ISSN 2631-4444

Compositional models for power systems

John S. Nolan[1], Blake S. Pollard[2], Spencer Breiner[2], Dhananjay Anand[2], and Eswaran Subrahmanian[3]

[1]University of Maryland, College Park, MD

[2]National Institute of Standards and Technology, Gaithersburg, MD

[3]Carnegie Mellon University, Pittsburgh, PA

The problem of integrating multiple overlapping models and data is pervasive in engineering, though often implicit. We consider this issue of model management in the context of the electrical power grid, as it transitions from centralized generation with unidirectional power flow towards a more decentralized 'Smart Grid.' Considering the problem of distributing power on the grid, we show how to connect a generic problem specification with implementation specific numerical solvers using the paradigm of categorical databases. Zooming in to the device level, we define a symmetric monoidal category of distributed energy resources (DERs), i.e. entities which produce, consume, or store power. The tensor product in this category gives a way of aggregating such devices and their feasible operating regimes.

1 Introduction

The modeling of complex systems, engineered or natural, entails certain generic challenges: the existence and interaction of multiple models, multiple algorithms, and multiple implementations. This paper presents a methodology rooted in category theory to manage this complexity. We concretize this methodology within a model-driven engineering approach to designing a modern electrical grid, dubbed the 'Smart Grid.'

The existing electrical grid architecture is often decomposed into generation, transmission, distribution, and consumption layers. Spatially, these layers span from neighborhood to nation, temporally from microsecond to month. The relevant physical and mathematical models vary across scales as do the tools available to specify, analyze, and implement them. Distribution networks or feeders tend to be tree-structured. Historically, you are connected to the service transformer in your backyard, but not to your neighbor's house. High-voltage transmission systems have more cycles as is depicted in Figure 1. In addition to scale (number of nodes etc.), such structural properties of the network impact the efficiency and applicability of algorithms or analyses one wishes to perform. For this paper we ignore the transmission layer, focusing on distribution and device-level modeling.

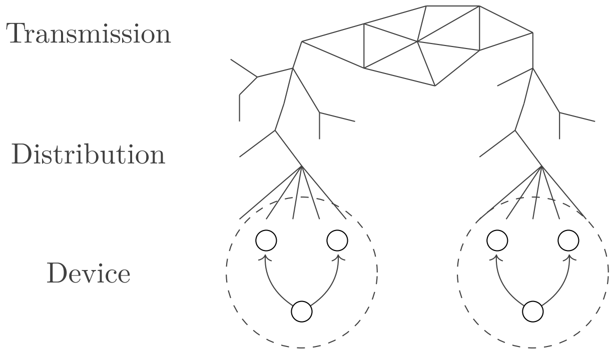

Figure 1: A schematic of a multiscale grid architecture.

In Section 2, we introduce a schema for a basic power flow or distribution problem. A database schema is a finitely presented category (i.e. directed, reflexive multi-graph) and a database instance is a functor from this schema category to Set [11, 13, 20]. Section 2.1 illustrates how to relate different numerical solvers in this setting.If models are schemas regarded as a categories, then model mappings are functors between schemas. Such a functor naturally induces several translations between instances conforming to the source and target schemas [22]. Such mappings provide the basis for a methodology for connecting multiple models and tools, outlined in Section 2.2. These models and methodologies were implemented in the Categorical Query Languge (CQL), developed by Categorical Informatics, an MIT-based startup [19].

In Section 3 we consider a category of distributed energy resources (DERs). The notion of DER is meant to provide a uniform abstraction or characterization summarizing the essential properties of a wide array of different energy resources, e.g. photovoltaic (PV) systems, batteries, conventional loads. For operations at the distribution level, the essential information is the *net* power demand of a collection of DERs. We present a category of models of DERs as reflexive graphs equipped with power demand data, where morphisms can be viewed as model transformations or inclusions. This category admits a natural symmetric monoidal structure with DER aggregation as tensor product.

2 Power flow problems

In this Section, we begin by describing the simplest 'power flow problem,' namely the problem of distributing electricity through the grid so as to match production and consumption in real time, while obeying various operational constraints.

Definition 1. A ***power flow graph*** consists of a set of ***nodes*** (also called ***buses***) N, a set of edges (also called ***branches***) E, connected by source and target functions $s, t\colon E \to N$, together with functions $g, b\colon E \to \mathbb{R}$, assigning a ***conductance*** and ***susceptance*** to each edge.

