arXiv:2305.01041v1 [math.CT] 1 May 2023

字符串图的数据并行算法

Paul Wilson[1] 和 Fabio Zanasi[2]

[1]独立[2]伦敦大学学院和博洛尼亚大学

我们为表示为*ACSets* 的结构余跨的弦图提供并行算法。具体来说，我们给出用于复合、张量积、从任意Σ项构造图表以及将函子应用于图表的线性 （顺序）和对数（并行）时间算法。

我们的数据结构既可以表示任意签名上的自由对称幺半范畴的态射，也可以表示带有选定特殊 Frobenius 结构的态射。我们展示了如何利用这一额外的（超图）结构将图表映射为光学图。由此，我们展开了一项案例研究，定义了一种算法，能够基于反向导数高效地计算基于梯度的学习器的符号表示。

我们在此介绍的工作旨在作为一种通用数据结构。实现只需要整数数组和广为人知的算法，并且其设计本身就是数据并行的。因此，我们预计它适用于各种场景，包括嵌入式和并行硬件以及低级语言。

1 引言

弦图是一种形式化的图形语法 [23] ，用于表示幺半范畴中的态射，如今已得到广泛应用（例如参见 [16, 17, 18, 5]）。本文旨在使弦图不仅成为代数推理中方便的记号，也成为一种高效的通用工具，用于以复合的方式计算图形结构。为此，我们定义的数据结构和算法满足以下设计要求。

快速且数据并行。 我们的算法在构造上就是数据并行的，其时间复杂度分别为线性 （顺序执行）和对数级 （并行执行）。

最少的基本原语。 我们的数据结构是用简单的整数数组定义的。此外，我们只假设少量简单且广为人知的基本原语运算（例如前缀和）。这使得我们能够在各种环境中实现我们的算法，例如嵌入式硬件和并行（即 GPU）硬件。

易于正确实现。我们的数据结构的关键部分是根据近期提出的 $ACSets$ [21] 构造定义的。因此，其实现与相应的范畴论定义基本相同，从而更容易确保正确性。

文献中已经探讨了多种弦图表示形式，例如 Catlab.jl [20] 的布线图，以及 [26] 的“超图邻接表示”。我们的目标与后者最为接近：我们旨在让弦图成为一种通用的“可扩展组合语法”。例如，我们希望我们的实现能够在程序员目前会使用树或有向图的场景中，提供一种替代方案。

不过，我们这项工作的主要动机应用是将基于梯度的学习器表示为光学态射，如 [11] 所述。具体而言，这促使我们对 [26]作出了或许最重要的扩展：我们的数据结构能够“原生地”表示带有选定特殊弗罗贝尼乌斯幺半群的签名上的自由对称幺半范畴中的态射。这为范畴赋予了超图结构，我们将证明它可用于模拟光学图。进而，这使得许多应用得以基于“双向信息流”进行建模，

诸如 [11, 25, 7]光学态射。在我们的具体示例中，它使我们能够定义一种高效算法来计算反向导数 [9] ，并对基于梯度的学习器进行一般性建模。

我们的主要贡献如下：

- 给出了签名上的自由对称幺半范畴的态射的一种表示，即*ACSets* 的结构化余跨。
- 证明了该表示与自由对称幺半范畴之间的对应关系
- 针对以下操作，给出了具有线性（顺序）和对数（并行）时间复杂度的数据并行算法：– 图表的复合与张量积– 从任意 Σ-项构造图表– 将函子应用于图表
- 给出了一种利用超图结构将图表映射到光学图的算法，进而给出了一种能以线性（顺序）和对数（并行）时间计算图表反向导数的算法。

本文结构如下。第 2 节介绍必要的背景知识，包括弦图、由生成元和方程给出的表示、结构化余跨以及 ACSets。我们还回顾 [26], 中引入的超图二部图表示，并详细说明如何将有限函数表示为数组，这是我们实现的基础。我们的贡献从第 3节开始：我们展示如何使用 ACSets 表示图表的“内部连线”。第 4节在此基础上给出本文的主要定义：将弦图表示为这些连线的结构化余跨的数据结构。第 5 节证明这些结构化余跨与给定签名及选定的特殊弗罗贝尼乌斯幺半群上的自由对称幺半范畴之间的对应关系。第 6节将 [4]中首次引入的组合条件转化为我们的数据结构，从而能够表示自由对称幺半范畴的态射，而无需额外的 Frobenius 结构。第 7 节和第 8 节共同定义了一种高效算法，用于从 Σ-项构造图表。最后，第 9节定义了一种将函子应用于图表的算法；第 10节展示如何利用该算法将图表映射到光学图，从而高效计算反向导数。我们在第 11 节总结全文，并提出一些未来工作方向。本文所述部分算法的参考实现可见 https://yarrow.id。

