useful. The rest of this paper is structured as follows:

1. In section 2 we describe how spreadsheets can be represented as ologs and vice-versa.

2. In section 3 we describe the two source sheets/theories/models of MASP calculations to be merged, the overlap between them, and integrated result and its applications. We also discuss the practicalities of the case study.

3. In section 4 we describe the verification conditions generated by relating the sources to each other and during construction of the integrated result. This section will be of interest primarily to computer scientists.

4. In section 5 we conclude by discussing applications of our methodology across the enterprise.

It is our hope that our methodology can be applied by engineers without any training in category theory, logic, or algebra, and that this paper should be readable without such knowledge as well. However, to fully understand this paper, knowledge of all three areas is required, and to that end, we have included appendices that review them (Appendices 6, 7, and 8, respectively).

1.3 MASP Background

Maximum anticipated surface pressure (MASP) is the highest (worst case) surface pressure that is expected to be encountered throughout the well construction process [4]. This MASP calculation is a key engineering activity for both preliminary and detailed well design because it impacts well control equipment specifications and pressure test plans. In particular, MASP determines the sizing and ratings for pressure control/containment equipment such as blowout preventers and strings of casing [2]. The results of the MASP calculation are also used to derive numerical values to test said pressure control/containment devices in the field. These are key aspects of maintaining process safety and integrity of oil and gas wells during drilling, completion, or intervention operations.

MASP is normally calculated for each casing section as part of the well design process. This typically assumes a shut-in well and various types of fluid in the wellbore (e.g. mud or gas). Guidance for determining the appropriate calculation scenarios (e.g. the correct ratio of fluid in the well) are often provided by global standards and technical requirements. Other inputs include pore pressure and fracture gradient data (obtained from offset wells or regional analogues), fluid density values, well depths, and casing specifications.

The determination of MASP typically includes pore pressure uncertainties, new geologic zones of interest being added/encountered, regulatory requirements, and other unanticipated factors. For these reasons there are verification activities to assure/audit engineering calculations are sound and compliant with technical requirements. Moreover, there are often several engineers working on singular aspects of the well plan and then collaborating to ensure an optimal design. In either case, auditors or engineers may want to integrate their respective viewpoints to generate new insights and ensure explicit alignment between each other, suggesting use cases described in the conclusion.
The primary research contribution of this paper, besides the case study, is a methodology for representing certain spreadsheets as ologs \[13\] (theories and models), which we describe in this section. We begin by observing that it is trivial to represent every spreadsheet as a model \[11\] of a signature containing one constant symbol for every used cell. For example, the initial model of the theory:

\[
\begin{align*}
& b_{17} \quad c_{12} : \text{Float} \\
& b_{17} = .07 \\
& c_{12} = 13722
\end{align*}
\]

formalizes a fragment of the MASP sheet in Figure 1. In contrast, this paper’s method is based on leveraging the typical, natural decomposition of most sheets into smaller tables into order to provide additional semantics (sorts and symbols and equations). First, we define a notion of categorical normal form for a sheet, from \[12\]. Then we define translations to/from olog sheets in this normal form. A spreadsheet is in this form when:

- the sheet is composed entirely of rectangular sub-tables, each with a name, a set of named columns, and a distinguished “primary key” column. The cells in the primary key column are called the row-ids of t and semantically row-ids are meaningless identifiers, not data;
- for every column c of a table t there is either some table t’ such that the value in each cell of c refers to some row-id of t’ (in which case we say that c is a “foreign key” column from t to t’; or column c of table t is a “pure data” / “value” column of t with values in some non-row-id type such as Float.

Of course, most spreadsheets are not in categorical normal form to begin with. Fortunately, we have found that human subject matter experts (not computer scientists) can rapidly and accurately create new spreadsheet tabs that are in the above form and that reference back into the original sheets. These new tabs provide an auditable “olog view” of the original sheet, and heuristic methods to automatically generate these olog tabs are a subject of current work. In our case study, two engineers (Brandon and James)
both independently created olog tabs from the government-provided MASP calculation spreadsheet shown in Figure 1. In practice, each engineer may start with a different spreadsheet, but the olog views of the two engineers for this one government sheet are so different that for the purposes of this paper there’s no difference between starting from one sheet or two– what matters is starting from two different ologs.

Remark. In this paper we will always be working with spreadsheets that correspond to models in the sense of logic [11], but the CQL software that we are using [1] allows us to work with “presentations” of spreadsheets (those that are missing values in foreign key columns; missing values in data columns don’t matter here) by automatically constructing initial models of equational theories for us.

2.1 Spreadsheet to Olog

The above categorical norm suggests an easy way to generate a signature (sorts and symbols), especially when column headers are used to indicate the foreign keys directly in Excel. However, categorical normal form doesn’t suggest any axioms along with the generated signature. The way in which we recover axioms is by examining the formulae of the sheet. In particular, to be imported as a universally quantified equation, we require that a column’s rows all contain “the same” formula– a formula that is a function of only its row – which itself must be written entirely of spreadsheet functions applied to “lookups” of foreign-key columns. For example, in the MASP olog from Source A, the MASP Calc. Step 1 table contains a column called 70% Burst (not-corrected) (column C) whose rows have the form:

\[
\begin{align*}
C76 &= \text{LOOKUP}(E76, \$A\$13:\$A\$41,\$F\$13:\$F\$41)\ast B76 \\
C77 &= \text{LOOKUP}(E77, \$A\$13:\$A\$41,\$F\$13:\$F\$41)\ast B77 \\
&\ldots
\end{align*}
\]

where column E has Casing Section as a header and column B has De-rated Percent as a header. Moreover, the indicated ranges are those of the ID and Burst Rating column in another table. As such we import the following equation (also seen in Figure 4):

\[
\forall x::\text{"MASP Calc. Step 1", "70% Burst (not corrected)"}(x) = \\
\text{"Burst Rating"("Casing Section"(x)) \ast "De-Rated Percent"(x)}
\]

In the above example our columns are all producing numbers. In general, however, columns may also produce boolean values indicating the equivalence of two arbitrary expressions. We say that such columns are not “definitional” in the sense of the 70% burst column in the example above. The input MASP calculations for this paper contain entirely “definitional” formulae, but as described later the integrated result contains some columns that are not definitional (the integrated sheet contains some boolean-valued columns that constrain other columns).

2.2 Olog to Spreadsheet

With one caveat, to convert from ologs to spreadsheets it is necessary to merely invert the representation in the above section. The caveat is that the above discussion does not describe how to encode “type algebras” [11] – missing values and equations between missing values that can appear in ologs but not in sheets that people would construct naturally. To encode a non-trivial type algebra, we must include a single-column table in the spreadsheet for each type, such as integer and string. Then, each “skolem variable”/missing value at that type becomes a row in this new table, and formulae may refer to this new cell (formulae that do may compute as N/A in Excel, of course). Finally, for each type we also require another table with two columns wherein ground equations between these missing values are encoded as in the preceding section. An example is shown below.

| A | B   | C   | D   | E               | F               |
|---|-----|-----|-----|-----------------|-----------------|
| 1 | Person | Age | Integer | IntegerEqs     |                 |
| 2 | p1   | 20  | x     | \(=(D2\ast D3=20)\) |
| 3 | p2   |     | =D2   | y               |                 |
| 4 | p3   |     | =D3   |                 |                 |
Of course, when we import a spreadsheet containing a type algebra as encoded above, we must be careful to import the type tables as a type algebra, and not as “user tables”. In this paper’s example, the integrated result does have a non-trivial type algebra, because it contains blank cells in calculated columns in the original sheets. Whether our result’s type algebra is contradictory is discussed in Section 4.

Remark. Our olog to spreadsheet and reverse encodings fail on empty ologs because there are no corresponding spreadsheet cells with which to record equations. This problem can be remedied by including, for each empty table, a “phantom” row that records the equations with non-formulae cells left blank. Or, by writing the equations down as additional column headers.

2.3 Free Theories Not Closed Under Colimits

We might ask, “if we start with two definitional/free sheets, turn them into ologs, compose them using colimits as described later, and then turn the result back into a sheet, will the resulting sheet be entirely definitional/free too”? That is, if we start with sheets where every column is or computes a number will we end up with a sheet where every column is or computes a number? The answer is no: in mathematical terms, a free theory such as \( f(x) = x^2 \) and a free theory such as \( g(y) = y^3 \) can be merged along the free theory \( f(z) = g(z) \) to give a theory in which \( x^2 = x^3 \), i.e., a non-free (non-definitional) theory. A non-free theory requires boolean-valued columns to represent as a sheet and requires more effort to reason about than a free theory but is otherwise benign\(^1\). In our MASP example, failure of definitionality preservation under colimits manifests when we have multiple equations that define the same symbol, such as Well below:

\[
\text{forall } x, \quad x.\text{Well} = x."\text{TVD Shoe}".\text{Well}
\]

\[
\text{forall } x, \quad x.\text{Well} = x."\text{TVD Deepest OH}".\text{Well}
\]

The theory above is not free because it contains non-trivial equations such as

\[
\text{forall } x, \quad x."\text{TVD Shoe}".\text{Well} = x."\text{TVD Deepest OH}".\text{Well}
\]

and so when exported as a spreadsheet will contain a boolean-valued column witnessing the truth of the above equation; this boolean-valued column is considered part of the schema, not part of the data, of the sheet’s corresponding olog. For example, any sheet containing the equations above would be exported as

| Step1 | Well | TVD Shoe | TVD Deepest OH | x.\text{Well} = x."\text{TVD Shoe}".\text{Well} | x.\text{Well} = x."\text{TVD Deepest OH}".\text{Well} |
|-------|------|----------|----------------|-----------------------------------------------|-----------------------------------------------|
| 1     | s1   | w1       | x1             | y1                                           | true                                          |
|       |      |          |                |                                              | x."\text{TVD Deepest OH}".\text{Well}        |
| 2     | s2   | w2       | x2             | t2                                           | true                                          |
| 3     | ...  |          |                |                                              | true                                          |

where each true is obtained by evaluating the formula in the column it appears in, where that formula will contain lookups etc as described earlier.

A related question is whether or not the merge of two finite sheets is always finite, where the answer is also no, such as when the schema \( \cdot \rightarrow_p \cdot \) is merged with the schema \( \cdot \leftarrow_q \cdot \) along the left and right dots, resulting in a schema with a cycle that is not “broken” by equations \( p(q(x)) = x \) and \( g(p(x)) = x \) and thus which will force the integrated sheet to be infinite when there is no data level overlap of the sources (having e.g. a row \( c \), and a row \( c.p \), and a row \( c.p.q \) not equal to \( c \), and so on). In such a case it is customary to add additional equations to break the cycle and force a finite model.