A power flow graph is a directed multigraph whose edges are labelled by two real-valued parameters, summarized by the diagram $\begin{tikzcd} \mathbb{R} & E \arrow[l, shift left, "b"'] \arrow[l, shift right, "g"] \arrow[r, shift left, "s"] \arrow[r, shift right, "t"'] & N \end{tikzcd}$. Conductance and susceptance are the real and imaginary parts of the complex admittance, a measure of how easily current flows through a branch. The resistance, reactance, and complex impedance of a branch can be calculated from these .

Associated to each node in a power flow graph are four variables which we regard as partial functions $P, Q, V, \theta\colon \mathbb{N} \to \mathbb{R}$ called respectively **real power**, **reactive power**, **voltage magnitude**, and **voltage angle** or **phase**[1]. These variables are the real and imaginary parts of the complex power $P + iQ$ and the magnitude and phase of the complex voltage $Ve^{i\theta}$. For practical purposes, engineers use both polar and rectangular representations of complex numbers.

Throughout the following we fix an indexing $\{N_i\}_{i\in I}$ of N and write E_{ij} for the edge from N_j to N_i when it exists. We also use subscripts to denote the values of the (partial or total) functions above when they exist, e.g. $P_i := P(N_i)$ and $g_{ij} := g(E_{ij})$. Typically, g and b are symmetric so that $g_{ij} = g_{ji}$. Using this notation, imposing Ohm's and Kirchhoff's laws gives the power flow equations.

[1] In practical applications the voltage magnitude and voltage angle are given relative to those of a fixed "slack bus," although we will not concern ourselves with this.

Definition 2. The ***power balance equations*** [16] for a power flow graph are the $2|N|$ equations

$$\begin{aligned} P_i &= V_i \sum_j V_j\,(\,g_{ij}\cos(\theta_i-\theta_j)+b_{ij}\sin(\theta_i-\theta_j)\,) \\ Q_i &= V_i \sum_j V_j\,(\,g_{ij}\sin(\theta_i-\theta_j)-b_{ij}\cos(\theta_i-\theta_j)\,)\,. \end{aligned} \tag{1}$$

Each sum is taken over all buses adjacent to i.

Conventional power flow problems require the buses to have fixed values for two of the four P, Q, V, θ for each bus. Typically, three types of buses are considered: PQ buses, PV buses, and $V\theta$ (or slack) buses, at each of which the corresponding variables are fixed. A PQ bus represents a typical load, whose real and reactive power demands are known and fixed, at any moment of time. Generators are regarded as types of PV buses, producing constant power at a specific voltage. A single slack bus or $V\theta$ bus is chosen which provides the reference angle with respect to which the other phases are measured. Once the fixed variables are specified, the power balance equations can be solved using methods, e.g. Newton-Raphson or Gauss-Seidel.

Solving for the free variables attached to a network satisfying the power balance equations is known as the *power flow problem*. The *optimal power flow problem* is a related constrained optimization problem in which the power balance equations are viewed as constraints for the maximization or minimization of some objective function, such a line losses or overall cost. This static typing of buses as loads, generators, etc. is incompatible with modern distributed energy resources which switch between consuming and producing power.

2.1 Connecting to a tool

Due in part to their non-linearity, solving the power flow equations is typically done numerically either using freely available software, commercial tools, or customized code. Such tools usually require specific input file formats, specific solver parameters, and use their own internal data structures.

To interface with these specifications we use the functorial data model advocated in [19, 20, 22] as well as its computational implementation in the CQL tool. In the functorial data model, database schemas are interpreted as finite presentations of categories. Instances of a database schema correspond to Set-valued functors out of the associated category. Some subtleties arise when working with computational data such as strings and integers, though we will not concern ourselves with these difficulties; see [20] for a thorough treatment.

MATPOWER is a commonly used power systems toolbox, implemented in Matlab. The MATPOWER input/output data format specifications are organized into tables in Appendix B of the MATPOWER manual [24]. We translate these specifications into a MATPOWER-specific schema in CQL. Each table becomes an entity and each column becomes an attribute. This process is summarized in Figure 2 where we have omitted some of the attributes. In a similar fashion, we implemented schemas representing the parameters required to run a specific solver, in this case an iterative Newton-Raphson solver.

