Let $\text{Po}$ denote the category of finite partially ordered sets. We can consider objects in $\text{Po}$ as categories in which case the morphisms between the objects are exactly the functors between corresponding categories.

Let $\text{FC}$ denote the category of finite categories.

1. Derivation of tables

Given a functor $\pi: T \to S$, define $d(S) = (\text{FC}^\text{op} \Downarrow S)$ as follows. An object in $d(S)$ is a finite category $C$ together with a functor $\sigma: C \to S$. A morphism, denoted $(F,F^\sharp): (C,\sigma) \to (C',\sigma')$, in $d(S)$, consists of a functor $F: C' \to C$ and a natural transformation $F^\sharp: (\pi \circ r) \to \sigma'$, as depicted in the diagram

\[
\begin{array}{ccc}
C' & \xrightarrow{F} & C \\
\downarrow{\sigma'} & & \downarrow{\sigma} \\
S & \xleftarrow{\pi} & \end{array}
\]

There is a canonical functor $\Gamma: d(S) \to \text{Cat}$, defined by assigning to each $(\sigma: C \to S) \in d(S)$ the category for which an object is a pair $(r,r^\sharp)$, where $r: C \to T$ is a functor and $r^\sharp: (\pi \circ r) \to \sigma$ is a natural transformation, as depicted in the diagram

\[
\begin{array}{ccc}
C & \xrightarrow{\pi} & T \\
\downarrow{r} & & \downarrow{\pi} \\
S & \xleftarrow{r^\sharp} & \end{array}
\]

A morphism $(r_1,r_1^\sharp) \to (r_2,r_2^\sharp)$ in $\Gamma(C,\sigma)$ is a natural transformation $\alpha: r_1 \to r_2$ such that $r_2^\sharp \circ \alpha = r_1^\sharp$.

Let us define $\Gamma$ on morphisms in $dS$. Given a morphism $(F,F^\sharp): (C,\sigma) \to (C',\sigma')$, let $(r,r^\sharp) \in \Gamma(C,\sigma)$ denote an object. Then $(r \circ F): C' \to T$ is a functor and we have a composition of natural transformations

\[
\pi \circ (r \circ F) = (\pi \circ r) \circ F \xrightarrow{r^\sharp} \sigma \circ F \xrightarrow{F^\sharp} \sigma'.
\]

Together, these form a pair $((r \circ F),(F^\sharp \circ r^\sharp)) \in \Gamma(C',\sigma')$, and we define $\Gamma(F,F^\sharp)$ on objects in this way. In a similar way, we define $\Gamma(F,F^\sharp)$ for morphisms $\alpha: r_1 \to r_2$.

This project was supported in part by the Office of Naval Research.
Define a new category \( d(T) := (\text{Cat} \downarrow \Gamma) \): an object is a pair \((K, \sigma, \tau)\) where \( K \in \text{Cat} \) is a category, \( \sigma \in d(S) \) is an object, and \( \tau: K \to \Gamma(\sigma) \) is a functor; a morphism \((K, \sigma, \tau) \to (K', \sigma', \tau')\) consists of a functor \( g: K \to K' \), a map \( F: \sigma \to \sigma' \) in \( dS \), and a natural transformation \( \beta: \Gamma(F) \circ \tau \to \tau' \circ g \), depicted in the diagram

\[
\begin{array}{ccc}
K & \xrightarrow{\tau} & \Gamma(\sigma) \\
\downarrow{g} & & \downarrow{\beta} \\
K' & \xrightarrow{\tau'} & \Gamma(\sigma').
\end{array}
\]

Finally, define \( d(\pi): d(T) \to d(S) \) by the natural projection, \( d(\pi)(K, \sigma, \tau) := \sigma \). We call \( d(\pi) \) the derivation of \( \pi \).

