Week Twelve

Siva Sundar, EE23B151

January 3, 2025

Notes for Chapter 5

29^{th} December

- **Signal Flow Graph** is similar to a UWD with extra features:
  - ⋆ A **labelled component** which takes in an input (from left) and give output (to the right) as product of input with its labeled value.
  - ⋆ A **black dot** representing **copy** operation.
  - ⋆ A **white dot** representing **sum** operation.

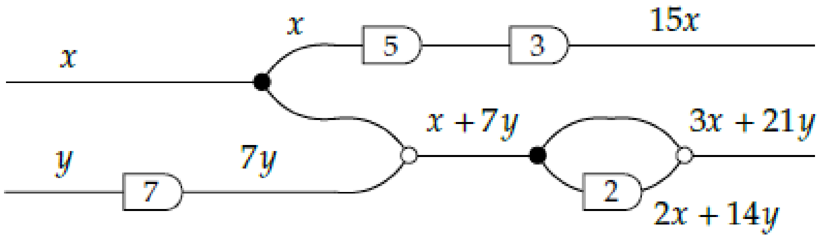


Figure 1: Signal flow graph example

- **Prop** (Products and permutations category) is a **symmetric strict monoidal category** $(\mathcal{C}, 0, +)$ for which $\mathrm{Ob}(\mathcal{C}) = \mathbb{N}$ (identified with finite sets), the **monoidal unit** is $0 \in \mathbb{N}$, and the **monoidal product** is given by *addition*.

  Hence, we can say every object n is the n-fold monoidal product of the *generating object* 1.

- Let $\mathcal{C}$ and $\mathcal{D}$ be props. A functor $F : \mathcal{C} \to \mathcal{D}$ is called a **prop functor** if
  (a) F is identity-on-objects: $F(n) = n \forall n \in \mathbb{N}$.
  (b) $\forall f : m_1 \to m_2$ & $g : n_1 \to n_2$ in $\mathcal{C}$, we have $F(f + g) = F(f) + F(g)$ in $\mathcal{D}$.

30^{th} December

- **Port graphs:** For $m, n \in \mathbb{N}$, an (m, n)**-port graph** (V, in, out, l) is given by:
  - ⋆ a set of vertices V.
  - ⋆ functions $in, out : V \to \mathbb{N}$, where $in(v)$ and $out(v)$ are called **in degree** and **out degree** of $v \in V$.
  - ⋆ a bijection $l : \underline{m} \sqcup O \xrightarrow{\cong} I \sqcup \underline{n}$[1], where I is the **set of vertex inputs** and O is the **set of vertex outputs**.

  This should also obey the **acyclicity condition**. (See page 151)

[1] $\underline{n} := \{1, 2, \ldots, n\}$

- Example 5.14 in text. It shows the acyclicity condition pictorically.

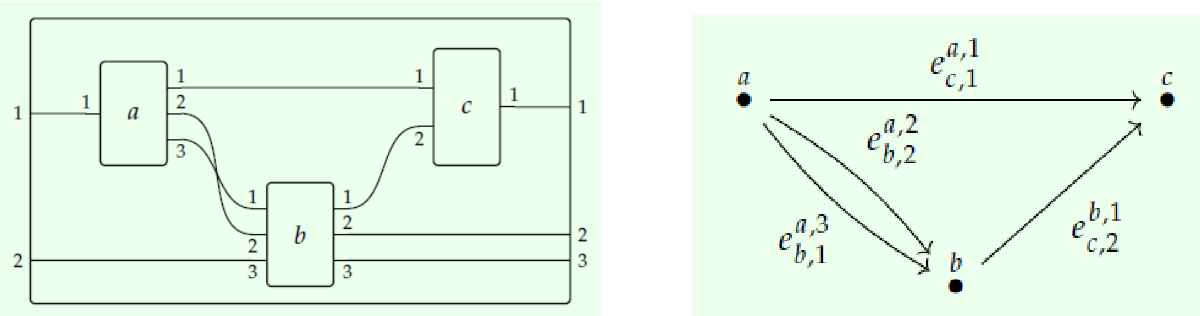


Figure 2: $(2, 3)$-port graph in the left, acyclicity condition on the right

- The category **PG** contains *objects* as natural numbers, *morphisms* as port graphs $\mathbf{PG}(m, n)$.

  Composition of a (m, n)-port graph (V, in, out, l) and a (n, p)-port graph (V', in', out', l') results in a (m, p)-port graph $(V \sqcup V', [in, in'], [out, out'], l'')$, where:

  - ⋆ $[in, in'] : V \sqcup V' \to \mathbb{N}$ which maps elements of V using in and of V' using in'. Similarly for $[out, out']$ as well.
  - ⋆ The bijection $l'' : m \sqcup O \sqcup O' \to I \sqcup I' \sqcup p$ is defined as:

  $$l''(x) = \begin{cases} l(x) & \text{if } l(x) \in I, \\ l'(l(x)) & \text{if } l(x) \in n, \\ l'(x) & \text{if } x \in O'. \end{cases}$$

  Identity morphism on n is given by (n, n)-port graph $(\varnothing, !, !, \mathrm{id}_{\underline{n}})$, where $! : \varnothing \to \mathbb{N}$ is the unique function.