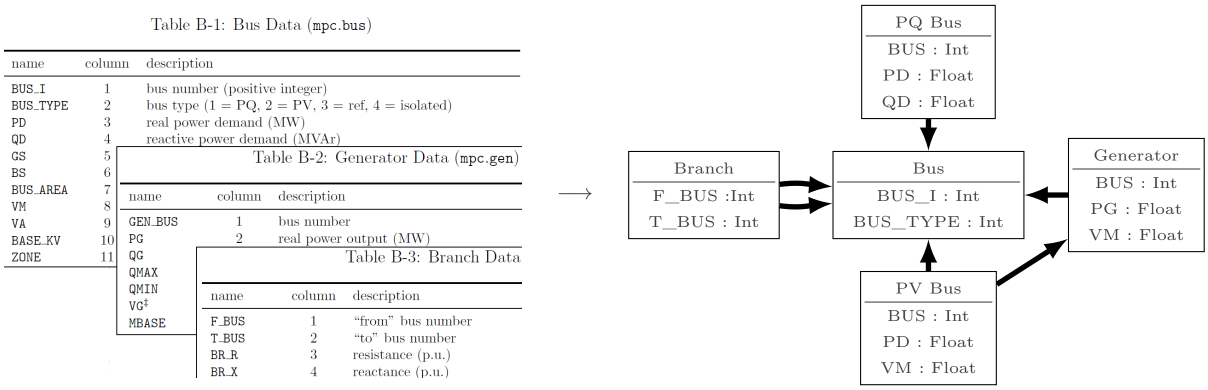

Table B-1: Bus Data (mpc.bus)

name	column	description
BUS_I	1	bus number (positive integer)
BUS_TYPE	2	bus type (1 = PQ, 2 = PV, 3 = ref, 4 = isolated)
PD	3	real power demand (MW)
QD	4	reactive power demand (MVAr)
GS	5	
BS	6	
BUS_AREA	7	
VM	8	
VA	9	
BASE_KV	10	
ZONE	11	

Table B-2: Generator Data (mpc.gen)

name	column	description
GEN_BUS	1	bus number
PG	2	real power output (MW)
QG		
QMAX		
QMIN		
VG‡		
MBASE		

Table B-3: Branch Data

name	column	description
F_BUS	1	“from” bus number
T_BUS	2	“to” bus number
BR_R	3	resistance (p.u.)
BR_X	4	reactance (p.u.)

Figure 2: Creating a MATPOWER power flow schema. Entities correspond to tables and attributes corespond to columns in each table. For simplicity we only show a few attributes for each entity. The arrows from branch assign source and target buses, giving the graph structure. The other arrows represent the typing of buses as PQ, PV, or generator buses.

CQL allows for the enforcement of constraints in the form of path equations. For example, in MATPOWER, buses come equipped with an integer ID, BUS_I and branches come equipped with integer variables F_BUS, T_BUS, specifying the values of BUS_I for the source and target of each branch. Considering the chunk of our schema on the left, the equations on the right enforce this constraint on instances:

$$\begin{tikzcd}
\texttt{Branch} \arrow[r, shift left, "\texttt{s}"] \arrow[r, shift right, "\texttt{t}"'] \arrow[d, shift right, "\texttt{T\_BUS}"'] \arrow[d, shift left, "\texttt{F\_BUS}"] & \texttt{Bus} \arrow[dl, "\texttt{BUS\_I}"] \\
\texttt{Int} &
\end{tikzcd}
\qquad
\begin{array}{l}
\texttt{s.BUS\_I} = \texttt{F\_BUS} \\[2em]
\texttt{t.BUS\_I} = \texttt{T\_BUS}
\end{array}$$

This provides the input data for a power flow problem. Solving such a problem would constitute assigning (P, Q, V, θ) values which satisfy the power balance equations. One such method for finding a solution is an iterative Netwon-Raphson solver. Such a solver requires initialization as well as certain parameters such as derivative approximation method, tolerance, etc. We can similarly encode such information in a schema, as well as information the solver might provide beyond the solution such as elapsed time or number of iterations to convergence. This enables both systematic experimentation, i.e. varying inputs, parameters, initialization, while providing flexible and traceable documentation, i.e. storing just solutions or including the solver settings used in each run.

2.2 Connecting tools

For any common problem in systems engineering one can expect a wide range of tools to have been developed. Engineers may create different tools to handle different variations on the same problem or to allow for the use of different methods in solving such a problem. The power flow and optimal power flow problems are no exception to this rule. Numerical solvers for these problems exist within tools such as MATPOWER [24], GridLAB-D [6, 7], and many others.

The transformation of an abstract solver model into a model for a specific solver can be interpreted as a mapping (i.e. a functor) between the associated schemas. This functor typically takes the form of an inclusion sending entities and attributes of the generic model to the corresponding entities and attributes in the specific model, leaving some attributes of the specific model uninitialized.