Remark 1.0.1. We might call \( d(T) \) the category of databases on \( \pi \). Be careful, though, because these are not database states, but update diagrams. In other words if \( K = \{\ast\} \), then \( \tau: K \to \Gamma(\sigma) \) can be identified with a pair \((r, r')\), as above. That is, it is a \( C \)-shaped diagram of tables with schema over \( \sigma \). But morphisms in \( K \) correspond to morphisms of such database states, such as unions (and deletions going the other way), joins, projects, and selects, as long as everything has at least as much information as required by \( \sigma \).

Lemma 1.0.2. Let \( q: dT \to (\text{Cat} \times dS) \) denote the functor \((K, \sigma, \tau) \mapsto (K, \sigma)\). We can recover \( \Gamma \) from \( q \).

Proof. Let us first produce \( \Gamma: dS \to \text{Cat} \) on objects. Given \( \sigma: C \to S \) in \( dS \), we wish to find \( \Gamma(\sigma) \in \text{Cat} \). Consider the object \((\{\ast\}, \sigma) \in (\text{Cat} \times dS) \). The category of sections of \( q \) over \((\{\ast\}, \sigma) \) is \( \Gamma(\sigma) \).

Example 1.0.3. Let \( \pi: U \to DT \) denote a type specification. Let \( T = U \) and \( S = DT \). Then one can show that there is an equivalence of categories \( d(S) \cong (\text{Fin} \downarrow DT)^{op} \), that the functor \( \Gamma: d(S) \to \text{Sets} \subset \text{Cat} \) is defined as usual, and that there is another equivalence of categories \( d(T) \cong \text{Tables}^{\pi} \).

Example 1.0.4. Let \( p: U \to DT \) denote a type specification. Define \( S = d(DT) \cong \text{Sch}(p)^{op}, T = d(U) \cong \text{Tables}^{p}, \) and \( \pi = d(p): \text{Tables} \to \text{Sch}(p)^{op} \). We wish to investigate \( d\pi: dT \to dS \).

An object \( \sigma: C \to S \) of \( dS \) is a finite diagram of schema. For example, \( \sigma \) could be a finite set of schema. Then \( \Gamma(\sigma) \) is the category of \( \sigma \)-shaped diagrams of tables, where the tables do not have to be the exact shape specified by \( \sigma \): they must have at least that much info – they come equipped with a natural transformation to tables on \( \sigma \).

2. Comparison

There is a natural functor \( \varphi: S \to dS \) defined by sending an object \( s \in S \) to

\[ \varphi(s) := (\{\ast\} \mapsto s) \]
in \(dS = (\text{Cat}^{\text{op}} \downarrow S)\). A morphism \(f: s \to s' \in S\) defines a morphism \(\varphi(f): \varphi(s) \to \varphi(s')\) as depicted in the diagram

\[
\begin{array}{ccc}
\{s\} & \xrightarrow{\text{id}(\ast)} & \{\ast\} \\
\downarrow f & & \downarrow s \\
S & & \{s\}
\end{array}
\]

3. To do

- Do \(\Gamma \to d\Gamma\) and \(T \to dT\). Then write out the “change of datatypes” functor for these.
- Write down the definition of “topological space,” “continuous map of topological spaces,” and “category,” and show that topological spaces form a category.
- Perhaps show something like: continuous maps from a space to \(\mathbb{R}\) form a ring.
- Phrase the fact that two points determine a line.
- A “vertical table” has three columns: subject-property-object (e.g. David address Kincaid; Ralph eat carrot). These are generally considered unreadable. A “horizontal table” has lots of columns, one for each property, but is perhaps sparse (lots of nulls). In other words, it is a simplicial database with one simplex (and all its subsimplices).

To convert a vertical table into a horizontal table, first sort by property. That is, for each datum in the “property” column, find the table of subject-object pairs that has that datum. This process gives a new 2-column table: the first column is properties and the second column is subject-object tables. Now join these tables along their “subject” column. This gives a horizontal table.

Except it has one problem: the columns of the horizontal table are untyped. We never used the properties to type them. Doing so requires more thinking about the dynamics of tables.