2.4 Axiomatizing Spreadsheet Functions

In this subsection we axiomatize a small collection of spreadsheet functions that are sufficient for our case study: (a dependent type theory induced by) the resulting theory is the Type input in the algebraic integration design pattern in Figure 2. We start with an infinite ground (variable-free) equational theory for addition

\(^1\)For example, spreadsheets can only compute initial models of free/definitional theories, implying that there is no spreadsheet that can compute the algebraic merge of two other spreadsheets– the merge must be done externally to Excel (where it can, for example, create an infinite number of rows or fail with contradictions) and then re-expressed as a sheet.
\[ S_O \xrightarrow{M_A} S_A \]
\[ M_B \xrightarrow{k_B} \]
\[ S_B \xrightarrow{N_B} S \]

For any other \( S' \), \( N'_A, N'_B \), \( I' \), \( k'_A, k'_B \) making the above diagram commute, there is a unique schema mapping \( M : S \rightarrow S' \) and data mapping \( k : I \rightarrow I' \) such that \( N'_A = N_A; M \) and \( N'_B = N_B; M \) and \( k'_A = k_A; k \) and \( k'_B = k_B; k \).

- Type is an algebraic (equational) theory representing types and functions (addition, subtraction, etc)
- \( S_O, S_A, S_B \) are input schemas/theories (with \( S_O \) the overlap) and \( S \) is the output integrated schema/theory (excel formulae)
- \( I_O : S_O \rightarrow \text{Type}, I_A : S_A \rightarrow \text{Type}, \) and \( I_B : S_B \rightarrow \text{Type} \) are input models/databases (with \( I_O \) the overlap) and \( I : S \rightarrow \text{Type} \) the output integrated model/database (excel values)
- \( M_A : S_O \rightarrow S_A \) and \( M_B : S_O \rightarrow S_B \) are input schema mappings/theory morphisms, and \( N_A : S_A \rightarrow S \) and \( N_B : S_B \rightarrow S \) are output schema mappings/theory morphisms (functors)
- \( h_A : I_0 \Rightarrow M_A; I_A \) and \( h_B : I_O \Rightarrow M_B; I_B \) are input data mappings/modem morphisms, and \( k_A : I_B \Rightarrow N_A; I \) and \( k_B : N_B; I \) are output data mappings/model morphisms (natural transformations).

**Figure 2: Algebraic Schema and Data Integration**

\[ +, \text{ subtraction} -, \text{ multiplication} *, \text{ maximum} \ \text{MAX}, \text{ and minimum} \ \text{MIN}, \text{ all of which are binary operations on floating point numbers:} \]

\[
0+0 = 0 \quad 0+1 = 1 \quad 1+2.2 = 3.2 \quad \ldots \quad 0*0 = 0 \quad 0*1 = 0 \quad 1*2.2 = 2.2 \quad \ldots \quad \text{MAX}(3.5, 2) = 3.5 \quad \ldots
\]

From there, we add the non-ground (universally quantified) axioms stating that \((0, 1, +, \times)\) form a commutative ring [5], as well as some axioms stating basic properties of how max and min interact with arithmetic.

**Remark.** In this paper, we have associated a type with each spreadsheet cell, either floating point or string. However, in many spreadsheets, cells are not typed, and functions such as addition will throw errors on non-numeric inputs. This dynamically-typed situation can also be represented in the manner of this section, with a “universal type”. Multiple types also allow for the possibility of representing different units (feet, inches, etc) as different types, reducing risk of error.

**Remark.** One of the original sheets used division in a single column, which for expediency we removed in favor of treating the column as plain data so as not to need to deal with division at all in this case study.

## 3 MASP Sheet Formalizations and Integration

We now recall the algebraic data integration design pattern [11], displayed in Figure 2. Our goal in this section is to describe all of the inputs and some of the outputs in the above diagram, including how they were constructed, either by human or machine, for our MASP example.

**Remark.** Intuitively, the guarantee provided by algebraic data integration is that the output of our methodology “is at least as good as every solution” to the problem of integrating the input ologs and overlap. Universality also guarantees a complete “lineage” or “provenance” of how every output row was constructed, although we do not elaborate further in this paper.

### 3.1 Schema Integration

In this section, we assume that the input spreadsheets have been converted to ologs as described in the previous sub-sections and discuss the resulting schema (theory part) of the olog conversion. Recall that schema integration is the process of taking input schemas (theories) \( S_O, S_A, S_B \) and computing output schema (theory) \( S \), according to the usual algebraic integration diagram in Figure 2.
3.1.1 Schema A

The signature (sorts and symbols) of schema A are defined in Figure 3. Each edge denotes a unary function between tables and each row in the table associated to a node is a unary function targeting a type (String, Int, etc) (i.e., a “data column”). The equations for Schema A are defined in Figure 4.

Schema A contains information about the subject well as defined by the Header Info table in Figure 3. This sets a broad context for where MASP is being calculated along with high level data that applies to all hole sections where the engineering analysis is performed. These specific hole sections are defined by multiple tables: Casing Section, Zone of Interest, and Interval Info. These tables reflect the general well construction process and ensures alignment between equipment specifications (such as casing) and their associated depths (where formation pressure inputs apply). The specific pressure inputs, correlated by well depth, are defined in the FG-PP Inputs table and are usually provided by teams working in the subsurface domain. Finally, the sequence of the MASP calculation itself is defined by the following tables: MASP Calc. Step 1, MASP Calc. Step 2a, and MASP Calc. Step 2b. In addition to the computations, these tables include attributes for the appropriate calculation scenarios (such as the ratio of mud or gas in the well).

3.1.2 Schema B

The signature (sorts and symbols) of schema B are defined in Figure 5. The equations for schema B are defined in Figure 6.

Schema B follows a similar pattern of information as described by schema A because they shared the same MASP sheet as a starting point. However, because the input models were independently interpreted and represented as ologs there are key differences. Much of the disparity was driven by how the engineers chose to approach the original MASP calculation sheet. Schema A models five hole sections and began olog conversion from the Header Info table. Schema B, on the other hand, approached the olog conversion from the bottom-up and modeled only one hole section starting from the MASP computation itself.

Schema B defines the high level well information in the Well Data Key table. The particular depths of
forall x: "Header Info", "RKB-ML"(x) = "Water Depth"(x) + "RKB Height"(x)

forall x: "MASP Calc. Step 1", "70% Burst (corrected)"(x) = "70% Burst (not corrected)"(x) - "Corrected Hydrostatic"(x)

forall x: "MASP Calc. Step 1", "70% Burst (not corrected)"(x) = "Burst Rating"("Casing Section"(x)) * "De-Rated Percent"(x)

forall x: "MASP Calc. Step 1", "EMW - Backup"(x) = "Downhole Mud Weight"("Interval"(x)) - "Backup Pore Pressure"("Zone Name"(x))

forall x: "MASP Calc. Step 1", "Corrected Hydrostatic"(x) = .052 * ("RKB TVD"("RKB TVD"(x)) * "EMW - Backup"(x))

forall x: "MASP Calc. Step 2a", "MASP BHP"(x) = (("Pressure at Deepest OH Depth"(x) - "SW Hydrostatic"(x)) - "Gas Hydrostatic (BHP)"(x)) - "Mud Hydrostatic (BHP)"(x)

forall x: "MASP Calc. Step 2a", "MASP Shoe"(x) = ("Frac Pressure at Deepest Shoe"(x) - "Gas Hydrostatic (Shoe)"(x)) - "Mud Hydrostatic (Shoe)"(x) - "SW Hydrostatic"(x)

forall x: "MASP Calc. Step 2a", "EMW - Backup"(x) = "Downhole Mud Weight"("Interval"(x)) - "Backup Pore Pressure"("Zone Name"(x))

forall x: "MASP Calc. Step 2a", "Corrected Hydrostatic (BHP)"(x) = .052 * "Mud BHP TVD"("Mud BHP TVD"(x)) * "Max Mud Weight at OH Depth (Downhole)"("Interval"(x)))

forall x: "MASP Calc. Step 2a", "Gas Hydrostatic (BHP)"(x) = "Gas Gradient"(x) * "Gas TVD (BHP)"(x)

forall x: "MASP Calc. Step 2a", "SW Hydrostatic"(x) = 0.052 * "Water Depth"("Well"(x)) * "Water Density"("Well"(x))

forall x: "MASP Calc. Step 2a", "Pressure at Deepest OH Depth"(x) = "RKB TVD"("TVD Deepest OH"(x)) - "Max Mud Weight at OH Depth (Downhole)"("Interval"(x))

forall x: "MASP Calc. Step 2a", "Interface TVD (BHP)"(x) = "RKB-ML"("Well"(x)) + "Gas Ratio"(x) * ("RKB TVD"("TVD Deepest OH"(x)) - "RKB-ML"("Well"(x)))

forall x: "MASP Calc. Step 2a", "Mud BHP TVD"("Mud BHP TVD"(x)) - "RKB-ML"("Well"(x))) * "Mud Ratio"(x)

forall x: "MASP Calc. Step 2a", "Gas TVD (BHP)"(x) = ("RKB TVD"("TVD Deepest OH"(x)) - "RKB-ML"("Well"(x))) * "Gas Ratio"(x)

forall x: "MASP Calc. Step 2a", "Frac Pressure at Deepest Shoe"(x) = "RKB TVD"("TVD Shoe"(x)) - "Mud TVD (BHP)"(x) - .052 * "Frac Gradient at Deepest Shoe"("Interval"(x))

forall x: "MASP Calc. Step 2a", "Mud TVD (Shoe)"(x) = MAX(0,("RKB TVD"("TVD Shoe"(x)) - "Gas TVD (Shoe)"(x)) - "RKB-ML"("Well"(x)))

forall x: "MASP Calc. Step 2a", "Gas TVD (Shoe)"(x) = MIN("RKB TVD"("TVD Shoe"(x)) - "RKB-ML"("Well"(x))), "Interface TVD (BHP)"(x) - "RKB-ML"("Well"(x)))

forall x: "MASP Calc. Step 2a", "Minimum MASP"(x) = MIN("MASP BHP"(x),"MASP Shoe"(x))

forall x: "MASP Calc. Step 2b", "MAWP"(x) = "Minimum MASP"("Reference MASP"(x)) + "SW Hydrostatic"("Reference MASP"(x))

forall x: "MASP Calc. Step 2b", "MAWP Surface"(x) = "MAWP"(x) - (.052 * "RKB-ML"("Well"(x))) * "Downhole Mud Weight"("Interval"(x))

forall x: "MASP Calc. Step 2b", "MAWP Surf + 500"(x) = "MAWP Surface"(x) + 500

Figure 4: Equations for Schema A
interest, where the MASP calculation was performed for the lone hole section, are defined by the Item of Interest Key table\textsuperscript{2}. Schema B also captures a series of tables to differentiate information related to the various steps of the MASP calculation: Exposed Shoe Key, OH Key, and Casing Burst Key. The subsurface depth and pressure inputs are defined in the Pore Pressure Frac Pressure Key table, and the mud/gas scenarios are defined in the Mud Gradient Key table. The MASP computation is decomposed across the following tables: Burst Calculation Key, MASP Open Hole Key, MASP Shoe Key, and MASP Key. Schema B also includes the Exposed Shoe Key table, which informs the fracture gradient variables in the MASP shoe calculation. The different MASP uses (open hole and shoe) are separated in Schema B for transparency. Finally, schema B contains tables that contain a few data points that serve more as an efficient modeling structure to map keys between tables (Casing Section Key and Casing Key); this approach was taken to ensure each critical “variable” received its own table to better inform future auditing of results.

3.1.3 Overlap Schema and Mappings

The signature (sorts and symbols) of schema O, the overlap schema, are defined in Figure 7, along with its equations. The overlap schema is inspired by the English definition of the MASP calculation included with the original Excel sheet, shown in Figure 11.