2 背景

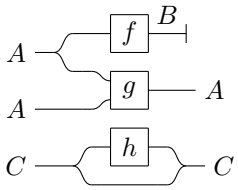
我们将介绍描述本文贡献所需的背景知识。第2.1 节回顾弦图、幺半群签名，以及由签名表示的自由对称幺半群范畴。第2.2节详细介绍两个同构的有限函数范畴，并给出复合、余积和张量积基本算法的计算复杂度。我们还列出实现所假设的基本原语数组运算。最后，在第2.4节中，我们讨论在[4], 中使用的将超图编码为二部图的方法，以及在 [26], 中引入的将弦图组合编码为超图的方法，并讨论ACSet的定义，据此我们将定义数据结构。

2.1 弦图、幺半群签名与自由$_\Sigma$

弦图是一种二维图形语法，用于表示范畴的态射。简单来说，这种语法由放置在页面上的小部件以及连接它们的带标签的线组成。下面的示例使用大写字母作为线标签，并由以下小部件构成： f， g， h、 ，



以及⊣



(1)

“小部件”和“线标签”的选择对应于生成态射和生成对象，它们属于一个幺半群签名。

定义 2.1（幺半群签名）。 幺半群签名 Σ 由以下部分组成。

- Σ_0 该生成对象
- Σ_1 这些生成态射或运算
- Σ_2 的方程
- 一个 类型关系 τ： $\Sigma_1 \to (\Sigma_0^* \times \Sigma_0^*)$

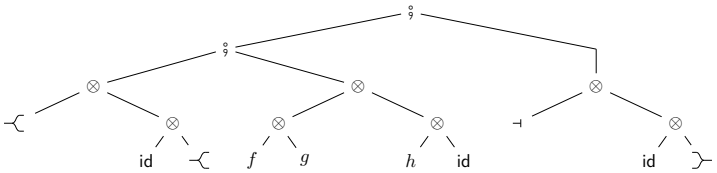
备注 2.2。 注意，我们纳入了一个类型赋值 关系，以便允许“多态”生成元存在。这意味着，例如，一个具有生成态射 f： $A \to A$ 对所有 A 都成立的对称幺半范畴表示，可以仅用 Σ_1中的一个标签来表示。我们已经在 (1)中看到了这种做法有用的例子：生成元被描绘为同时具有类型 $A \to A \otimes A$ 和 $C \to C \otimes C$。尽管这种区分很有用，但我们会经常假设给定生成元具有一个选定的类型，因此谈论生成元的“该”类型。

给定一个表示——定义一个范畴的基本小部件——我们现在可以定义 Σ-项：通过对生成元进行张量积和复合而归纳构造、表示图表的“语法树”。

定义 2.3。 给定一个幺半群签名 Σ， Σ-项 是一个二叉树，其叶子标记为$\Sigma_1 \cup \{\mathsf{id}\ \sigma\}$，其节点标记为 $\otimes$ 或 ⨟。

直观地说，一个 Σ-项由以下构件构成： Σ_1-运算以及“结构性”态射id (恒等态射)和 σ (对称性)，通过顺序复合（⨟）和并行复合（$\otimes$）构成。我们将在下面的例子中，将例((1) 中的字符串图重新表示为一个 Σ-项。

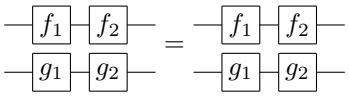
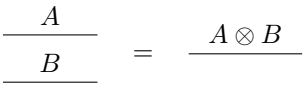
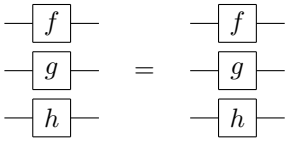
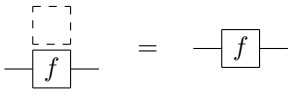
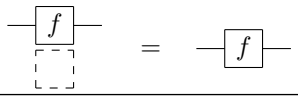
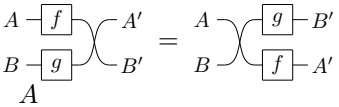
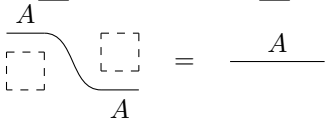
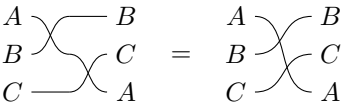
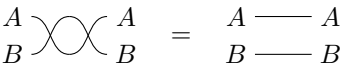
Exa例 2.4。 方程 (1) 可以表示为如下 Σ-项：$((\otimes(\mathsf{id} \otimes)) ⨟ ((f \otimes g) \otimes (h \otimes \mathsf{id}))) ⨟ (\otimes(\mathsf{id} \otimes))$，我们将其绘制为下面的二叉树。