In [22] it is shown that functors $F\colon S \to T$ between database schemas give rise to adjoint triples of functors $\Sigma_F, \Delta_F, \Pi_F$ between the associated categories of instances. Here $\Delta_F\colon T\text{-}\mathsf{Inst} \to S\text{-}\mathsf{Inst}$ and $\Sigma_F, \Pi_F\colon S\text{-}\mathsf{Inst} \to T\text{-}\mathsf{Inst}$. These functors can be used to construct queries between databases [19]; a query Q between schemas S and T is written $Q\colon S \to T$, although it is not a functor between the categories in question. As in the relational data model (implemented, for example, by SQL), these queries can be evaluated to yield a result, i.e. there exists a functor $eval(Q)\colon S\text{-}\mathsf{Inst} \to T\text{-}\mathsf{Inst}$. However, the functorial data model also allows for a dual operation of "coevaluation" of queries, i.e. another functor $coeval(Q)\colon T\text{-}\mathsf{Inst} \to S\text{-}\mathsf{Inst}$. Using data migration functors and queries, along with CQL's built-in ability to compute colimits in categories of instances, offers a useful way to translate between the data associated to different solvers.

We present a general description of this translation process. Suppose we want to translate instances for a specific solver schema S into instances for another specific solver schema S' using the generic solver schema G. We define a query $Q\colon S \to G$ and a query $Q'\colon S' \to G$. Then $eval(Q) \circ coeval(Q')\colon S\text{-}\mathsf{Inst} \to S'\text{-}\mathsf{Inst}$ translates all the "generic" data from an initial instance of S to the resulting instance of S'. However, this translation loses the "non-generic" data associated to that instance. To resolve this difficulty, we define an auxiliary schema A for data which does not appear in the generic solver schema but which we would like to preserve in the translation process. We define functors $F\colon A \to S$ and $F'\colon A \to S'$ inserting the data of A into both specific solver schemas. The composite functor $\Delta_F \circ \Sigma_{F'}\colon S\text{-}\mathsf{Inst} \to S'\text{-}\mathsf{Inst}$ projects data from an instance of S to an instance of A and then includes data of A as an instance of S'. In this way, for every instance I of S, we obtain two instances of S': $coeval(Q')(eval(Q)(I))$, which contains the data of I that appears in G, and $\Sigma_{F'}(\Delta_F(I))$, which contains the data of I that appears in A. These can be combined using a suitable colimit (in $S'\text{-}\mathsf{Inst}$) to get a single instance of S' containing all the data it was possible to translate over from S.

3 Distributed Energy Resources

In this Section we describe the modeling of distributed energy resources using directed graphs, with physical states (On, Off, High, Low, Charging, Discharging, etc.) as nodes and transitions among the states as edges. To each state, we associate a region representing the feasible real and reactive power consumption/generation in that state. Morphisms in the category of such DERs correspond to adjusting the level of granularity of the state space. We describe a product of DERs which aggregates collections of DERs, thereby enabling one to reason about net power demand/generation from collections of connected devices.

Definition 3. A ***distributed energy resource*** (abbreviated ***DER***) $\mathcal{D} = (S, T, s, t, a, d)$ consists of a pair of finite sets S, T of ***states*** and ***transitions***, a pair of functions $s, t\colon T \to S$ specifying the ***source*** and ***target*** states of each transition, a function $a\colon S \to T$, satisfying $s \circ a = t \circ a = \mathrm{id}_S$, picking out ***identity transitions*** from each state to itself, and a function $d\colon S \to 2^{\mathbb{C}}$ assigning to each state $\sigma \in S$ a ***power demand region*** $d(\sigma) \subseteq \mathbb{C}$. For each state $\sigma \in S$ we write $1_\sigma = a(\sigma)$ and call 1_σ the identity transition of σ.

In particularly simple models, operating regions are just single points and the data of d is equivalent to that of a function from S to $\mathbb{C}$. Our syntax is patterned on that in [23].

$$\begin{tikzcd} T \arrow[r, shift left=2, "s"] \arrow[r, "t"] & S \arrow[l, shift left=2, "a"] \arrow[r, "d"] & 2^{\mathbb{C}} \end{tikzcd}$$

Figure 3: The underlying data for a DER model.

This definition can be summarized by the diagram in Figure 3. In reality, much more information is required to characterize a DER depending on its type, such as state of charge, energy characteristics of transitions, location, and so on [5, 18]. Here we maintain a simplified approach in which we only consider the net power generated or consumed by a DER as that is the information required to solve the power flow problem in the distribution network of DERs.