The three equations in the overlap schema generate the verification conditions described in section 4; our methodology checks that the equations in the overlap schema are entailed by the equations of each source schema. In this way, we need not appeal to the authors of the source schemas; we can check if the equations (semantics) we desire (in the overlap) are already present in the source schemas. It is for this reason we describe our methodology as “consensus-free”.

The schema mappings from the overlap schema to source A’s schema and source B’s schema are shown in Figures 8 and 9 respectively. As required by algebraic data integration, for each overlap table, we give a table in source A (or B, respectively), and for each overlap column, we give an expression in source A (or B, respectively). Our data sources contain typos in their original schemas and data which are reflected in our development.
forall x: "Casing Key", x."Max Pore Pressure at Hole Section TD (ppg)" = x."PPFP Key"."Pore Pressure_mid (ppg)"
forall x: "Item of Interest Key", x."Item of Interest Backup Pore Pressure (ppg)" = x."PPFP Key"."Pore Pressure_mid (ppg)"
forall x: "OH Key", x."Pore Pressure at OH Depth (ppg)" = x."PPFP Key"."Pore Pressure_mid (ppg)"
forall x: "Burst Calculation Key", x."Burst Rating" = x."Item of Interest"."Item of Interest - Burst Rating (psi)"
forall x: "Burst Calculation Key", x."Backside EMW" = x."Item of Interest"."Item of Interest Backup Pore Pressure (ppg)"
forall x: "Burst Calculation Key", x."TVD" = x."Item of Interest"."Item of Interest Depth - RKB TVD (Ft)"
forall x: "Burst Calculation Key", x."Burst Rating Corrected for MW & Backside" = (x."Material Utilization Factor" * x."Burst Rating") - (x."DHEMW" - x."Backside EMW" * x.Constant * x.TVD)
forall x: "MASP Open Hole Key", x."Pore Pressure at OH Depth (ppg)" = x."OH Key"."Pore Pressure at OH Depth (ppg)"
forall x: "MASP Open Hole Key", x."MW" = x."OH Key"."Max Mud Weight at OH Depth (Downhole,ppg)"
forall x: "MASP Open Hole Key", x."HC Grad." = x."OH Key"."Kick Fluid Gradient if Gas per BSEE (psi/ft)"
forall x: "MASP Open Hole Key", x."MASP Open Hole" = -(-((-x."Open Hole Depth yielding highest MASP- RKB TVD (Ft)"
* x."Pore Pressure at OH Depth (ppg)" * x.Constant),
x.TVDmud * x.MW * x.Constant), x.TVDHC * x."HC Grad."), x."Sw Hydrostatic")
forall x: "MASP Open Hole Key", x."TVDMud" = MAX(0, *(x."Open Hole Depth yielding highest MASP- RKB TVD (Ft)", x."Casing Section Key"."Well Data Key"."RKB-ML"), x."Mud Gradient Key"."Mud Fraction"))
forall x: "MASP Open Hole Key", x."TVDHHC" = MAX(0, *(x."Open Hole Depth yielding highest MASP- RKB TVD (Ft)", x."Casing Section Key"."Well Data Key"."RKB-ML"), x."Mud Gradient Key"."Gas Fraction"))
forall x: "MASP Shoe Key", x."MASP Shoe" = -(-(-(x."Deepest Exposed Shoe Below This Shoe - RKB TVD (Ft)"
* x."Frac Gradient at Deepest Exposed Shoe (ppg)" * x.Constant),
x.TVDmud * x.MW * x.Constant, x.TVDHC * x."HC Grad."), x."Sw Hydrostatic")
forall x: "MASP Shoe Key", x."MW" = x."OH Key"."Max Mud Weight at OH Depth (Downhole,ppg)"
forall x: "MASP Shoe Key", x."Frac Gradient at Deepest Exposed Shoe (ppg)" = x."Exposed Shoe Key"."Frac Gradient at Deepest Exposed Shoe (ppg)"
forall x: "MASP Key", x.MASP = MIN(x."MASP Openhole", x."MASP Shoe")
forall x: "MASP Key", x."MASP Openhole" = x."MASP Open Hole Key"."MASP Open Hole"
forall x: "MASP Shoe Key", x."MASP Shoe" = x."MASP Shoe Key"."MASP Shoe"
forall x: "Exposed Shoe Key", x."Frac Gradient at Deepest Exposed Shoe (ppg)" = x."PPFP Key"."Salt FG (Using OBG + 1000 psi (ppg)"
forall x: "Burst Calculation Key", x.Constant=.052

Figure 6: Equations for Schema B

forall i:Step1, casingburst(i) = (i.seventy * i.pb) - i.tvd * i.pointohfivetwo * (i.pmud - i.ppore)
forall i:Step2, maspbhp (i) = i.bhp - (i.mhs + i.ghs + i.swhs)
forall i:Step2b, maspshoe(i) = i.bhp - (i.mhs + i.ghs + i.swhs)

Figure 7: Signature and Equations for Overlap Schema
Step1 -> "MASP Calc. Step 1"  
pointohfivetwo -> lambda x, .052  
pnud -> lambda x, x."Interval"."Downhole Mud Weight"  
ppore -> lambda x, x."Zone Name"."Backup Pore Pressure"  
casingburst -> lambda x, x."70% Burst (corrected)"  
seventy -> lambda x, x."De-Rated Percent"  

castingburst -> lambda x, x."Zone Name"."Backup Pore Pressure"  

Step2 -> "MASP Calc. Step 2a"  
bhp -> lambda x, x."Pressure at Deepest OH Depth"  
mhs -> lambda x, x."Mud Hydrostatic (BHP)"  
sgs -> lambda x, x."Gas Hydrostatic (BHP)"  
swhs -> lambda x, x."SW Hydrostatic"  
maspbhp -> lambda x, x."MASP BHP"  

Step2b -> "MASP Calc. Step 2b"  
bhp -> lambda x, x."Frac Pressure at Deepest Shoe"  
mhs -> lambda x, x."Mud Hydrostatic (Shoe)"  
sgs -> lambda x, x."Gas Hydrostatic (Shoe)"  
swhs -> lambda x, x."SW Hydrostatic"  
maspshoe -> lambda x, x."MASP Shoe"  

WellInfo -> "Header Info"  
well -> lambda x, x."Well Name"  
rkb -> lambda x, x."RKB-ML"  
depth -> lambda x, x."WD"  
field -> lambda x, x."Field"  

Figure 8: Schema Mapping from Overlap to A

Figure 9: Schema Mapping from Overlap to B
forall bc:"MASP Calc. Step 1", bc."Casing Section" = bc."Zone Name"."Casing Section"
forall bc:"MASP Calc. Step 1", bc."Casing Section".Interval = bc.Interval
forall bc:"MASP Calc. Step 1", bc.Interval.Well = bc."RKB TVD".Well
forall bc:"MASP Calc. Step 1", bc."Casing Section"."Total Vertical Depth".Well = bc."Zone Name"."RKB TVD".Well
forall bc:"MASP Calc. Step 1", bc.Interval."Planned Section Total Depth".Well = bc."Zone Name"."RKB TVD".Well

forall x:"MASP Calc. Step 2a", x.Well = x."TVD Shoe".Well
forall x:"MASP Calc. Step 2a", x.Well = x."TVD Deepest OH".Well
forall x:"MASP Calc. Step 2a", x.Well = x."Interval"."Planned Section Total Depth".Well

forall x:"Item of Interest Key", x."Liner Hangoff Key"."Top Casing Section / Nominal Casing - Size" =
  x."Top Casing Section / Nominal Casing - Size"
for x:"MASP Key", x."Casing Section Key" = x."MASP Open Hole Key"."Casing Section Key"
for x:"MASP Key", x."Casing Section Key" = x."MASP Open Hole Key"."Mud Gradient Key"."Casing Section Key"
forall x:"MASP Key", x."Casing Section Key" =
  x."MASP Open Hole Key"."OH Key"."Top Casing Section / Nominal Casing - Size"
forall x:"MASP Key", x."MASP Open Hole Key"."Mud Gradient Key"."Casing Section Key" =
  x."MASP Shoe Key"."Casing Section Key"
forall x:"MASP Key", x."MASP Shoe Key"."Exposed Shoe Key"."Top Casing Section / Nominal Casing - Size" =
  x."MASP Shoe Key"."Casing Section Key"
forall x:"MASP Key", x."MASP Open Hole Key"."OH Key"."Top Casing Section / Nominal Casing - Size" =
  x."MASP Shoe Key"."Casing Section Key"

forall x:"Interval Info", x.Interval.Well."Well Name" =
  x."Item of Interest"."Top Casing Section / Nominal Casing - Size"."Well Data Key"."Well"
forall x:"MASP Calc. Step 2a", x."MASP Open Hole Key"."Mud Gradient Key"."Casing Section Key"."Well Data Key" =
  x."MASP Shoe Key"."OH Key"."Top Casing Section / Nominal Casing - Size"."Well Data Key"
forall x:"MASP Calc. Step 1", x."Item of Interest"."Liner Hangoff Key"."Top Casing Section / Nominal Casing - Size".
  "Well Data Key" = x."Casing Section"."Interval".Well
forall x:"MASP Calc. Step 1", x."Item of Interest"."Liner Hangoff Key"."Top Casing Section / Nominal Casing - Size".
  "Well Data Key" = x."Zone Name"."Casing Section"."Interval".Well
forall x:"MASP Calc. Step 1", x."Item of Interest"."Liner Hangoff Key"."Top Casing Section / Nominal Casing - Size".
  "Well Data Key" = x."Interval".Well
forall x:"MASP Calc. Step 1", x."Item of Interest"."Liner Hangoff Key"."Top Casing Section / Nominal Casing - Size".
  "Well Data Key" = x."Zone Name"."Casing Section"."Interval".Well
forall x:"MASP Calc. Step 1", x."Item of Interest"."Liner Hangoff Key"."Top Casing Section / Nominal Casing - Size".
  "Well Data Key" = x."Interval".Well
forall x:"MASP Calc. Step 1", x."Item of Interest"."Liner Hangoff Key"."Top Casing Section / Nominal Casing - Size".
  "Well Data Key" = x."Zone Name"."Casing Section"."Interval".Well
forall x:"MASP Calc. Step 2a", x.Interval.Well =
  x."MASP Open Hole Key"."Mud Gradient Key"."Casing Section Key"."Well Data Key"
forall x:"MASP Calc. Step 2a", x.Well =
  x."MASP Open Hole Key"."Mud Gradient Key"."Casing Section Key"."Well Data Key"
forall x:"Item of Interest Key", x."Top Casing Section / Nominal Casing - Size"."Well Data Key" =
  x."Liner Hangoff Key"."Top Casing Section / Nominal Casing - Size"."Well Data Key"

Figure 10: Additional Equations for Better Data Integration
The four tables that make up the overlap schema are defined as follows. The WellInfo table captures general information about the subject well to allow the input models to integrate data for calculations being performed on the same well. The first step of the MASP calculation, defined in the Step1 table, represents the de-rated burst pressure rating of the subject casing strings after being corrected for mud weight and pore pressure backup values at the depth of interest. This is a requirement based on government regulations, which stipulate that the subject casing string should be de-rated to 70 percent of the casing burst pressure rating. The next requirement in the MASP calculation determines the maximum anticipated surface pressure. This is performed for both bottomhole pressure and fracture gradient scenarios and is defined by the Step2 and Step2b tables, respectively. The mappings from this overlap model to schema A and schema B define columns where each component of the overlap computation is contained within the input models.