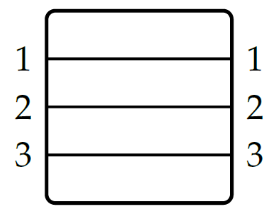


Figure 3: An identity morphism on 3

  The **monoidal product** of two port graphs, $G = (V, in, out, l)$ and $G = (V', in', out', l')$ is given by:

  $$G + G' := (V \sqcup V', [in, in'], [out, out'], l \sqcup l')$$

  The monoidal product of two morphisms is drawn by stacking the corresponding port graphs. (See Ex 5.18)

  The **monoidal unit** is $(\varnothing, !, !, !)$

  The category **PG** is in fact a prop.

1^{st} January

- The minimally-constrained structure that contains all the data you specify is called the **free structure** on your specification. (Free from unneccessary constraints.)

  Freeness of a category has to do with **maps out** of this category (See ex. 5.19).

  The maps between structured objects are defined to preserve constraints. This means the domain of a map must be somehow more constrained than the codomain. This is a universal property.

- A **prop signature** is a tuple (G, s, t), where G is a **set of generators** g and $s, t : G \to \mathbb{N}$ are functions, $s(g), t(g) \in \mathbb{N}$ are called its **in-arity** and **out-arity**.

  A G-**labelling** of a port graph $\Gamma = (V, in, out, l)$ is a function $\ell : V \to G$ such that the **arities agree**: $s(\ell(v)) = in(v)$ & $t(\ell(v)) = out(v) \forall v \in V$.

- Free prop **Free**(G) on a signature (G, s, t) has the property that $\forall$ prop $\mathcal{C}$, the prop functors **Free**$(G) \to \mathcal{C}$ are $\cong$ functions $G \to \mathcal{C}$. (which sends each generator $g \in G$ to an arrow $s(g) \to t(g)$ in $\mathcal{C}$)

- A G-**generated prop expression** $e : m \to n$ $(m, n \in \mathbb{N})$ is just G equipped with the following morphisms:

  ⋆ **Empty morphism** $\mathrm{id}_0 : 0 \to 0$.

  ⋆ **Identity morphism** $\mathrm{id}_1 : 1 \to 1$.

  ⋆ **Symmetry** (swap) $\sigma : 2 \to 2$.

  We call these morphsism in $G \cup \{\mathrm{id}_0, \mathrm{id}_1, \sigma\}$ as well as their *products and composition to each other* as **expressions** and denote this set of expressions in G as $\mathrm{Expr}(G)$.

  We can convert any port graph into a port expression, just draw vertical lines as shown in below figure and sum the morphisms.

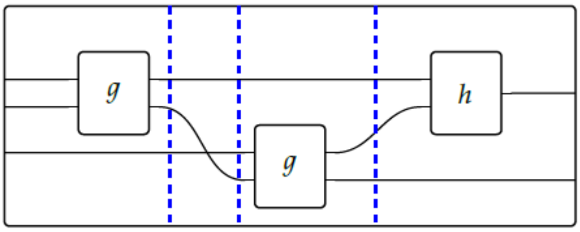


Figure 4: Port expression: $(h + \mathrm{id}_1) \circ (\mathrm{id}_1 + g) \circ (\mathrm{id}_1 + \sigma) \circ (g + \mathrm{id}_1)$

- **Rig** (also called *semi-rings*) is a tuple $(R, 0, +, 1, *)$, where R is a set, $0, 1 \in R$, and $+, * : R \times R \to R$:

  (a) $(R, +, 0)$ is a commutative monoid.

  (b) $(R, *, 1)$ is a monoid.

  (c) $a * (b + c) = a * b + a * c$ and $(a + b) * c = a * c + b * c \forall a, b, c \in R$.

  (d) $a * 0 = 0 = 0 * a \forall a \in R$.