Definition 4. A ***morphism of DERs*** $\phi\colon \mathcal{D} \to \mathcal{D}'$ consists of a pair of functions (ϕ_S, ϕ_T), where $\phi_S\colon S \to S'$ and $\phi_T\colon T \to T'$, such that:

- For all $\tau \in T$, $\phi_S(s(\tau)) = s'(\phi_T(\tau))$ and $\phi_S(t(\tau)) = t'(\phi_T(\tau))$
- For all $\sigma \in S$, $\phi(1_\sigma) = 1_{\phi(\sigma)}$ and $d(\sigma) \subseteq d'(\phi(\sigma))$

Together with these morphisms (and the obvious identity morphisms and composition law), DERs form a category which we denote DER.

In short, a morphism of DERs is a homomorphism of the underlying graphs that acts as an inclusion of subsets on the demand regions for each state.

Morphisms in DER can be used to translate between models of a given DER. For an example, consider a device with two parts (labeled A, B) that are prone to breaking. This device can be represented by the DER instance outlined on the left of Figure 4. A simpler representation of this device, in which only the question of whether or not the device is working is considered, is presented on the right of Figure 4. These can be connected by way of a morphism sending

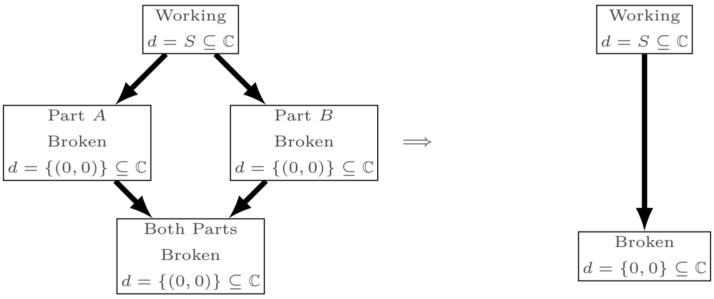
Working
$d = S \subseteq \mathbb{C}$
Part A
Broken
$d = \{(0,0)\} \subseteq \mathbb{C}$
Part B
Broken
$d = \{(0,0)\} \subseteq \mathbb{C}$
Both Parts
Broken
$d = \{(0,0)\} \subseteq \mathbb{C}$
$\Longrightarrow$
Working
$d = S \subseteq \mathbb{C}$
Broken
$d = \{0,0\} \subseteq \mathbb{C}$

Figure 4: A morphism of DERs. Dark arrows indicate transitions within a DER, and thin arrow indicates DER morphism.

the working state in the former representation to the working state in the latter model and the broken states in the former representation to the broken state in the latter representation.

3.1 Aggregation

In this subsection we describe a method of aggregating DERs into larger constructs to enable reasoning about DERs on different scales. This aggregation procedure, based on the category theoretic product of reflexive graphs as described explicitly in [23], serves as a symmetric monoidal product on DER. As a result, string diagrams can be used to reason about DERs and aggregation [14, 21].

Viewing aggregation in this way is especially useful because it enables the aggregation of *morphisms* of DERs. Consider again the device instances in Figure 4. Aggregates of several copies of the instance on the left would represent a collection of devices together with information on which part (if any) on each device is broken, while aggregates of several copies of the instance on the right would represent the same devices but would only include information on whether each device was working or broken. Aggregation of morphisms allows us to translate between these representations of aggregates using the same "forgetful" morphism described earlier.

Demands can be aggregated using Minkowski sums; see [10] for more details as well as [15] for an application to modeling the flexibility of DERs.

Definition 5. Given two subsets $X, Y \subseteq \mathbb{C}$, the ***Minkowski sum*** of X and Y is the set

$$X + Y = \{x + y : (x, y) \in X \times Y\} \subseteq \mathbb{C}. \tag{2}$$

Under this operation, $2^{\mathbb{C}}$ is a commutative monoid with unit $\{0\}$.

Now we are in a position to present a formal definition of aggregation.

Definition 6. Define a bifunctor $\otimes\colon \mathsf{DER} \times \mathsf{DER} \to \mathsf{DER}$ as follows.