These tables were chosen as overlap, and the schema mappings designed as they are, because the overlapping tables represent engineering requirements shared by many business units and engineering teams around the world. Mathematically verifying that an expression of these requirements is entailed by the equations in each source schema opens a door to new forms of well design assurance and improving the quality and cycle time of well engineering activities, as discussed in the conclusion.

### 3.2 Suggesting rules

For large spreadsheets, it is impractical for users to start from a blank screen and simply type hundreds or thousands of rules to connect them. Therefore data integration tools, including ours, provide heuristic suggestions as a starting point. Rules that Well Names are preserved as foreign keys are traversed are good examples of rules that are easy to guess based on column names and on data. The paper [?] describes some heuristic rule suggestion algorithms for our particular algebraic data model, and many traditional heuristic algorithms [8] can be re-used off-the-shelf in our system.
3.2.1 Result

The resulting schema of schema integration can be easily described: we start by taking the disjoint union $S_A + S_B$ of the source schemas; then, for every table $o$ in the overlap schema $O$, we merge tables $M_A(o)$ and $M_B(o)$ and add equations $M_A(f) = M_B(f)$ for every $f$ with domain $o$, ultimately resulting in a “colimit” [11] schema (theory) denoted $S_A + O.M_A.M_B S_B$. This schema is large and somewhat redundant in our case study because we are only merging three tables (step 1, step 2a, and the header/well info), so we do not display it here. However, in Figure 10 we display additional equations that we added to the colimit schema to aid data quality during data integration—arguably, some of these equations were forgotten in the original schemas. Additionally, the integrated schema can be read off the integrated data from the full result in the appendix.

**Remark.** The result of schema integration is only defined up to isomorphism, and so for example the names of tables in the result, and even the number of columns and equations in $S$, is not determined uniquely—the CQL software has many options to allow users to select which resulting schema (theory) they want from among many choices. In fact, under mild assumptions it is possible to prove that there is no “best” table and column naming scheme. In practice, the integrated schema is often customized with user-specific naming of tables and columns.

3.3 Data Integration and Exchange

We continue to follow the algebraic data integration pattern from Figure 2 and in this section define the data sources and their overlap that complete the pattern started in the previous section. We now walk through the data contained in each table below.

3.3.1 Data A

Input data for source A is shown in Figure 12.

- **Casing Section Table**
  - Multiple casing intervals were included (e.g. 22”, 16”, etc).
  - Question marks in the Casing Weight column reflect missing values in the original Excel MASP spreadsheet.

- **FG-PP Inputs Table**
  - Over 4,000 rows were originally included in this table’s sheet, but not all are imported into CQL for expediency.
  - This table represents geologic inputs from cross-domain stakeholders who govern the reservoir characteristics and Earth model.

- **Header Info Table**
  - Only one well was modeled, so there is only one row.

- **Interval Info Table**
  - Five rows reflect the casing/hole sections that were included.
  - The question mark in the table reflects a missing fracture gradient value in the original Excel MASP spreadsheet.

- **MASP Calc. Step 1 Table**
  - The first criteria for calculating MASP is based on government regulations and includes calculating 70 percent of the burst pressure rating for each casing string.
  - This step also considers the burst pressure rating of the subject casing string when corrected to mud weight and pore pressure backup at various depths or zones of interest.
## Figure 12: Data for Schema A

| Data Source | Running Time | Number of Events | Number of Rules | Number of Attributes | Number of Classes | Accuracy | Precision | Recall |
|-------------|--------------|------------------|-----------------|---------------------|------------------|----------|-----------|--------|
| Transaction 1 | 1237.56 | 87.57 | 56.24 | 34.56 | 23.45 | 98.9% | 97.8% | 96.5% |
| Transaction 2 | 1324.65 | 78.90 | 45.67 | 23.45 | 12.34 | 99.9% | 98.7% | 97.5% |
| Transaction 3 | 1456.78 | 67.89 | 34.56 | 23.45 | 10.98 | 99.8% | 98.6% | 97.4% |

Note: The data represents the performance metrics of various data mining models applied to different transactions. Each row corresponds to a different data source, with columns indicating running time, number of events, number of rules, number of attributes, number of classes, accuracy, precision, and recall.
The input data for source B is shown in Figure 13. B’s data includes the same source data as A, except the MASP calculation is only performed on a single casing section (as opposed to five). As we will see later, this is the primary reason why Data B will be extended significantly in the case study’s resulting data exchange, but Data A will be unchanged. Further, both Data A and Data B include ancillary columns from the source sheet that were not used in the MASP calculation, resulting in additional differences at the data level. We now walk through the data contained in each table below.

**Figure 13: Data for Schema B**

- **MASP Calc. Step 2a Table**
  - Five rows reflect the MASP result (for each casing section) based on bottom hole pressure, which is calculated using the deepest open hole depth.

- **MASP Calc. Step 2b Table**
  - Five rows reflect the MASP result (for each casing section) based on the fracture gradient, which is calculated using the depth of the deepest exposed casing string (known as the casing shoe).

- **Zone of Interest Table**
  - This table reflects key attributes associated with each casing string, the components that make up that casing string, and the associated geological characteristics at a particular depth.

### 3.3.2 Data B
• Burst Calculation Key Table
  – This table reflects the depths of interest considered for the MASP calculation of the 22" casing section.
  – Key variables for step 1 of the MASP calculation (i.e., 70 percent burst pressure rating of the casing, de-rated and corrected for backup) are also included.

• Casing Burst Key Table
  – This table contains casing specifications for the 22" hole section.
  – Source B captured this data from a separate tab in the original Excel MASP sheet, whereas Source A did not use this table at all for casing data.
  – Question marks represent missing values in the original Excel MASP sheet.

• Casing Key Table
  – Only one well was modeled, hence there is only one row.

• Casing Section Key Table
  – This table contains a basic description of the 22" surface casing.

• Exposed Shoe Key Table
  – This table informs the fracture gradient at the deepest exposed shoe, which is a key input for the MASP shoe calculation.

• Item of Interest [sic] Table
  – Similar to the Burst Calculation Key table, this data reflects the depths of interest where the MASP calculations were performed.

• Liner Hangoff Key Table
  – Data for this table was included in anticipation for MASP calculation steps beyond the scope of the case study. So while the data is not used for step 1 or step 2 for MASP as defined in Figure 11, it is typically used in subsequent steps of the casing design analysis.

• MASP Key Table
  – This table contains a summary of results from the MASP open hole and MASP shoe calculations.

• MASP Open Hole Key Table
  – This table contains the key inputs and calculations for the MASP open hole step of the analysis.

• MASP Shoe Key Table
  – This table contains the key inputs and calculations for the MASP shoe step of the analysis.

• Mud Gradient Key Table
  – This table is informs the fluid column scenario for the MASP calculation (i.e., the ratio of gas-to-mud in the wellbore).

• OH Key Table
forall x y : "MASP Calc. Step 1", x."70% Burst (corrected)" = y."Burst Rating Corrected for MW & Backside" -> x = y
forall x y : "MASP Calc. Step 2a", x."MASP BHP" = y."MASP Openhole" -> x = y
forall x y : "Header Info", x."Well Name" = y."Well Name" -> x = y
forall x y : "Header Info", x."Well" = y."Well" -> x = y

Figure 14: Row merge rules

- This table contains key information for the MASP open hole scenario, such as pore pressures and mud weights.

- **Pore Pressure Frac Pressure Key Table**
  - The original source table contains over 4,000 rows of data, but not all are imported into CQL for expediency.
  - This table represents geologic inputs from cross-domain stakeholders who govern the reservoir characteristics and earth model.

- **Shoe Track Key Table**
  - This table contains information for MASP shoe scenario, specifically the fracture gradient at the shoe depth.

- **Well Data Key Table**
  - Only one well is modeled, so there is a single row of data reflecting the general well header information.

### 3.3.3 Overlap Data and Mappings

Although the algebraic data integration design pattern requires an overlap instance $I_O$ on the overlap schema $S_O$, it can be unfeasible to generate such instances by hand, because they are often large and difficult to describe as models. So, instead we describe rules—Horn clauses; implications of equations—that implicitly specify the overlap, including the required data mappings out of the overlap data. Figure 14, states that step 1 rows, now coming from both source A and source B, should be merged when they agree on two particular columns, and similarly for step 1a. If such a (recursive) merge is contradictory, CQL reports this, as described in Section 4.

**Remark.** Not all sets of Horn clauses (implications of equations) induce an overlap instance and two data mappings as required for algebraic integration. If we were to follow the algebraic integration pattern precisely [11], we would instead give a “query” of a specific form that materializes the matching pairs of source A rows and source B rows as an overlap instance. Such a query is a re-statement of our Horn clauses.

**Remark.** As an alternative to thinking of row merge rules as specifying an overlap instance for which a colimit is taken, we can also think of them as specifying a “lifting problem” that we are solving. If we do this, as occurs often in practice, then we can generalize the rule language to include existential quantifiers on the right-hand side of rules [11], dramatically increasing their expressive power.