有许多不同的 Σ-项可以表示图表((1)中的图表。为了表示自由对称幺半范畴中的态射，因此我们需要考虑Σ-项的同构类。

定义 2.5 (自由$_\Sigma$)。 给定一个幺半群签名 Σ，由自由生成的严格对称幺半范畴 Σ 记作 Free_Σ。其对象是生成对象的列表 Σ_0^*，其中单位对象为空列表。态射是满足对称幺半范畴定律的 Σ-项的商（见图 *1*）。恒等态射、对称性、复合和幺半积分别由其对应的 Σ-项给出。

图 1：严格对称幺半范畴的定律。

公理	Σ-项	图示语法
$\otimes$ 函子性	$(f_1 \otimes g_1) \fatsemi (f_2 \otimes g_2) = (f_1 \fatsemi f_2) \otimes (g_1 \fatsemi g_2)$	
$\otimes$ 函子性	$\mathsf{id}_A \otimes \mathsf{id}_B = \mathsf{id}_{A \otimes B}$	
α 自然性	$f \otimes (g \otimes h) = (f \otimes g) \otimes h$	
ρ 自然性	$f \otimes \mathsf{id}_I = f$	
λ 自然性	$\mathsf{id}_I \otimes f = f$	
σ 自然性	$(f \otimes g) \fatsemi \sigma_{A',\ B'} = \sigma_{A,\ B} \fatsemi (f \otimes g)$	
单位相干性	$\sigma_{A,I} = \mathsf{id}_A$	
对称相干性	$(\sigma_{A,B} \otimes \mathsf{id}_C) \fatsemi \mathsf{id}_B \otimes \sigma_{A,C} = \sigma_{A,B \otimes C}$	
逆律	$\sigma_{A,B} \fatsemi \sigma_{B,A} = \text{标识}_A \otimes \text{标识}_B$	

2.1.1 特殊 Frobenius 幺半群与超图范畴

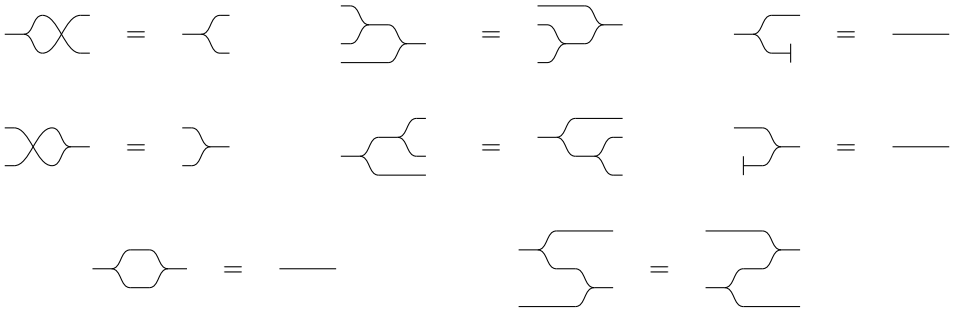
对我们而言，尤为重要的一种幺半群签名是特殊 Frobenius 幺半群的签名 .

定义 2.6 (特殊 Frobenius 幺半群)。 特殊弗罗贝尼乌斯幺半群的理论记作**Frob**，由生成元组成



(2)

以及方程



(3)

每个对象都配备了一个与幺半积相容的特殊弗罗贝尼乌斯幺半群的范畴，通常称为超图范畴 [13] (也称为良支撑紧闭范畴 [8])。

定义 2.7 (超图范畴)。 超图范畴是一个对称幺半范畴，其中每个对象 A 都配备了一个特殊 *Frobenius* 幺半群

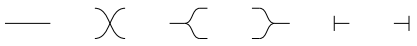


满足这些方程(3) 并且与张量积相容，即使得



超图结构将发挥特殊作用：它将用于表示字符串图的*Wires* 。具体来说，它将表示字符串图的*Wires* 。因此，引用仅由**Frob**中的生成元构造的态射会很有用。此类态射称为*Frobenius* 蜘蛛：我们现在对其进行定义。