- For objects $\mathcal{D}$ and $\mathcal{D}'$ in DER, define $\mathcal{D}\otimes\mathcal{D}' = (S\times S', T\times T', s\times s', t\times t', a\times a, d+d')$, where $d+d'\colon S\times S' \to 2^{\mathbb{C}}$ is defined by $(d+d')(\sigma,\sigma') = d(\sigma)+d'(\sigma') \subset \mathbb{C}$ for any $(\sigma,\sigma') \in S\times S'$.
- For morphisms $\phi\colon \mathcal{D}_1 \to \mathcal{D}_2$ and $\phi'\colon \mathcal{D}_1' \to \mathcal{D}_2'$ in DER define $\phi\otimes\phi'\colon \mathcal{D}_1\otimes\mathcal{D}_1' \to \mathcal{D}_2\otimes\mathcal{D}_2'$ by $(\phi\otimes\phi')_S = \phi_S \times \phi_S'$ and $(\phi\otimes\phi')_T = \phi_T\times\phi_T'$.

It is easy to see that this satisfies the axioms of a (bi)functor. Given DERs $\mathcal{D}$ and $\mathcal{D}'$, the DER $\mathcal{D}\otimes\mathcal{D}'$ is called the ***aggregate*** of $\mathcal{D}$ and $\mathcal{D}'$.

In short, the aggregate of two DERs is the categorical product of the underlying graphs [23], where each product state is equipped with demand equal to the Minkowski sum of its factors. Observe that the individual DERs within an aggregate DER can transition "independently:" if τ is a transition between states σ_1 and σ_2 in $\mathcal{D}$, then for any state σ' in $\mathcal{D}'$ there exists a transition $(\tau, 1_{\sigma'})$ between (σ_1,σ') and (σ_2,σ').

Theorem 7. Let $\mathcal{I}$ denote the DER with one state σ, a single transition 1_σ, and power demand $d(\sigma) = \{0\} \subseteq \mathbb{C}$. Then $\otimes$ makes DER into a symmetric monoidal category with unit $\mathcal{I}$.

Proof. Let Gph denote the category of graphs and graph homomorphisms. This category has finite products and is therefore symmetric monoidal under $\times$, with the terminal object, the underlying graph of $\mathcal{I}$, as unit [17]. We can define the associator, braiding, and unitors of DER via the corresponding natural transformations of Gph. These all give valid DER morphisms between aggregates of DERs: the associator because Minkowski sums are associative, the braiding because Minkowski sums are commutative, and the unitors because Minkowski sums are unital with unit $\{0\}$ (which is the demand of the single state of $\mathcal{I}$).

It remains to check commutativity of the required diagrams: naturality, triangle equations, et cetera. There exists a faithful forgetful functor $U\colon \mathsf{DER} \to \mathsf{Gph}$ sending a DER to its underlying graph and a morphism of DERs to its underlying graph homomorphism. In particular U sends the associator, braiding, and unitor to the corresponding natural transformations in Gph. Because U is faithful, a diagram is commutative in DER if and only if its image under U in Gph is commutative. By our choice of associator, braiding, and unitor for DER, we see that the images under U of the required diagrams are all commutative. Therefore the required diagrams all commute in DER. □

When aggregating DERs the state space grows rapidly. For operations at the distribution level, all that is relevant is the *net* power demand. Thus it is natural to mod out by an equivalence relation whereby states with identical power demand are identified. The following definition formalizes this notion.

Definition 8. Let $\mathcal{D}$ be a DER. Consider the equivalence relation $\sim$ on the states S of $\mathcal{D}$ where $\sigma \sim \sigma'$ if and only if $d(\sigma) = d(\sigma')$. This induces an equivalence relation $\approx$ on the edges T of $\mathcal{D}$ where $\tau \approx \tau'$ if and only if $s(\tau) \sim s(\tau')$ and $t(\tau) \sim t(\tau')$. We can define the ***net demand DER*** $\overline{\mathcal{D}}$ of $\mathcal{D}$ by $\overline{\mathcal{D}} = (S/{\sim}, T/{\approx}, \overline{s}, \overline{t}, \overline{a}, \overline{d})$, where $\overline{s}, \overline{t}, \overline{a}$, and $\overline{d}$ are defined by $\overline{s}([\tau]) = [s(\tau)]$, $\overline{t}([\tau]) = [t(\tau)]$, $\overline{a}([\sigma]) = [a(\sigma)]$, and $\overline{d}([\sigma]) = d(\sigma)$. (It is easy to see that $\overline{s}, \overline{t}$, and $\overline{d}$ are well-defined.)

The equivalence relation above gives rise to a DER morphism $\overline{(\ )}\colon \mathcal{D} \to \overline{\mathcal{D}}$ which identifies states with equal power demand and transitions among them. Composing this morphism with aggregation applied to a pair of DERs $\mathcal{D}$ and $\mathcal{D}'$ gives a DER $\overline{\mathcal{D} \otimes \mathcal{D}'}$ which summarizes the information relevant to operations in the distribution layer, namely the net power demand.

