The Commutative Monoidal Structure of Elections:

Introduction

This section was motivated by John Baez and his cohort of students at UCR; they have been using Category Theory to study a plethora of interesting structures that permeate mathematics, computer science, biology, chemistry and many other prolific fields. Their team has built a categorical formalism to describe all sorts of networks. Generally, a network is considered as a set with some extra structure. This structure captures the process of transforming from prescribed inputs to prescribed outputs. While exploring the field of strategic voting and attempting to settle on a specific topic, I attended a conference on Quantum Physics and Logic at Chapman University (QPL June 10-14 2019). It was here I was introduced to the idea of structured cospans, which Baez advertised as a categorical framework that could be applied to garner a deeper understanding of any type of network [2]. This seemed abundantly applicable within Social Choice Theory. What is a voting system other than a network with very specific types of inputs and outputs? Voters come in with their preferences on a given issue, and after going through a particular transition phase, they settle on a winner. Although Baez and his colleagues have analyzed many different conceptions of open networks, including linear networks, Markovian processes, Reaction Networks, Petri Nets, and others ([6],[4],[7],[5],[3]), these concepts have yet to be applied to voting. Conveniently, the abstraction of a voting method almost perfectly mirrors that of a Petri Net in [5]. Thus it is my intention to follow the work of Baez as a guideline, to define a the category of Voting Webs that characterize voting systems.[1]

Preliminaries

This project was my first experience with Category Theory, and thus I think it is appropriate for me to start at the very beginning by establishing some preliminary definitions and adopting a consistent notation for the remainder of this paper.

Definition 1. *A category $\mathcal{C}$ consists of the following components (1,2) and adhere to the following properties (3-5):*

1. *A class of objects, $Ob(\mathcal{C})$. As done in [Baez Category Theory Notes], if $x \in Ob(\mathcal{C})$ we simply write $x \in \mathcal{C}$.*

[1]For a more formal definition on the structure of a voting system, see Section 1 of *Strategic Voting: Safe Manipulations Under Uncertainty* (Wayland 2019).

2. *A set of morphisms between $x, y \in C$ called $Hom_C(x, y)$. As much of Category theory is expressed through diagrams, the morphisms represent the arrows between objects in C. Analogous to group theory operations, there are certain requirements for the morphisms of a Category. As for notation, I'll adopt the convention used in [Baez Category Theory Notes] and write $f \in Hom_C(x, y)$ as $f : x \longrightarrow y$.*

3. *C must be closed under composition of morphisms; given $f : x \longrightarrow y$ and $g : y \longrightarrow z$ then you must also have $g \circ f : x \longrightarrow z$. Furthermore, this composition must be associative. I.e., $(h \circ g) \circ f = h \circ (g \circ f)$.*

4. *For any $x \in C$, you must have an identity morphism $1_x : x \longrightarrow x$.*

5. *Finally, the unity laws must hold:(Left Unity) $1_x \circ f = f$ for any $f : y \longrightarrow x$, (Right Unity) $g \circ 1_x = g$ for any $g : x \longrightarrow y$.*

Definition 2. *A functor is a mapping between categories $F : C \longrightarrow C'$ that associates an object in C' to each object in C and a morphism $F(f) : F(a) \longrightarrow F(b)$ in C' for each morphism $f : a \longrightarrow b$ in C such that:*

- *$F(1_x) = 1_{F(x)}$;*
- *$F(g \circ f) = F(g) \circ F(f)$ for all morphisms f, g in C.*

The categories that will be relevant to this paper are as follows:

a) **Set**, the category with sets as objects and functions between sets as morphisms and the subcategory of finite sets **FinSet**.

b) **CommMon**, the category of commutative monoids and monoid homomorphims and **CMC**, the category with commutative monoidal categories as objects and strict monoidal functors as morphisms.

c) **POSet**, the category of partial orderings on sets; the objects are posets and the morphisms are order preserving functions.