  A rig is a ri**n**g without **n**egatives. (See ex 5.42)

- **Iconography of Signal Flow Graphs**

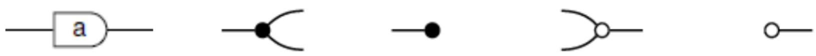
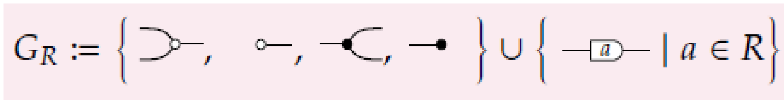


Figure 5: Scalar product, Copy, Discard, Add, Zero

2^{nd} January

- A **simplified signal flow graph** is a morphism in the free prop **Free**(G_R) =: $\mathbf{SFG}_R$ on the set G_R :



- **Prop of R-matrices Mat(R)** has morpshisms $m \to n$ as $(m \times n)$-matrices with values in R. Composition is given by:

$$N \circ M(a, c) := \sum_{b \in \underline{n}} M(a, b) \times N(b, c)$$

The **monoidal product** is given by the firect sum of matrices: given matrices $A : m \to n$ and $B : p \to q$, we define $A + B : m + p \to n + q$ to be the block matrix:

$$\begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix}$$

where each 0 represents a matrix of zeros of appropriate dimension ($m \times q$ and $n \times p$).

- **Turning signal flow graphs into matrices**: whatever be the wiring inside, we can create a matrix which represent the input and output relationship.

generator	icon	matrix	arity
amplify by $a \in R$		(a)	$1 \to 1$
add		$\begin{pmatrix} 1 \\ 1 \end{pmatrix}$	$2 \to 1$
zero		$()$	$0 \to 1$
copy		$\begin{pmatrix} 1 & 1 \end{pmatrix}$	$1 \to 2$
discard		$()$	$1 \to 0$

Figure 6: Interpretation of generators in matrix form

This is captured by the prop functor $S : \mathbf{SFG}_R \to \mathbf{Mat}(R)$, that sends generators $g \in G$ icons to the respective matrices.

ie, $S(g)$ represents the matrix for the signal flow graph g. The $(i, j)^{th}$ entry of this matrix describes the amplification of the ith input that contributes to the jth output.

We can also say that, for any matrix, we can make a signal flow graph:

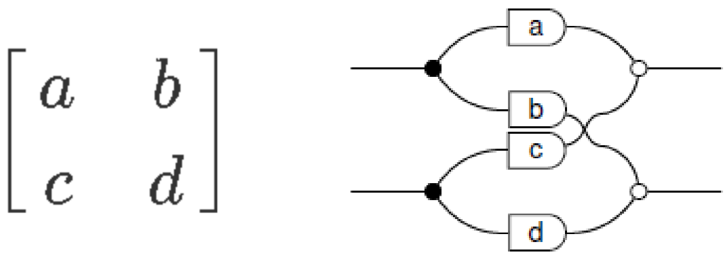


Figure 7: Equivalent graph for the 2×2-matrix

3^{rd} January

- A **monoid object** (M, μ, η) in a symmetric monoidal category $(\mathcal{C}, I, \otimes)$ is an object M of $\mathcal{C}$ with morphisms $\mu : M \times M \to M$ and $\eta : I \to M$:

  - ⋆ $\mu \circ (\mu \otimes \mathrm{id}) = \mu \circ (\mathrm{id} \otimes \mu)$
  - ⋆ $\mu \circ (\eta \otimes \mathrm{id}) = \mathrm{id} = \mu \circ (\mathrm{id} \otimes \eta)$

  A **commutative** monoid object (see example) also obeys:

  - ⋆ $\mu \circ \sigma_{M,M} = \mu$, where $\sigma_{M,M}$ is the **swap map**.

  See pictorial representation of these rules in example 5.68.

  Example 5.70 describes the **co-commutative comonoid object** in a sym. monoidal category, which is a commutative monoid object in the opposite category.

  A symmetric strict monoidal category is just a commutative monoid object in $(\mathbf{Cat}, \times, 1)$.

  A symmetric monoidal preorder is just a commutative monoid object in the symmetric monoidal category $(\mathbf{Preord}, \times, 1)$ of preorders and monotone maps.

- **Behavior** of a signal flow graph $g : m \to n$ (m inputs to n outputs) is given by $\mathrm{B}(g) \subseteq R^m \times R^n$:

  $$\mathrm{B}(g) = (x, S(g)(x)) | x \in R^m \subseteq R^m \times R^n$$

  This is similar to a **look-up table** (LUT).

  This definition gives us a way to interpret the mirror image of a generator icon (called **transposed relation**):

  $$\mathrm{B}(g^{op}) = (S(g)(x), x) | x \in R^m \subseteq R^n \times R^m$$

- We define the prop $\mathbf{SFG}_R^+ := \mathbf{Free}(G_R \sqcup G_R^{op})$, whose morphisms are called **(non-simplified) signal flow graphs**. These are extended versions of the simplified graph, because here we can also use the mirrored icons.

- **Feedback systems** using signal flow graphs: We have learnt about duals in chapter 4, where we discussed cup and cap icons. In signal flow graphs, the dual of an object is itself (See Page 178). Page 178, we have an example which shows usage of a feedback wire loop.