定义 2.8 （Frobenius 蜘蛛）。 超图范畴中的一个***Frobenius*** 蜘蛛 是指通过对**Frob**中的生成元进行张量积和复合而构造的任意态射，即，



2.2 有限函数及其表示

由于我们的工作是以 ACSets [21]，来定义的，本文介绍的数据结构最终都将用范畴 FinOrd（由有限集与函数构成）来表示。我们需要这种范畴的两种不同（但同构）的“编码”：

- 自由**CMon**：一种“数学家友好”的编码，由签名和方程自由生成。
- FinFun：一种“程序员友好”的编码，以数组来定义。

尽管这两种表示都已得到充分研究（例如，参见 [21]），但将它们记录下来以供将来使用仍然很有价值。

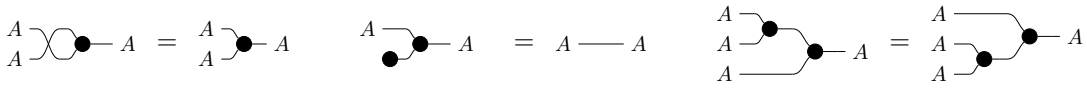
2.2.1 用交换幺半群表示有限函数

定义 2.9（交换幺半群）。给定选定的一组生成对象 Σ_0，交换幺半群理论 $\mathbf{CMon}$（Σ_0）有生成对象 Σ_0，并且对于每个 $A \in \Sigma_0$，生成箭头



(4)

及方程



(5)

交换幺半群理论的$\mathbf{CMon}_2$ 方程足以推出自然性方程（命题2.11 和2.10）。这些方程是可推导的（而非公理），这一点很重要，这样才能在第2.1.1节将有限函数与特殊 Frobenius 幺半群联系起来。

命题 2.10。对于所有态射 $f\colon A \to B$，其中属于 $\mathbf{CMon}(\Sigma_0)$，

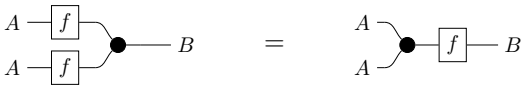


(6)

证明。归纳法。显然，所有生成元都满足（6）。在归纳步骤中，假设态射 f_0 和 f_1满足（6），那么很容易推出 $f_0 \otimes f_1$ 和$f_0 \fatsemi f_1$也满足（6）。

□

命题 2.11。对于所有态射 $f\colon A \to B$，其中属于 $\mathbf{CMon}(\Sigma_0)$，



(7)

证明。 按照命题 2.10进行归纳。 □

推论 2.12。 根据 *Fox* 定理 *[14]*, 命题 *2.10* 和 *2.11* 可知，$\mathbf{CMon}(\Sigma_0)$具有有限余积。

当 $\Sigma_0 = \{\bullet\}$ 时，我们将 **CMon** 写作 $\mathbf{CMon}(\Sigma_0)$，也就是说，只有一个生成对象。这是一个广为人知的事实，至少可追溯到 [6]: $\mathbf{Free}_{\mathbf{CMon}}$ 与有限集和函数的骨架化范畴 FinOrd 同构。

就我们的目的而言，研究 $\mathbf{Free}_{\mathbf{CMon}}$ 的一种略有不同的“函数式”表示 FinFun 很重要；它更明确地体现了这些结构的实现方式。为介绍它，我们首先需要说明数组和并行计算模型的必要背景。

2.2.2 用数组和并行程序表示有限函数

我们的实现所用的基本原语是数组，因此我们从记号开始。

定义 2.13（数组）。长度为 N、元素属于 S 的数组 x 是 N^S 的一个元素。数组 x 的第 i^{th} 个元素记作 x_i，当且仅当 x 的长度为 $N > i$ 时有定义。

具体来说，数组 x 是由 N 个元素组成的有序列表 $\langle x_0, \ldots, x_{N-1}\rangle$，其中每个元素都满足 $x_i \in S$。空数组记作 $\langle\rangle$，

为有限集采用专门的记号也很有用。

定义 2.14（有限集）。设 $A \in \mathbb{N}$ 是一个自然数。那么，有限集 $\{0, 1, \ldots, A-1\}$ 记作 A。

[1]骨架化意味着我们将同构的对象视为相同，即具有相同元素数量的集合。

在顺序 RAM 计算模型中，单次运算耗时一个时间步。例如，读取或写入一个记忆位置。不过，并行 计算 模型 [19]有多种。在本文中，我们将使用并行随机存取机（PRAM）模型的两种变体 [19, 第 11 页，第15]。其中第一种是 PRAM CREW（并发读／独占写）模型，我们将简称为 PRAM。该模型假设，尽管多个处理器可以并行读取同一记忆位置，但禁止多个处理器向同一位置进行相互冲突的写入。