4. Data Structures

Above, we defined for a category \(S\) a new category \(d(S)\) as

\[d(S) := (\text{FC}^{\text{op}} \downarrow S)\]

Perhaps instead, we should create something called “data structures” wherein we choose a category of finite categories \(D: \mathcal{C} \to \text{FC}\) and define \(d(S) := (D^{\text{op}} \downarrow S)\) as above. Namely, the objects of \((D^{\text{op}} \downarrow S)\) are pairs \((C, \sigma)\) where \(C \in \text{Ob}(\mathcal{C})\) and \(\sigma: D(C) \to S\) is a functor to the (fixed) category \(S\); the morphisms are pairs \((F, F^\dagger): (C, \sigma) \to (C', \sigma')\) are diagrams

\[
\begin{array}{ccc}
D(C') & \xrightarrow{D(F)} & D(C) \\
\downarrow \sigma' & & \downarrow \sigma \\
S & \xleftarrow{F^\dagger} & \{s\}
\end{array}
\]

of categories, where \(F: C' \to C\) is a morphism in \(\mathcal{C}\).
The old version, defined in Section 1, is found in this new version by setting \( C = \text{FC} \), the category of all finite categories, and \( D = \text{id}_{\text{FC}} \). In the new version, we can use any category that indexes finite categories. For example, we can create the data-structure “list” or the data-structure “binary tree” by setting \( C \) to be the category indexing these objects.

**Definition 4.0.5.** A data structure is a pair \((C, D)\) where \( C \) is a category and \( D: C \to \text{FC} \) is a functor from \( C \) to the category of finite categories. A morphism of data structures \((C_1, D_1) \to (C_2, D_2)\) consists of a pair \((a, a^\sharp)\) where \( a: C_1 \to C_2 \) is a functor and \( a^\sharp: (D_2 \circ a) \to D_1 \) is a natural transformation, as depicted in the diagram

\[
\begin{array}{ccc}
C_1 & \xleftarrow{a} & C_2 \\
\downarrow{D_1} & & \downarrow{D_2} \\
\text{Cat} & & \\
\end{array}
\]

**Remark 4.0.6.** It may be that the above is the opposite of the most convenient way to define data structures. This would not change the shape of Diagram (1), but it would now be regarded as a morphism \((C_2, D_2) \to (C_1, D_1)\).

**Remark 4.0.7.** It may be handy to require that the terminal category \( \{*\} \in \text{FC} \) is in the image of a data structure \( D: C \to \text{FC} \). That way, there will be a comparison functor \( \varphi: S \to d(S) \).

Given a morphism of data structures \( \alpha = (a, a^\sharp): (C_1, D_1) \to (C_2, D_2) \) as in Definition 4.0.5 and given a category \( S \), there is an induced map of derivations \( \alpha^*: d_1(S) \to d_2(S) \) defined as follows. To an object \( \sigma: D_1(C_1) \to S \) in \( d_1(S) \), the functor \( \alpha \) assigns the object

\[
D_2(aC_1) \xrightarrow{\alpha^\sharp} D_1(C_1) \xrightarrow{\sigma} S
\]

in \( d_2(S) \). To the morphism

\[
\begin{array}{ccc}
D_1(C_1') & \xrightarrow{D_1(F)} & D_1(C_1) \\
\downarrow{\sigma_1'} & & \downarrow{\sigma_1} \\
S & & \\
\end{array}
\]

one composes with the commutative square

\[
\begin{array}{ccc}
D_2(aC_1') & \xrightarrow{D_2(aF)} & D_2(aC_1) \\
\downarrow{a^\sharp} & & \downarrow{a^\sharp} \\
D_1(C_1') & \xrightarrow{F} & D_1(C_1)
\end{array}
\]

to obtain the morphism

\[
\begin{array}{ccc}
D_2(aC_1') & \xrightarrow{D_2(aF)} & D_2(aC_1) \\
\downarrow{\sigma'^*a^\sharp} & & \downarrow{\sigma a^\sharp} \\
S & & \\
\end{array}
\]
Notation may be subject to improvement.