### 3.3.4 Result

The semantics of the integrated result in algebraic data integration is given by taking a colimit of the input databases, each suitably pushed forward onto the colimit schema (plus additional user-defined equations, shown in Figure 10) via “left Kan extension” [11]. However, because this integrated result subsumes both sheets, it is too large to conveniently display here (we do so in the appendix). So instead, we will focus on the data exchange between the source sheets. That is, we project the integrated result back on to each source and analyze the difference with the original source. When we do this on this example, we find that data A is unchanged, but data B is extended significantly. Sometimes, the extensions to data B are with concrete values, such as in the Burst Calculation Key Table, but other times the extensions are with blank
| No. | Bubble BPM | Burst Rating | Burst Rating Corrected for Wil. - Constant | SNIMR | Material & Mission Factor | FID | Item of Interest |
|-----|-----------|--------------|-------------------------------------------|-------|--------------------------|-----|-----------------|
| 01  | 0.668     | 4275         | 0.052                                      | 0.052 | 0.052                    | 0.052 | 0.052           |
| 02  | 0.6       | 3500         | 0.052                                      | 0.052 | 0.052                    | 0.052 | 0.052           |
| 03  | 0.6       | 20090        | 0.052                                      | 0.052 | 0.052                    | 0.052 | 0.052           |
| 04  | 0.6       | 15752        | 0.052                                      | 0.052 | 0.052                    | 0.052 | 0.052           |
| 05  | 9.7       | 16890        | 0.052                                      | 0.052 | 0.052                    | 0.052 | 0.052           |
| 06  | 13.419    | 18150        | 0.052                                      | 0.052 | 0.052                    | 0.052 | 0.052           |
| 07  | 7.35      | 18100        | 0.052                                      | 0.052 | 0.052                    | 0.052 | 0.052           |
| 08  | 12.937    | 18700        | 0.052                                      | 0.052 | 0.052                    | 0.052 | 0.052           |
| 09  | 15.935    | 18550        | 0.052                                      | 0.052 | 0.052                    | 0.052 | 0.052           |
| 10  | 13.415    | 18350        | 0.052                                      | 0.052 | 0.052                    | 0.052 | 0.052           |
| 11  | 16.256    | 18550        | 0.052                                      | 0.052 | 0.052                    | 0.052 | 0.052           |
| 12  | 12.415    | 18500        | 0.052                                      | 0.052 | 0.052                    | 0.052 | 0.052           |
| 13  | 16.256    | 18550        | 0.052                                      | 0.052 | 0.052                    | 0.052 | 0.052           |
| 14  | 11.482    | 18500        | 0.052                                      | 0.052 | 0.052                    | 0.052 | 0.052           |
| 15  | 15.935    | 18700        | 0.052                                      | 0.052 | 0.052                    | 0.052 | 0.052           |
| 16  | 12.937    | 18700        | 0.052                                      | 0.052 | 0.052                    | 0.052 | 0.052           |
| 17  | 13.415    | 18350        | 0.052                                      | 0.052 | 0.052                    | 0.052 | 0.052           |
| 18  | 16.256    | 18550        | 0.052                                      | 0.052 | 0.052                    | 0.052 | 0.052           |
| 19  | 13.415    | 18350        | 0.052                                      | 0.052 | 0.052                    | 0.052 | 0.052           |
| 20  | 16.256    | 18550        | 0.052                                      | 0.052 | 0.052                    | 0.052 | 0.052           |
| 21  | 11.482    | 18500        | 0.052                                      | 0.052 | 0.052                    | 0.052 | 0.052           |
| 22  | 15.935    | 18700        | 0.052                                      | 0.052 | 0.052                    | 0.052 | 0.052           |
| 23  | 12.937    | 18700        | 0.052                                      | 0.052 | 0.052                    | 0.052 | 0.052           |
| 24  | 13.415    | 18350        | 0.052                                      | 0.052 | 0.052                    | 0.052 | 0.052           |
| 25  | 16.256    | 18550        | 0.052                                      | 0.052 | 0.052                    | 0.052 | 0.052           |
| 26  | 15.935    | 18700        | 0.052                                      | 0.052 | 0.052                    | 0.052 | 0.052           |
| 27  | 12.937    | 18700        | 0.052                                      | 0.052 | 0.052                    | 0.052 | 0.052           |
| 28  | 16.256    | 18550        | 0.052                                      | 0.052 | 0.052                    | 0.052 | 0.052           |
| 29  | 13.415    | 18350        | 0.052                                      | 0.052 | 0.052                    | 0.052 | 0.052           |
| 30  | 15.935    | 18700        | 0.052                                      | 0.052 | 0.052                    | 0.052 | 0.052           |

**Casing Burst Key (KB)**

| No. | Casing Type | Nominal Value | Lowest End Collapse | Plan Body for 1375 End | Plan Body for 1375 End Collapse | Size | Weight (lb/ft) |
|-----|-------------|---------------|----------------------|------------------------|----------------------------------|------|---------------|
| 31  | 13.415       | 18350         | 0.052                | 142.08                  | 3600                             | 0.052 | 0.052         |
| 32  | 16.256       | 18550         | 0.052                | 142.08                  | 3600                             | 0.052 | 0.052         |
| 33  | 11.482       | 18500         | 0.052                | 142.08                  | 3600                             | 0.052 | 0.052         |
| 34  | 15.935       | 18700         | 0.052                | 142.08                  | 3600                             | 0.052 | 0.052         |

**Casing Section Key (CS)**

| No. | Casing Type | Inner String | Hole Section | Max. Pressure at Hole Section | Nominal Weight Casing Run In End | Nominal Weight Casing Run In | Item of Interest | FPD | Top Casing Section / Nominal - Normal | Non Data Key |
|-----|-------------|--------------|--------------|-------------------------------|----------------------------------|-------------------------------|-----------------|-----|-------------------------------------|-------------|
| 35  | 16.256      | 18550        | 0.052        | 142.08                        | 3600                             | 0.052                         | 0.052           | FPD | 12.5                                 | 2.5         |

Figure 15: Data Exchange of Source A to Source B, 1 of 3
values (labelled nulls), indicating that the data B is incomplete with respect to source A as defined by the overlap and mappings. Intuitively, this phenomenon happens because data B is in some sense contained source A, which has more rows to begin with. The “round-tripped” B data is displayed in Figures 15 and 16. The source owners and subject matter experts made the following observations about each table after the data exchange:

- **Burst Calculation Key Table**
  - Whereas Source B started with six rows in this table, the data exchange with Source A through the integrated schema results in 30 rows of data. This is due to Source B sharing six overlapping rows with Source A (these were merged together) while receiving 24 rows as fresh information gain.
  - The specific rows being merged were based on the 70 percent burst rating (corrected) values for each source.
  - Filled-in values can be traced primarily to the MASP Calc. Step 1 table from Source A, but includes other information implied by myriad equations and mappings.

- **Casing Burst Key Table**
  - This table is primarily associated with the Casing Section table from Source A. This is observed by the 29 rows in both tables.
  - The questions marks are due to the fact that Source A did not include any information about these columns (such as Lowest Value Collapse). However, it was observed while creating rules that the Source B Size column did not find a match in Source A because of a difference in data type. This casing size data was represented as a string in Source A and as a float in Source B, in particular, where source B writes 22 source A writes 22". To merge along such attributes, we can...
add float to string and string concatenation functions to our ambient “theory of excel” and then write an equation using this new primitives (e.g. 22"=concat(floatToString(22),".")

– From Source A’s perspective, this table’s columns are not necessary in the MASP calculation, so they are largely excluded from data exchange. Instead, source B includes these values because source B’s data was pulled from a separate dataset containing casing specification information.

Casing Key Table

– Most of this data is not used for the MASP calculation and is in Source B and not Source A, explaining why there is only one row for Source B.

• Casing Section Key Table

– Source B only models one string of casing (the 22” casing section), which is reflected in the fully populated row. The remaining rows are from Source A and reflect the additional casing sections/components that were transferred from the original MASP sheet.

• Exposed Shoe Key Table

– Source B gains four rows of data because the MASP calculation was performed for five hole sections for Source A.

• Item of Interest Key Table [sic]

– Like the Burst Calculation Key table, the Item of Interest Key table began with six rows of data and, through the data exchange with Source A, resulting in an additional 24 rows of fresh data. However, in this case, there are additional rules in place to infer data values for two additional columns (Item of Interest - Burst Rating (psi) and Item of Interest Depth - RKB TVD (Ft)).

• Liner Hangoff Key Table

– This table is not used for the MASP calculation, and the only row of data comes from Source B.

• MASP Key, MASP Open Hole Key, MASP Shoe Key, and Mud Gradient Key Tables

– For these tables, Source B gains four rows of data each because the MASP calculation is performed for five hole sections for Source A.

• OH Key Table

– Most of the data from this table is not used for the MASP calculation, so there are minimal opportunities to create merge rules.

• Pore Pressure Frac Pressure Key Table

– This table matches with the sources on six rows because six pore pressure / fracture gradient inputs were used in the 22” casing section MASP calculated by Source B. Because this data is shared by the subsurface discipline, and Source A, and Source B, the entire table could, in principle, be integrated with additional row merge rules in place.

• Shoe Track Key Table

– Most of this table is not used for the MASP calculation, and the only row of data comes from Source B.

• Well Data Key Table

– There is only one row in this table because the data is merged based on the well name. This ensures both Source A and Source B are performing the MASP calculation for the same well.
3.4 Discussion

Our main result from the case study is that two engineers were able to conduct the same engineering analysis (without ever communicating with each other) and compose their individual models to create an integrated perspective. Specifically, two engineers performed a MASP calculation, which is a key aspect of well construction and is duplicated on every well around the world in a variety of fashions. The integrated result guarantees that individual ologs and integrated ologs are compliant with components of the MASP calculation technical requirement—without pre-coordination and consensus. As a bonus, we also obtain a semantics-preserving information-exchange between the source sheets.

Engineering teams largely go about integration and exchange today by treating subject matter experts as the personification of requirements, which are originally written in natural language as text inside documents that we call global standards or regulations. The current mainstream approach taken by industry is to try to organize these requirements by extracting them from documents and placing them into more sophisticated management systems (i.e., a database where each requirement is an object, and a user interface includes workflows to create, edit, and approve). This approach relies on the idea that after the requirements are organized in the requirement management system, engineers will be able to convert them into rules that can be incorporated into computable models. The challenge to do this, so far, has been the lack of an adequate language that is expressive enough to capture both the general nature of an engineering requirement and its specific computational nature.

By representing objects and models as ologs, engineers can define the structure in a way that makes sense to them. Those structures can then be translated to a shared space within which these objects reside, as well as methods for their construction and inter-relation. We pick up this line of thought in the conclusion.

3.5 Governance

Left as open questions are the authorship and/or governance of the overlap ologs required to integrate sheets—either of the original engineers can create (potentially many, depending on the desired integration semantics) such overlaps, as can 3rd parties (and in fact, CQL can automatically suggest overlap between ologs). In fact, at a formal level, managing an overlap between two ologs is “dual” to managing a collection of WHERE clauses in SQL queries that join two databases together (the algebraic integration semantics of Figure 2 being formally dual to SQL’s join semantics as well). As such, governance of overlaps is too large of a topic to go into here. Regardless of authorship, semantics-preservation is guaranteed without requiring consensus by relying on the formulas in the sheets to fully define semantics.

3.6 Generalization to Multi-Model Merge

In this paper we have focused on an olog merge of the form \( \cdot \leftarrow \cdot \rightarrow \cdot \), where the center dot is the overlap olog and the other two dots are the source olog. However, all of the theory and algorithms in this paper generalize to diagrams of any shape [11], such as

\[ \cdot \quad \longrightarrow \quad \cdot \]

\[ \cdot \quad \longrightarrow \quad \cdot \]

\[ \cdot \quad \leftarrow \quad \cdot \]

\[ \cdot \quad \leftarrow \quad \cdot \]

It is easy to see that an integration of the above form requires only logarithmically many mappings in the number of nodes, rather than quadratically many that would be required if all nodes were to be mapped to all nodes. It is because our methodology can merge any shape of olog mappings, such as the above, that we say it does not require every source to be related to every source.

4 Generating and Checking Verification Conditions

This section will be of interest primarily to computer scientists.
4.1 Functoriality Conditions

In this section we describe how to generate and solve the verification conditions required to ensure the semantic consistency of the source schemas with respect to the overlap schema. The verification conditions associated to schema mapping \( M_A : S_O \rightarrow S_A \) ensure/are that \( S_A \) proves \( M_A(p) = M_A(q) \) for every equation \( p = q \) of \( S_O \), and similarly for \( M_B \). This establishes that \( M_A \) and \( M_B \) are functors and hence that we can proceed with algebraic data integration. In Figure 18 we display these verification conditions for \( M_A \) and \( M_B \) in the Coq proof assistant \([7]\) where they can be proved by humans. The CQL tool also emits the verification conditions in TPTP form \([3]\) so that they can be input to a variety of automated theorem provers \([6]\). It took one of the authors approximately four hours to prove all the generated conditions in this paper in Coq with elementary methods, using a handful of basic assumptions about e.g. commutativity of addition, but with no domain expertise. Although checking functoriality is undecidable in general, the conditions can always be decidably checked for a particular model, providing a slightly weaker but still useful guarantee.