Any path in $\overline{\mathcal{D}}$ will give a set of paths in $\mathcal{D}$ traveling among DER states. We can then consider methods for selecting the 'best' or 'least-costly' sequence of DER transitions which accomplish some desired transition in net demand. This allows for dynamic tasking of DERs to accommodate demand fluctuations without requiring distribution level operators to have full knowledge of the details of a collection of DERs.

One such method of path selection proceeds by applying a breadth-first search [8] from the initial state until a state meeting specified constraints on the complex power is found. More sophisticated methods of path selection are possible; for example, one could label the edges in the aggregate graph with nonnegative weights giving the virtual cost of each transition, and then apply Dijkstra's algorithm [8].

We implemented a version of the above DER specification in CQL. With this, we can consider a hybrid framework where the net demand at each PQ bus in a power flow problem is specified by an aggregate collection of DERs. One could then investigate the dependence of the power flow solution on the properties, dynamics, and mixes of DERs at the device level.

4 Conclusions and Future Work

This paper provides a window into our efforts to concretize the potential utility of a category theoretic viewpoint for problems dealing with multiple models, tools, and scales in the context of power systems engineering. We engaged with only the simplest relevant models from power systems ignoring costs, constraints, control, and so on. Of particular relevance for Smart Grid technologies are aspects of control and communication enabled by new devices such as Smart Meters and increased deployment of phasor measurement units (PMUs), devices which measure current, voltage, or phase across the grid. Managing this coupling of an information network with a physical power network presents ample opportunities for applied category theorists.

We also made no mention of functorial semantics. Each layer of the grid can be described using certain types of network or graph-based syntax, equipped with additional data. Different layers may admit different semantic categories, e.g. the category of relations. It would be natural to apply the recent work done by Baez, Fong, and collaborators on open network or hypergraph categories in this setting [1, 2, 4, 12].

In this work, we considered a category whose objects are DERs and whose morphisms were transformations between more and less fine-grained descriptions of the DER. It would be sensible to level-shift and consider a category whose morphisms are DERs, so that composition corresponds to building up more complex systems of DERs. Different levels of detail could then be controlled at the 2-morphism level of a suitable bi or 2-category, [9]. Of particular relevance for the modelling of DERs would be the related work on open Petri Nets [3, 4].

Further work is required to extend this category-theoretic modeling paradigm to other engineering domains as well as to other areas of power systems engineering. What is really desired is not a modelling framework which captures the full complexity of the today's grid, but rather a framework which enables the expedient exploration and evaluation of various possible future architectures and pathways to those. The need for such an modeling ecosystem is not unique to power systems. It would be interesting to analyze other attempts to evolve ecosystems for systems modelling, in order to make progress towards a general theory. Of particular importance is the development of tools for specifying and modeling systems using category theory, e.g. CQL. In terms of engagement with domains, being able to point practitioners to a system they can get their hands on and play with goes a long way towards arriving at a useful common understanding.

Official contribution of the National Institute of Standards and Technology; not subject to copyright in the United States. Certain commercial equipment, instruments, or materials are identified in this paper in order to specify the experimental procedure adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose.

Parts of this paper may have been presented in technical seminars and included in government publications. Recorded versions of those seminars and copyright free versions of publications are available through the National Institute of Standards and Technology.

References

[1] John C Baez and Jason Erbele. Categories in control. *Theory and Applications of Categories*, 30(24):836–881, 2015.

[2] John C Baez and Brendan Fong. A compositional framework for passive linear networks. *Theory and Applications of Categories*, 33(38):1158–1222, 2018.

[3] John C Baez and Jade Master. Open Petri nets. *arXiv preprint arXiv:1808.05415*, 2018.

[4] John C Baez and Blake S Pollard. A compositional framework for reaction networks. *Reviews in Mathematical Physics*, 29(09):1750028, 2017.

[5] Andrey Bernstein, Lorenzo Reyes-Chamorro, Jean-Yves Le Boudec, and Mario Paolone. A composable method for real-time control of active distribution networks with explicit power setpoints. part I: Framework. *Electric Power Systems Research*, 125:254 – 264, 2015. ISSN 0378-7796. DOI: https://doi.org/10.1016/j.epsr.2015.03.023. URL http://www.sciencedirect.com/science/article/pii/S0378779615000905.