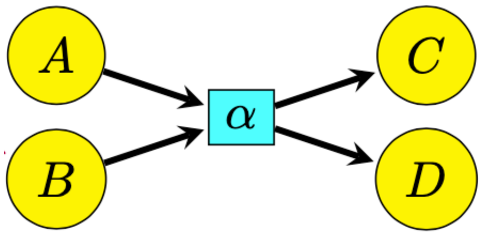
d) **Cat**, the category of "small" categories whose morphisms are functors.[2]

Naturally, I cannot possibly define all the categorical machinery working behind the scenes that allow for the construction of Petri Nets and Voting webs. However, I will attempt to be as comprehensive as possible with the qualities that are directly relevant to capturing the behavior of a voting system.

Petri Nets

What is a Petri net? Here is a simple example as shown in [5].

[2]Here "small" means each category's objects is a set



Effectively they are simple computational modeling tools that appear ubiquitously in the areas of computer science, biology, and chemistry. It was first suggested by Meseguer and Montanari in 1990 that Petri nets have a commutative monoidal structure[3]. They were interested in capturing the behaviour of computation in concurrent and compositional systems, which lead them to associate Petri Nets with strict symmetric monoidal categories (as defined in [8]). They also added an additional *distributivity property* that captured their notion of concurrent computations [9]. Although credit for this initial conception must be given to Meseguer and Montanari, Baez and Master take a more streamlined approach in [5] that is more appropriate for the scale of this paper: they asscociate Petri nets with *commutative monoidal categories* that inherently captures this distributivity property. I will adopt their methodology:

Definition 3. *A commutative monoidal category is a strict monoidal category* $(C, \otimes, I)$ *such that for all* $a, b \in C$ *and morphisms f,g:*

$$a \otimes b = b \otimes a \textit{ and } f \otimes g = g \otimes f$$

.

This takes a strict monoidal category and forces the symmetry isomorphisms $\sigma_{a,b} : a \otimes b \tilde{\rightarrow} b \otimes a$ to be identity morphisms [5]. This allows for the definition of **CMC**, the category whose objects are commutative monoidal categories and whose morphisms are strict monoidal functors.[4]

Section 2 of Baez and Master develops a pipeline for taking an interpretation of a Petri net P to the free commutative monoidal category on P and element of **CMC**. The collection of these Petri Categories which they call **PetriCat**, is a subcategory of **CMC**. They take a very similar approach as Meseguer and Montanari but with a more restrictive interpretation of the morphisms allowed in a Petri Net.

Voting Webs

This section will implement further restrictions to Petri nets that capture a) the implementation of a social welfare function to an aggregation of voters preferences and b) the

[3]The distinguishing feature is that the commutative monoid of objects is free

[4]Every monoidal functor between commutative monoidal categories is automatically a strict symmetric monoidal functor, which makes this a well defined subcategory.

execution of a computation that decides a set of winners according to a given voting rule. First I will define a categorical representation of a voting system (**VotingWeb**), from which you can (freely) generate a commutative monoidal category (**VotingWebCat**). Then I show an equivalence of **VotingWeb** and **VotingWebCat**, proving that this conception of voting systems has a commutative monoidal structure.

There are many different possible subtleties (or lack thereof) one could take to design a more specific (or more general) category that captures this computation. However, the key realization is that regardless of fine details, they should come to represent a commutative monoidal category. What is the motivation for this? It is nearly identical to the initial interest of Meseguer and Montanari. We should be able to use the monoidal structure to run two voting systems in parallel (via disjoint union) and combine populations of voters to produce an aggregated preference on a common issue. Furthermore, we should compose results of simple voting systems to create complex multistage elections that can span all combinations of population subsets and voting rules (ponder the presidential election in the United States for a minute if you don't think this is necessary).

Taking from the notation in my previous paper, consider a voting system with $|C| = n$ and $|V| = m$ under voting rule f. We can envision the system as a collection of voters coming together with packaged information (a total order on candidates) that then undergoes a transition phase (votes are counted) and returns a winning candidate. In the case of positional scoring rules, you are really transforming m strict linear orders on C, into a single (potentially non-strict) linear order on C that represents the rankings of the candidates. The similarity to a common Petri net is undeniable.