4.2 Conservativity/Consistency Conditions

In this section we describe how to generate and solve the verification conditions required to ensure the semantic consistency of the integrated result olog. The integrated result is a quotient of a coproduct of input databases; to compute the quotient, a step-by-step process is used, where at each step a new olog is constructed from the previous one (starting from the coproduct of the inputs and terminating on the integrated result). At each step of the process an equational theory representing the result is modified by adding additional equations and/or symbols \([11]\). For example, at one step we might add \( Alice.age = 20 \) and at another step add \( Bob.age = 30 \) and at another step add \( Alice = Bob \), from which it follows that \( 20 = 30 \) — a contradiction, or rather, a “non-conservative extension” of our original definition of numbers, for which \( 20 \) and \( 30 \) are not equal (number systems in which \( 20 = 30 \) are found in e.g. modular arithmetic and can in fact be represented by ologs). At every step of the integration computation, CQL checks that there are no equations that reduce to \( c_1 = c_2 \) for distinct numerals, strings, etc under the usual reduction rules for arithmetic such as \( 1+1 \rightarrow 2 \) etc, providing significant defense against contradictions. In general, contradiction detection is undecidable but definitional/free spreadsheets (see section 2) are always contradiction free and CQL can check ologs for freeness, providing more protection still.

5 Conclusion and Applications

Our case study is an example of a non-human-consensus-based method for merging engineering models, built on the idea that formality drives consensus in the semantic sense even if the people involved never meet. Although consensus-free model merge methods are required for scalability beyond small numbers of models, model merge capability itself does not directly suggest a use case and/or drive business value, and so we conclude by describing a particular enterprise use case and then with related work.

5.1 An Example Use Case

Our use case is to certify that an (equational) engineering model complies with (equational) requirements, a role today performed by human auditors working with English text and/or spreadsheets. We focus on this use case because it is the smallest use case that we can think of that is captured by any part of this paper’s case study, but as we will see it can still have significant impact. In particular, if we have \( R \) (equational) requirements to certify on \( W \) (equational) models each year at a cost of \( C \) per certification then the direct cost savings of this use case is \( T \times W \times C \) annually, a number that can immediately be seen to be \( 8+ \) figures in large enterprises when there are hundreds of thousands of (equational) requirements to check each year.

To implement the use case we require two olog schemas, \( S \) and \( T \), and a schema mapping \( F : S \rightarrow T \). For example, we may let \( S \) be this paper’s overlap schema \( S_O \) (Figure 7) and let \( T \) be this paper’s source schema \( A \), namely, \( S_A \) (Figure 3) (or source schema B, namely, \( S_B \), Figure 5) and let \( F \) be this paper’s schema mapping \( S_O \rightarrow S_A \), shown in Figure 8 (resp. Figure 9 for source B). The core of the use case is to check if \( F \) denotes a functor, i.e., check if \( F \) translates the equations of \( S \) into \( T \) in a way that preserves provability as described in Section 4. If so, then we say that \( T \) complies with the requirements of \( S \) as viewed by \( F \).
//Schema A
Parameters String Float : Type.
Parameters plus times : Float * Float -> Float.

Parameters CasingSection ZoneofInterest MASPCalcStep1 : Set.

Parameter HeaderInfo_TypeofWell : HeaderInfo -> String.
Parameter HeaderInfo_RigName : HeaderInfo -> String.
Parameter HeaderInfo_Field : HeaderInfo -> String.

Axiom ax0 : forall (x : HeaderInfo), HeaderInfo_RKBML(x) = plus(HeaderInfo_WaterDepth(x), HeaderInfo_RKBHeight(x)).
Axiom ax1 : forall (x : MASPCalcStep2a), MASPCalcStep2a_MASPShoe(x) =
  minus(minus(minus(MASPCalcStep2a_FracPressureatDeepestShoe(x), MASPCalcStep2a_MudHydrostaticShoe(x)),
  MASPCalcStep2a_GasHydrostaticShoe(x)), MASPCalcStep2a_SWHydrostatic(x)).
Axiom ax2 : forall (x : MASPCalcStep2a), MASPCalcStep2a_InterfaceTVDBHP(x) =
  plus(HeaderInfo_RKBML(MASPCalcStep2a_Well(x)),
  times(MASPCalcStep2a_GasRatio(x), minus(FGPPInputs_RKBTVD(MASPCalcStep2a_TVDDeepestOH(x)),
  HeaderInfo_RKBML(MASPCalcStep2a_Well(x))))).
Axiom ax3 : forall (x : MASPCalcStep2b), MASPCalcStep2b_MASP(x) =
  plus(MASPCalcStep2b_MinimumMASP(MASPCalcStep2b_ReferenceMASP(x)),
  MASPCalcStep2b_SWHydrostatic(MASPCalcStep2b_ReferenceMASP(x))).

Definition t_Step1_pb : MASPCalcStep1 -> Float := fun x => CasingSection_BurstRating(MASPCalcStep1_CasingSection(x)).
Definition t_Step1_pmud : MASPCalcStep1 -> Float := fun x => IntervalInfo_DownholeMudWeight(MASPCalcStep1_Interval(x)).
Definition t_Step1_ppore : MASPCalcStep1 -> Float := fun x => ZoneofInterest_BackupPorePressure(MASPCalcStep1_ZoneName(x)).

Conjecture conj22 : forall (i : MASPCalcStep1), t_Step1_casingburst(i) =
  minus(times(t_Step1_seventy(i), t_Step1_pb(i)),
  times(t_Step1_tvd(i), times(t_Step1_pointohfivetwo(i),
  minus(t_Step1_pmud(i), t_Step1_ppore(i)))).
Conjecture conj23 : forall (i : MASPCalcStep2a), t_Step2_maspbhp(i) =
  minus(t_Step2_bhp(i), plus(t_Step2_mhs(i), plus(t_Step2_ghs(i), t_Step2_swhs(i)))).

//Schema B
Parameters String Float : Type.
Parameters plus times : Float * Float -> Float.

Parameters WellDataKey CasingKey BurstCalculationKey MASPShoeKey MASPSKey : Set
Parameters WellDataKey_Block : WellDataKey -> String.
Parameters WellDataKey_Well : WellDataKey -> String.
Parameters WellDataKey_WellBore : WellDataKey -> String.

Axiom ax0 : forall (x : CasingKey), CasingKey_MaxPorePressureatHoleSectionTDppg(x) =
  PorePressureFracPressureKey_PorePressuremidppg(CasingKey_PPFPKey(x)).
Axiom ax1 : forall (x : BurstCalculationKey), BurstCalculationKey_TVD(x) =
  ItemofInterestKey_ItemofInterestDepthRKBTVDIt(BurstCalculationKey_ItemofInterest(x))
Axiom ax2 : forall (x : MASPShoeKey), MASPShoeKey_FracGradientatDeepestExposedShoeppg(x) =
  ExposedShoeKey_FracGradientatDeepestExposedShoeppg(MASPShoeKey_ExposedShoeKey(x)).
Axiom ax3 : forall (x : MASPSKey), MASPSKey_MASPOpenhole(x) =
  MASPOpenHoleKey_MASPOpenHole(MASPSKey_MASPOpenHoleKey(x)).

Definition t_Step1_pb : BurstCalculationKey -> Float :=
  fun x => ItemofInterestKey_ItemofInterestBurstRatingpsi(BurstCalculationKey_ItemofInterest(x)).
Definition t_Step1_pmud : BurstCalculationKey -> Float := fun x => BurstCalculationKey_DHew(x).
Definition t_Step1_ppore : BurstCalculationKey -> Float := fun x => BurstCalculationKey_BacksideEMW(x).

Conjecture conj21 : forall (i : MASPSKey), t_Step2b_maspshoe(i) =
  minus(t_Step2b_bhp(i), plus(t_Step2b_mhs(i),
  plus(t_Step2b_ghs(i), t_Step2b_swhs(i)))).
Conjecture conj22 : forall (i : BurstCalculationKey), t_Step1_casingburst(i) =
  minus(times(t_Step1_seventy(i), t_Step1_pb(i)),
  times(t_Step1_tvd(i), times(t_Step1_pointohfivetwo(i),
  minus(t_Step1_pmud(i), t_Step1_ppore(i)))).
Conjecture conj23 : forall (i : MASPSKey), t_Step2_maspbhp(i) =
  minus(t_Step2_bhp(i), plus(t_Step2_mhs(i),
  plus(t_Step2_ghs(i), t_Step2_swhs(i)))).

Figure 18: Coq Verification Conditions for Mapping Functoriality
Optionally, we may additionally take as input an $S$-Instance $X$ and $T$-Instance $Y$ and check that $X = F; Y$, which in this case study corresponds to checking that the equations satisfied by the overlap instance $I_O$ hold in the source instance $I_A$ (resp. in $I_B$) once translated along $F$. That is, our use case can apply to models without data (equations only) as well as to models with data.

The discussion in the preceding paragraph does not describe any method for constructing the inputs to the use case. One option is to simply build up a library of ologs and olog mappings manually, and with composition, such as we did in this paper. Another option is to heuristically generate ologs and mappings from spreadsheets, or even English-text, on-the-fly, to be reviewed/modified by a human— in general, choice will be required among various possible oligifications of a given spreadsheet and among the various possible mappings between ologs. Such heuristics are the subject of future work, but no matter which ologs and mappings are chosen, we can be confident that if the functoriality check succeeds then the (equational) requirements are met. And we are confident that there are many more use cases besides this one waiting to be discovered.

5.2 Related Work

Besides relations to algebraic databases and traditional data integration [11], our algorithm is related to the idea of extending spreadsheets with deductive logic, for example with prolog, an idea whose long history is described in [10]. Indeed, our rules in this paper are expressible in prolog, at least when prolog is understood to include excel’s arithmetic operations, which often it isn’t (it is unclear how much arithmetic is modeled in [10] for example). However, unlike any work in [10], our goal is to connect multiple spreadsheets, not query different parts of the same sheet, which constrains the form of the rules we can write to a subset of prolog, and also constrains the vocabulary we can use within prolog. That is, the algebraic design pattern of Figure 2 says which rules to write, whereas 2 provides no guidance.

Acknowledgements. The authors would like to thank Chevron’s Ben Randell for the support and leadership to conduct this case study, and also Eswaran (Sub) Subrahmanian, Spencer Breiner, and Priyaa Srinivasan of NIST for their help on understanding the relationship between category theory and compositional structures for systems engineering and design. The authors would like to thank Conexus’s Joshua Meyers for help implementing the paper described in this software.

References

[1] http://categoricaldata.net.

[2] https://www.ecfr.gov/current/title-30/chapter-II/subchapter-B/part-250/subpart-G/subject-group-ECFR045ffcd99ad03d3/section-250.737.

[3] http://tptp.cs.miami.edu/TPTP/QuickGuide/Problems.html.

[4] http://www.iadc.org/wp-content/uploads/2017/08/WSP_Definitions_rev1.pdf.

[5] Steve Awodey. Category theory, volume 49 of Oxford Logic Guides. The Clarendon Press Oxford University Press, New York, 2006.