[6] D. P. Chassin, K. Schneider, and C. Gerkensmeyer. GridLAB-D: An open-source power systems modeling and simulation environment. In *2008 IEEE/PES Transmission and Distribution Conference and Exposition*. IEEE, apr

2008. DOI: 10.1109/tdc.2008.4517260. URL https://doi.org/10.1109%2Ftdc.2008.4517260.

[7] David P. Chassin, Jason C. Fuller, and Ned Djilali. GridLAB-D: An agent-based simulation framework for smart grids. *Journal of Applied Mathematics*, 2014:1–12, 2014. DOI: 10.1155/2014/492320. URL https://doi.org/10.1155%2F2014%2F492320.

[8] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009. ISBN 0262033844, 9780262033848.

[9] Kenny Courser. A bicategory of decorated cospans. *Theory Appl. Categ*, 32 (995):1027, 2017.

[10] Rida T Farouki, Hwan Pyo Moon, and Bahram Ravani. Minkowski geometric algebra of complex sets. *Geometriae Dedicata*, 85(1-3):283–315, 2001.

[11] Michael Fleming, Ryan Gunther, and Robert Rosebrugh. A database of categories. *Journal of Symbolic Computation*, 35(2):127 – 135, 2003. ISSN 0747-7171. DOI: https://doi.org/10.1016/S0747-7171(02)00104-9. URL http://www.sciencedirect.com/science/article/pii/S0747717102001049.

[12] Brendan Fong. Decorated cospans. *Theory and Applications of Categories*, 30(33):1096–1120, 2015.

[13] Michael Johnson, Robert Rosebrugh, and RJ Wood. Entity-relationship-attribute designs and sketches. *Theory and Applications of Categories*, 10(3): 94–112, 2002.

[14] André Joyal and Ross Street. The geometry of tensor calculus, I. *Advances in Mathematics*, 88(1):55 – 112, 1991. ISSN 0001-8708. DOI: https://doi.org/10.1016/0001-8708(91)90003-P. URL http://www.sciencedirect.com/science/article/pii/000187089190003P.

[15] Soumya Kundu, Karanjit Kalsi, and Scott Backhaus. Approximating flexibility in distributed energy resources: A geometric approach. In *2018 Power Systems Computation Conference (PSCC)*, pages 1–7. IEEE, 2018.

[16] Prabha Kundur. *Power System Stability and Control*. McGraw-Hill, 1994.

[17] Saunders Mac Lane. Monoids. In *Categories for the Working Mathematician*, pages 161–190. Springer New York, 1978. DOI: 10.1007/978-1-4757-4721-8_8. URL https://doi.org/10.1007%2F978-1-4757-4721-8_8.

[18] Lorenzo Reyes-Chamorro, Andrey Bernstein, Jean-Yves Le Boudec, and

Mario Paolone. A composable method for real-time control of active distribution networks with explicit power setpoints. part II: Implementation and validation. *Electric Power Systems Research*, 125:265 – 280, 2015. ISSN 0378-7796. DOI: https://doi.org/10.1016/j.epsr.2015.03.022. URL http://www.sciencedirect.com/science/article/pii/S0378779615000899.

[19] Patrick Schultz and Ryan Wisnesky. Algebraic data integration. *Journal of Functional Programming*, 27:e24, 2017. DOI: 10.1017/S0956796817000168.

[20] Patrick Schultz, David I. Spivak, Christina Vasilakopoulou, and Ryan Wisnesky. Algebraic Databases. *arXiv e-prints*, art. arXiv:1602.03501, Feb 2016.

[21] P. Selinger. *A Survey of Graphical Languages for Monoidal Categories*, pages 289–355. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-12821-9. DOI: 10.1007/978-3-642-12821-9_4. URL https://doi.org/10.1007/978-3-642-12821-9_4.

[22] David I. Spivak. Functorial data migration. *Information and Computation*, 217:31 – 51, 2012. ISSN 0890-5401. DOI: https://doi.org/10.1016/j.ic.2012.05.001. URL http://www.sciencedirect.com/science/article/pii/S0890540112001010.

[23] Michael Stay and L. G. Meredith. Representing operational semantics with enriched Lawvere theories. *arXiv e-prints*, art. arXiv:1704.03080, Apr 2017.

[24] Ray Daniel Zimmerman, Carlos Edmundo Murillo-Sánchez, and Robert John Thomas. Matpower: Steady-state operations, planning, and analysis tools for power systems research and education. *IEEE Transactions on Power Systems*, 26(1):12–19, Feb 2011. ISSN 0885-8950. DOI: 10.1109/TPWRS.2010.2051168.