First, we need to get a handle on the our domain and codomain of a voting web computation and frame this categorically. As stated above, we are in fact transitioning between a subset of total orders on C to a singleton partial order. Thus the domain, called the set of transitions, needs to span all election possibilities. Call this domain T_{vote}. Each element of T_{vote} can be generated by the following triple: $\langle C, m, f \rangle$ for $C \in \mathbf{FinSet}, m \in \mathbb{N}, f$ is a voting method. Thus $\tau \in T_{\text{vote}}$ should package together a collection of preorders that represent the preferences of m voters on the set of candidates C and a voting method. These preorders can be generated by the following forgetful functor $G : \mathbf{POset} \to \mathbf{Set}$. Consider the preimage $G^{-1}(C)$; this is a discrete subcategory of POset where the objects are all possible preorderings on the set C with only identity morphisms.

Definition 4. *Let $\mathbb{O} = G \circ G^{-1}$. This is a functor*

$$\mathbb{O} : \textbf{\textit{Set}} \to \textbf{\textit{Set}}$$

and $\mathbb{O}[C]$ represents the set of all preorders on a set C.

Baez and Master use a monad defined by the composition $\mathbb{N} = K \circ J : \mathbf{Set} \to \mathbf{Set}$, where K,J give an adjunction between **CommMon** and **Set**. Their domain of interest is $\mathbb{N}[C]$ which captures finite linear combinations of C with natural number coefficients; clearly $\mathbb{O}[C]$ can be coded as a (strict) subset of $\mathbb{N}[C]$. Thus, a Voting Web can be defined as two functions going from T_{vote} to $\mathbb{O}[C]$, in analogous fashion to a Petri Net in [5].

Definition 5. *A Voting web is a specialized Petri Net that is defined by a source and a target function that have the following form:*

$$\begin{tikzcd} T_{\text{vote}} \arrow[r, shift left, "s"] \arrow[r, shift right, "t"'] & \mathbb{O}[C] \end{tikzcd}$$

These functions interact with a *fixed* set of transitions, T_{vote}. The functions s, t are also subject to the following restrictions:

- $|\text{Image}(s)| = n$ for $n \in \mathbb{N}$. So for a system with n voters the source function picks out precisely n preorders.
- $|\text{Image}(t)| = 1$. So the target function picks out a single preorder on C that represents the outcome of the election according to a well defined voting method f.[5]

This will be a very small subset of the possible source and target functions that appear in the context of generalized Petri Nets. However, considering the specific action of computing the outcome of an election (even with manipulation), this is to be expected. Since we are considering a fixed transition set, let $i : T_{\text{vote}} \to T_{\text{vote}}$ be the identity map on the set of voting methods. This allows for the following defintion:

Definition 6. *A **Voting Web morphism** from a petri net $V = (s, t : T_{vote} \to \mathbb{O}[C])$ to $V' = (s', t' : T_{vote} \to \mathbb{O}[C'])$ is a function $g : \mathbb{O}[C] \to \mathbb{O}[C']$ such that the diagrams below commute:*

$$\begin{tikzcd} T_{vote} \arrow[r, "s"] \arrow[d, "i"] & \mathbb{O}[C] \arrow[d, "\mathbb{O}(g)"] \\ T_{vote} \arrow[r, "s'"] & \mathbb{O}[C'] \end{tikzcd} \qquad \begin{tikzcd} T_{vote} \arrow[r, "t"] \arrow[d, "i"] & \mathbb{O}[C] \arrow[d, "\mathbb{O}(g)"] \\ T_{vote} \arrow[r, "t'"] & \mathbb{O}[C'] \end{tikzcd}$$

These last two definitions outline the behaviour of the category of voting webs, **Voting-Web**. As a restriction of Petri Nets, we know there is an underlying commutative monoidal structure. You can construct a commutative monoidal category $\hat{F}V$ from a Voting Web V $= (s, t : T_{\text{vote}} \to \mathbb{O}[C])$ as follows: let the objects of the category, $\text{Ob}(\hat{F}V)$, be defined as the free commutative monoid on C. For morphisms, consider the following recursive definition from [5]:

- for every $\tau \in T_{\text{vote}}$, include a morphism $s(\tau) \to t(\tau)$
- for any object a, include the morphism $1_a : a \to a$
- for morphisms $f : a \to b$ and $g : a' \to b'$, include a morphism to represent their tensor product $f + g : a + a' \to b + b'$
- for morphisms $f : a \to b$ and $g : b \to c$ include a morphism to represent their composition $g \circ f : a \to c$.