[6] Franz Baader and Tobias Nipkow. Term Rewriting and All That. Cambridge University Press, USA, 1998.

[7] Yves Bertot and P. Castéran. Le Coq’Art. Hardcover, January 2015.

[8] A.H. Doan, A. Halevy, and Z. Ives. Principles of Data Integration. Elsevier Science, 2012.

[9] Till Mossakowski, Ulf Krumnack, and Tom Maibaum. What is a derived signature morphism? In Mihai Codescu, Razvan Diaconescu, and Ionut Tutu, editors, WADT, volume 9463 of Lecture Notes in Computer Science, pages 90–109. Springer, 2014.

[10] António Martins Pedro Ramos1, Luís Botelho. Enhancing excel business tools with additional relational and recursive capabilities.

26
[11] Patrick Schultz and Ryan Wisnesky. Algebraic data integration. *Journal of Functional Programming*, 27:e24, 2017.

[12] David I. Spivak. Functorial data migration. *Inf. Comput.*, 217:31–51, 2012.

[13] David I. Spivak and Robert E. Kent. Ologs: A categorical framework for knowledge representation. *PLOS ONE*, 7(1):1–1, 01 2012.
6 Appendix: Category theory

In this section, we review standard definitions and results from category theory [5]. A quiver, (aka directed multi-graph) \( Q \) consists of a class \( \text{Ob}(Q) \), the members of which we call objects (or nodes), and for all objects \( c_1, c_2 \), a set \( Q(c_1, c_2) \), the members of which we call morphisms (or arrows) from \( c_1 \) to \( c_2 \). We may write \( f : c_1 \to c_2 \) or \( c_1 \to_f c_2 \) instead of \( f \in C(c_1, c_2) \). For an arrow \( f : c_1 \to c_2 \) in a quiver, we call \( c_1 \) the source of \( f \) and \( c_2 \) the target of \( f \). In a quiver \( Q \), a path from \( c_1 \) to \( c_k \) is a non-empty finite list of nodes and arrows \( c_1 \xrightarrow{e_1} c_2 \xrightarrow{e_2} \cdots \xrightarrow{e_{k-1}} c_k \). A category \( C \) is a quiver equipped with the following structure:

- for all objects \( c_1, c_2, c_3 \), a function \( \circ_{c_1,c_2,c_3} : C(c_2,c_3) \times C(c_1,c_2) \to C(c_1,c_3) \), which we call composition, and for which we write \( x; y \) to mean \( y \circ x \), and
- for every object \( c \), an arrow \( \text{id}_c \in C(c,c) \), which we call the identity for \( c \).

We may drop subscripts on \( \text{id} \) and \( \circ \), when doing so does not create ambiguity. These data must obey axioms stating that \( \circ \) is associative and \( \text{id} \) is its unit:

\[
\text{id} \circ f = f \quad f \circ \text{id} = f \quad f \circ (g \circ h) = (f \circ g) \circ h
\]

An object \( c \) of a category \( C \) is called initial if for all \( c' \in C \), there is a unique morphism \( c \to c' \). Dually, an object \( c \) of a category is called final/terminal if for all \( c' \in C \), there is a unique morphism \( c' \to c \). A functor \( F : C \to D \) between categories \( C \) and \( D \) consists of:

- a class function \( F : \text{Ob}(C) \to \text{Ob}(D) \), and
- for every \( c_1, c_2 \in \text{Ob}(C) \), a function \( F_{c_1,c_2} : C(c_1,c_2) \to D(F(c_1),F(c_2)) \), where we may omit object subscripts when they can be inferred, such that

\[
F(\text{id}_c) = \text{id}_{F(c)} \quad F(f \circ g) = F(f) \circ F(g).
\]

A natural transformation \( h : F \Rightarrow G \) between functors \( F, G : C \to D \) consists of a family of morphisms \( h_c : F(c) \to G(c) \), indexed by objects in \( C \), called the components of \( h \), such that for every \( f : c_1 \to c_2 \) in \( C \) we have \( h_{c_2} \circ F(f) = G(f) \circ h_{c_1} \). The family of equations defining a natural transformation may be depicted as a commutative diagram:

\[
\begin{array}{ccc}
F(c_1) & \xrightarrow{F(f)} & F(c_2) \\
\downarrow h_{c_1} & & \downarrow h_{c_2} \\
G(c_1) & \xrightarrow{G(f)} & G(c_2)
\end{array}
\]

The commutativity of such a diagram means that any two parallel paths in the diagram have the same composition in \( D \). A pushout of objects \( A, B, C \) and morphisms \( f, g \) in a category, as shown below, is an object \( D \) and morphisms \( \alpha \) and \( \beta \) as shown below, having the universal property that for any other such \( D' \) and \( \alpha' \) and \( \beta' \), there is a unique morphism \( \theta \) making the diagram commute:

\[
\begin{array}{ccc}
A & \xrightarrow{g} & C \\
\downarrow f & & \downarrow \beta \\
B & \xrightarrow{\alpha} & D \\
\end{array}
\]

\[
\begin{array}{ccc}
\alpha' & \xrightarrow{\theta} & \beta' \\
\end{array}
\]

The dual notion of pushout is pullback. A pullback over an initial object is called a co-product. A pullback over a final object is called a product. Pullbacks generalize to arbitrary diagrams, in which case they are called colimits, but we do not define colimits here. Given two functors \( F : C \to D \) and \( G : D \to C \), we say that \( F \) is left adjoint to \( G \), written \( F \dashv G \), when for every object \( c \) in \( C \) and \( d \) in \( D \) that the set of morphisms \( F(c) \to d \) in \( D \) is isomorphic to the set of morphisms \( c \to G(d) \) in \( C \), naturally in \( c \) and \( d \) (i.e., when we independently consider each side of the isomorphism as a functor \( C \to \text{Set} \) and as a functor \( D \to \text{Set} \)).
7 Appendix: Equational Logic and Universal Algebra

In this section we review standard material on multi-sorted equational logic, following the exposition in [11]. A signature $\text{Sig}$ consists of:

1. A set $\text{Sorts}$ whose elements are called sorts,
2. A set $\text{Symbols}$ of pairs $(f, s_1 \times \ldots \times s_k \to s)$ with $s_1, \ldots, s_k, s \in \text{Sorts}$ and no $f$ occurring in two distinct pairs. We write $f : X$ instead of $(f, X) \in \text{Symbols}$. When $k = 0$, we may call $f$ a constant symbol and write $f : s$ instead of $f : \to s$. Otherwise, we may call $f$ a function symbol.

We assume we have some countably infinite set $\{v_1, v_2, \ldots\}$, whose elements we call variables and which are assumed to be distinct from any sort or symbol we ever consider. A context $\Gamma$ is defined as a finite set of variable-sort pairs, with no variable given more than one sort:

$$\Gamma := \{v_1 : s_1, \ldots, v_k : s_k\}$$

When the sorts $s_1, \ldots, s_k$ can be inferred, we may write a context as $\{v_1, \ldots, v_k\}$. We may write $\{v_1 : s_1, \ldots, v_k : s_k\}$. We may write $\Gamma \cup \{v : s\}$ as $\Gamma, v : s$. We inductively define the set $\text{Terms}^*(\text{Sig}, \Gamma)$ of terms of sort $s$ over signature $\text{Sig}$ and context $\Gamma$ as:

1. $x \in \text{Terms}^*(\text{Sig}, \Gamma)$, if $x : s \in \Gamma$,
2. $f(t_1, \ldots, t_k) \in \text{Terms}^*(\text{Sig}, \Gamma)$, if $f : s_1 \times \ldots \times s_k \to s$ and $t_i \in \text{Terms}^*(\text{Sig}, \Gamma)$ for $i = 1, \ldots, k$. When $k = 0$, we may write $f$ for $f()$. When $k = 1$, we may write $t_1.f$ instead of $f(t_1)$. When $k = 2$, we may write $t_1.f t_2$ instead of $f(t_1, t_2)$.

We refer to $\text{Terms}^*(\text{Sig}, \emptyset)$ as the set of ground terms of sort $s$. We will write $\text{Terms}(\text{Sig}, \Gamma)$ for the set of all terms in context $\Gamma$, i.e., $\bigcup\text{Terms}^*(\text{Sig}, \Gamma)$. An equation over $\text{Sig}$ is a formula $\forall \Gamma. t_1 = t_2 : s$ with $t_1, t_2 \in \text{Terms}^*(\text{Sig}, \Gamma)$; we will omit the $: s$ when doing so will not lead to confusion. A theory $\text{Th}$ is a pair of a signature and a set of equations over that signature. Associated with a theory $\text{Th}$ is a binary relation between (not necessarily ground) terms, called provable equality. We write $\text{Th} \vdash \forall \Gamma. t = t' : s$ to indicate that the theory $\text{Th}$ proves that terms $t, t' \in \text{Terms}^*(\text{Sig}, \Gamma)$ are equal according to the usual rules of multi-sorted equational logic. From these rules it follows that provable equality is the smallest equivalence relation on terms that is a congruence, closed under substitution, closed under adding variables to contexts, and contains the equations of $\text{Th}$. A morphism of signatures $F : \text{Sig}_1 \to \text{Sig}_2$ consists of:

- a function $F$ from sorts in $\text{Sig}_1$ to sorts in $\text{Sig}_2$, and
- a function $F$ from function symbols $f : s_1 \times \ldots \times s_n \to s$ in $\text{Sig}_1$ to terms in $\text{Terms}^F(s)\{v_1 : F(s_1), \ldots, v_n : F(s_n)\}$.

To clearly indicate the context $\{v_1, \ldots, v_n\}$, the function $F(f)$ may be written in “$\lambda$ notation”, i.e. as $F(f) = \lambda v_1, \ldots, v_n : g(v_1, \ldots, v_n)$ for some term $g$, where the $\lambda$ is omitted if $n = 0$.

For example, let $\text{Sig}_1$ consist of two sorts, $a, b$, and one function symbol, $f : a \to b$, and let $\text{Sig}_2$ consist of one sort, $c$, and one function symbol, $g : c \to c$. There are countably infinitely many morphisms $F : \text{Sig}_1 \to \text{Sig}_2$, one of which is defined as $F(a) := c, F(b) := c, F(f) := \lambda v : c : g(g(v))$. In the literature on algebraic specification, our definition of signature morphism is called a “derived signature morphism” [9].

A morphism of theories $F : \text{Th}_1 \to \text{Th}_2$ is a morphism of signatures that preserves provability:

$$\text{Th}_1 \vdash \forall v_1 : s_1, \ldots, v_n : s_n. t_1 = t_2 : s \implies \text{Th}_2 \vdash \forall v_1 : F(s_1), \ldots, v_n : F(s_n). F(t_1) = F(t_2) : F(s)$$

where we have extended $F$ to operate on terms. An algebra $A$ over a signature $\text{Sig}$ consists of:

- a set of carriers $A(s)$ for each sort $s$, and
- a function $A(f) : A(s_1) \times \ldots \times A(s_k) \to A(s)$ for each symbol $f : s_1 \times \ldots \times s_k \to s$. 

29
Let \( \Gamma := \{ v_1 : s_1, \ldots, v_n : s_n \} \) be a context. An \( A \)-environment \( \eta \) for \( \Gamma \) associates each \( v_i \) with an element of \( A(s_i) \). The meaning of a term in \( \text{Terms}(\text{Sig}, \Gamma) \) relative to \( A \)-environment \( \eta \) for \( \Gamma \) is recursively defined as:

\[
A[v] \eta = \eta(v) \quad A[f(t_1, \ldots, t_n)] \eta = A(f)(A[t_1] \eta, \ldots, A[t_n] \eta)
\]

An algebra \( A \) over a signature \( \text{Sig} \) is a model of a theory \( T h \) on \( \text{Sig} \) when \( T h \vdash \forall \Gamma. \ t = t' : s \) implies \( A[t] \eta = A[t'] \eta \) for all terms \( t, t' \in \text{Terms}(\text{Sig}, \Gamma) \) and \( A \)-environments \( \eta \) for \( \Gamma \). Deduction in multi-sorted equational logic is sound and complete: two terms \( t, t' \) are provably equal in a theory \( T h \) if and only if \( t \) and \( t' \) denote the same element in every model of \( T h \).

From a signature \( \text{Sig} \) we form its term algebra \( [\text{Sig}] \) as follows. The carrier set \( [\text{Sig}](s) \) is defined as the set of ground terms of sort \( s \). The function \( [\text{Sig}](f) \) for \( f : s_1 \times \cdots \times s_k \to s \) is defined as the function \( t_1, \ldots, t_n \mapsto f(t_1, \ldots, t_n) \). From a theory \( T h \) on \( \text{Sig} \) we define its term model \( [T h] \) to be the quotient of \( [\text{Sig}] \) by the equivalence relation \( T h \vdash \). In other words, the carrier set \( [T h](s) \) is defined as the set of equivalence classes of ground terms of sort \( s \) that are provably equal under \( T h \). The function \( [T h](f) \) is \( [\text{Sig}](f) \) lifted to operate on equivalence classes of terms. To represent \( [T h] \) on a computer, or to write down \( [T h] \) succinctly, we must choose a representative for each equivalence class of terms; this detail can be ignored in this paper.

A morphism of algebras \( h : A \to B \) on a signature \( \text{Sig} \) is a family of functions \( h(s) : A(s) \to B(s) \) indexed by sorts \( s \) such that:

\[
h(s)(A(f)(a_1, \ldots, a_n)) = B(f)(h(s_1)(a_1), \ldots, h(s_n)(a_n))
\]

for every symbol \( f : s_1 \times \cdots \times s_n \to s \) and \( a_i \in A(s_i) \). We may abbreviate \( h(s)(a) \) as \( h(a) \) when \( s \) can be inferred. The term algebras for a signature \( \text{Sig} \) are initial among all \( \text{Sig} \)-algebras: there is a unique morphism from the term algebra to any other \( \text{Sig} \)-algebra. Likewise, term models are initial among models.

8 Appendix: The Categorical Theory of Algebraic Databases

In this section we briefly describe the categorical theory of algebraic databases, following [11]. We first fix a theory, \( T y \), called the type side of our formalism. The sorts of \( T y \) are called types and the functions of \( T y \) are the functions that can appear in schemas and instances. The intended meaning of this theory, \( [T y] \), is a category with products. A schema \( S \) on type side \( T y \) is a theory extending \( T y \) with new sorts (called entities), new unary functions from entities to types (called attributes), new unary functions from entities to entities (called foreign keys), and new equations (called data integrity constraints) of the form \( \forall v : s, t = t' \), where \( s \) is an entity and \( t, t' \) are terms of the same type, each containing a single free variable \( v \). The intended meaning of this theory, \( [S] \), is a category extending \( [T y] \). Let \( S \) and \( T \) be schemas on the same type side \( T y \). A schema mapping \( F : S \to T \) is defined as a “derived signature morphism” [9] from \( S \) to \( T \) that is the identity on \( S \). That is, \( F : S \to T \) assigns to each entity \( \tau \in T \) an entity \( F(\tau) \in S \), and to each attribute / foreign key \( f : s \to s' \) a term \( F(f) \), of type \( F(s') \) and with one free variable of type \( F(s) \), in a way that respects equality: if \( S \vdash t = t' \), then \( T \vdash F(t) = F(t') \). The intended meaning of \( F \) is a functor \( [S] \to [T] \) that is the identity on \( T y \). The collection of all schema mappings forms a category with colimits.

An instance \( I \) on schema \( S \) is a theory extending \( S \) with new 0-ary function (constant) symbols called generators and non-quantified equations. The intended meaning of an instance \( I \), written \( [I] \), is the term model (i.e., initial algebra) for \( I \) which contains, for each sort \( s \), a carrier set consisting of the closed terms of sort \( s \) modulo provability in \( I \). That is, the intended meaning of an \( S \)-instance \( I \) is a functor \( [S] \to \text{Set} \), namely, the initial such functor in the category of all functors consistent with \( I \). Let \( I \) and \( J \) be instances on the same schema \( S \). A data mapping \( h : I \to J \) is defined as a “derived signature morphism” [9] from \( I \) to \( J \) that is the identity on \( S \). That is, \( h : I \to J \) assigns to each generator \( g : s \in I \) a closed term \( h(g) : s \in J \) in a way that respects equality: if \( I \vdash t = t' \), then \( J \vdash h(t) = h(t') \). A morphism of instances thus denotes homomorphism (natural transformation) of algebras \( [I] \to [J] \). The database instances and morphisms on a schema \( S \) constitute a category with all colimits, denoted \( \text{S-Inst} \), and a schema mapping \( F : S \to T \) induces a functor \( \Sigma_F : \text{S-Inst} \to \text{T-Inst} \) defined by substitution. The functor \( \Sigma_F \) has a right adjoint, \( \Delta_F : \text{T-Inst} \to \text{S-Inst} \), which corresponds to composition when we are thinking semantically: \( [\Delta_F(I)] = [F][I] \). Semantically, \( [\Sigma_F(I)] \) computes the “left Kan-extension” of \( [I] \) along \( [F] \) [11].

30
| Well | BHP Rating | Casing Grade | Casing Size | Casing Weight | Collapse Rating | Measured Depth | Surface No. | Interval | Total Vertical Depth |
|-----|------------|--------------|-------------|---------------|----------------|----------------|-------------|----------|---------------------|
| 1   | 3275       | A216-Gr B X-70 | 2-9/16''  | 28.21         | 351200000000 | 7679           | 600000000000 | 61       | 2                   |
| 2   | 15802      | 11'' 12.74# A-35 | 27''      | 75         | 1135100000000 | 787571         | 150000000000 | 62       | 4                   |
| 3   | 13723      | 2 1/2'' NPS Extension | 23''    | 75         | 1254500000000 | 7015           | 100000000000 | 63       | 6                   |
| 4   | 15888      | 9 5/8'' 13% N80-Q | 24''    | 88.886      | 1656000000000 | 7276          | 200000000000 | 64       | 8                   |
| 5   | 18380      | TN-Q215-6 CCTMV | 10-1/4'' | 75.9       | 1673800000000 | 28532         | 600000000000 | 65       | 12                  |
| 6   | 18390      | TN-Q215-6 CCTMV | 10-1/4'' | 75.9       | 1673800000000 | 28532         | 600000000000 | 66       | 15                  |
| 7   | 18310      | TN-Q215-6 CCTMV | 10-1/4'' | 75.9       | 1673800000000 | 28532         | 600000000000 | 67       | 18                  |
| 8   | 15875      | 14 1/2'' 13% N80-Q | 10-1/4'' | 75.9       | 1673800000000 | 28532         | 600000000000 | 68       | 21                  |
| 9   | 18390      | TN-Q215-6 CCTMV | 10-1/4'' | 75.9       | 1673800000000 | 28532         | 600000000000 | 69       | 24                  |
| 10  | 18095      | 13-3/8'' 13% N80-Q | 10-1/4'' | 75.9       | 1673800000000 | 28532         | 600000000000 | 70       | 27                  |
| 11  | 18095      | 13-3/8'' 13% N80-Q | 10-1/4'' | 75.9       | 1673800000000 | 28532         | 600000000000 | 71       | 30                  |
| 12  | 18095      | 13-3/8'' 13% N80-Q | 10-1/4'' | 75.9       | 1673800000000 | 28532         | 600000000000 | 72       | 33                  |
| 13  | 18095      | 13-3/8'' 13% N80-Q | 10-1/4'' | 75.9       | 1673800000000 | 28532         | 600000000000 | 73       | 36                  |
| 14  | 18095      | 13-3/8'' 13% N80-Q | 10-1/4'' | 75.9       | 1673800000000 | 28532         | 600000000000 | 74       | 39                  |
| 15  | 18095      | 13-3/8'' 13% N80-Q | 10-1/4'' | 75.9       | 1673800000000 | 28532         | 600000000000 | 75       | 42                  |
| 16  | 18095      | 13-3/8'' 13% N80-Q | 10-1/4'' | 75.9       | 1673800000000 | 28532         | 600000000000 | 76       | 45                  |
| 17  | 18095      | 13-3/8'' 13% N80-Q | 10-1/4'' | 75.9       | 1673800000000 | 28532         | 600000000000 | 77       | 48                  |
| 18  | 18095      | 13-3/8'' 13% N80-Q | 10-1/4'' | 75.9       | 1673800000000 | 28532         | 600000000000 | 78       | 51                  |
| 19  | 18095      | 13-3/8'' 13% N80-Q | 10-1/4'' | 75.9       | 1673800000000 | 28532         | 600000000000 | 79       | 54                  |
| 20  | 18095      | 13-3/8'' 13% N80-Q | 10-1/4'' | 75.9       | 1673800000000 | 28532         | 600000000000 | 80       | 57                  |
| 21  | 18095      | 13-3/8'' 13% N80-Q | 10-1/4'' | 75.9       | 1673800000000 | 28532         | 600000000000 | 81       | 60                  |
| 22  | 18095      | 13-3/8'' 13% N80-Q | 10-1/4'' | 75.9       | 1673800000000 | 28532         | 600000000000 | 82       | 63                  |
| 23  | 18095      | 13-3/8'' 13% N80-Q | 10-1/4'' | 75.9       | 1673800000000 | 28532         | 600000000000 | 83       | 66                  |
| 24  | 18095      | 13-3/8'' 13% N80-Q | 10-1/4'' | 75.9       | 1673800000000 | 28532         | 600000000000 | 84       | 69                  |
| 25  | 18095      | 13-3/8'' 13% N80-Q | 10-1/4'' | 75.9       | 1673800000000 | 28532         | 600000000000 | 85       | 72                  |
| 26  | 18095      | 13-3/8'' 13% N80-Q | 10-1/4'' | 75.9       | 1673800000000 | 28532         | 600000000000 | 86       | 75                  |
| 27  | 18095      | 13-3/8'' 13% N80-Q | 10-1/4'' | 75.9       | 1673800000000 | 28532         | 600000000000 | 87       | 78                  |
| 28  | 18095      | 13-3/8'' 13% N80-Q | 10-1/4'' | 75.9       | 1673800000000 | 28532         | 600000000000 | 88       | 81                  |

**Figure 19:** Full Integrated Olog, 1 of 4
Figure 20: Full Integrated Olog, 2 of 4