[5]Note that this supports single or multiple winners and that the prescription for f is contained in T_{vote}.

This recursive definition is more than exhaustive in acquiring all necessary morphisms. In order to obtain the commutative monoid $\mathrm{Mor}(\hat{F}V)$, take the set of morphisms described above and quotient out by an equivalence relation that imposes the laws of a commutative monoidal category [5].

Definition 7. *Let* $\hat{F} : \textbf{VotingWeb} \to \textbf{CMC}$ *be the functor as reflected below:*

$$\begin{tikzcd}
T_{vote} \arrow[r, shift left, "s"] \arrow[r, shift right, "t"'] \arrow[d, "i"] & \mathbb{O}[C] \arrow[d, "\mathbb{O}(g)"] & \mapsto & \hat{F}V \arrow[d, "\hat{F}(g)"] \\
T_{vote} \arrow[r, shift left, "s'"] \arrow[r, shift right, "t'"'] & \mathbb{O}[C'] & & \hat{F}V'
\end{tikzcd}$$

Here $\hat{F}(g) : \hat{F}V \to \hat{F}V'$ is defined on objects by $\mathbb{O}(g)$. On morphisms, $\hat{F}(g)$ is just the identity map. This functor is not necessarily a left adjoint. As done in [5], consider the corestriction of $\hat{F}$ to its image.

Definition 8. *Let* ***VotingWebCat*** *be the image of* $\hat{F}$.

This is the collection of commutative monoidal categories picked out by the functor $\hat{F}$. Each object is itself a **VotingWeb** and the morphisms are strict monoidal functors of the form $\hat{F}(g) : \hat{F}V \to \hat{F}V'$ where V, V' are related by a morphism g.

Theorem 1. *Consider the following functor:*

$$F : \textbf{VotingWeb} \to \textbf{VotingWebCat}$$

which is defined as the corestriction of $\hat{F}$. *This is a left adjoint, showing an equivalence between the two categories.*

Proof. This proof uses a condition for functoriality that is outlined in [1]: a functor defines an equivalence of categories if and only if it is faithful, full, and essentially surjective. F is essentially surjective and full by construction [5]. Thus all that remains to be shown is that F is faithful (i.e., the restriction is injective). If $\hat{F}(g) : \hat{F}V \to \hat{F}V'$ and $F(g') : \hat{F}V \to \hat{F}V'$ are the same functor then clearly it follows that $\mathbb{O}(g) = \mathbb{O}(g')$. This implies that $g = g'$, which means that they represent the same morphism in **VotingWeb**. ∎

Conclusion:

This last theorem shows that the underlying structure of a voting system is a commutative monoid. This result barely scratches the surface on what has been shown by Baez and Master in [5] regarding Petri Nets. They develop even more structure by considering framing the Petri Nets as symmetric monoidal double categories, with the goal of designing a syntax for describing open systems of this nature with "reachability as a choice of semantics". Arguably, this is possible for Voting Webs as well, which could provide a syntax for describing the most fundamental question in voting systems: reachability. I hope to develop this formalism further in a subsequent work.

References

[1] John C. Baez. Category Theory Course. pages 1–40, 2018.

[2] John C. Baez. STRUCTURED COSPANS John C. Baez. pages 1–47, 2019.

[3] John C. Baez and Brendan Fong. A Compositional Framework for Passive Linear Networks. pages 1–54, 2015.

[4] John C. Baez, Brendan Fong, and Blake S. Pollard. A compositional framework for Markov processes. *Journal of Mathematical Physics*, 57(3):1–43, 2016.

[5] John C. Baez and Jade Master. Open Petri Nets. 2018.

[6] Brendan Fong. The Algebra of Open and Interconnected Systems. 2016.

[7] Brendan Fong and Maru Sarazola. A recipe for black box functors. pages 1–34, 2018.

[8] Saunders Mac Lane. Categories for the Working Mathematician, 1971.

[9] José Meseguer and Ugo Montanari. Petri nets are monoids. *Information and Computation*, 88(2):105–155, 1990.