稍强一些的模型是 PRAM CRCW（并发读／并发写），它允许多个处理器并行写入同一记忆位置。当发生写入冲突时，任意一个处理器都可能成功。我们很少需要 PRAM CRCW 的全部能力。除非明确说明，否则下文均假设采用 PRAM CREW 模型。

本文介绍的所有算法均基于少量基本原语整数数组运算。下面我们给出一张表，列出这些运算相对于输入 n 规模的顺序复杂度和 PRAM CREW 复杂度。注意，其中两个运算（`repeat` 和 `segmented arange`）的复杂度取决于其输入值，我们用 $|s|$ 表示数组 s的规模。

基本原语	复杂度（顺序）	复杂度（并行随机存取机）
`arange`	$O(n)$	$O(1)$
`zeros`	$O(n)$	$O(1)$
`sum`	$O(n)$	O(对数 n)
`prefix sum`	$O(n)$	$O(\log\ n)$
`dense integer sort`	$O(n)$	$O(\log\ n)$
`concatenate`	$O(n)$	$O(\log\ n)$
`connected components`	$O(n)$	$O(\log n)$（并发读并发写）
`repeat(x, s)`	$O(sum(s))$	$O(\log\ \|s\|)$
`segmented arange(s)`	$O(sum(s))$	$O(\log\ \|s\|)$

我们将以下内容归入 `sum` 和 `prefix sum` 其他运算，例如 `max` 和 `all`，这些运算可以通过并行扫描和折叠来实现。关于这些基本原语的更深入解释，我们请读者参阅我们的实现，地址为 https://yarrow.id。不过，我们现在先作简要概述。 `dense integer sort` 运算特指这样一类排序算法：作用于长度为 n 、最大元素为 $O(n)$的正整数数组。这类排序可以通过计数排序在 $O(n)$ 顺序时间内完成，也可以通过基数排序在 $O(\log n)$ 并行时间内完成。 `concatenate` 运算只是将多个数组复制到一个连续内存位置。在并行情形下使用该运算有一些细节问题，我们将在第9节进一步讨论。

`repeat(x, s)`运算接收两个长度相等的数组，并输出一个数组，其元素是将 x 重复由 s指示的次数所得的结果。例如， `repeat`$(\langle a, b, c\rangle, \langle 0, 1, 2\rangle) = \langle b, c, c\rangle$。 `arange` 基本原语输出一个长度为 n 的索引数组$\langle 0, 1, \ldots n-1\rangle$，而 `segmentedarange` 计算这些数组的串接，其长度由输入参数s指定。还要注意，`segmented arange` 实际上可以用其他运算来表示，因此不需要将其作为基本原语。

最后请注意，大多数运算的时间复杂度至多为 $O(n)$（顺序）和$O(\log n)$（PRAM CRCW）。我们稍后介绍的大多数算法都只涉及固定数量的此类运算，因此可确保线性（顺序）和对数（并行）时间复杂度。

选定并行计算模型后，我们现在可以给出将有限函数表示为数组时相关运算的基本复杂度结果。

2.2.3 FinFun：作为数组的有限函数

除了第 2.2.1 节中提到的以范畴 $\mathrm{Free}_{\mathbf{CMon}}$ 表示之外，有限函数也可以表示为 整数数组。这个范畴称为 FinFun，将在我们的实现中发挥核心作用。因此，我们将对其进行详细介绍，包括若干运算的复杂度结果。

定义 2.15（FinFun）。范畴 FinFun 的对象是自然数 $\mathbb{N}$。一个箭头 $f\colon A \to B$ 是一个元素 $f \in \underline{B}^{A}$。具体而言，f 是一个值数组，写作 $\langle x_0, x_1, \ldots, x_{A-1} \rangle$，其中每个 $x_i \in B$。

我们可以用目标（陪域）以及一个由其输出元素构成的表（数组）来表示 FinFun 中的箭头。箭头的源（定义域）是其元素表的长度。

```
class FiniteFunction:
  target: int
  table:  array

@property
def source(self):
  return len(self.table)
```

命题 2.16。FinFun 构成一个范畴，其恒等态射和复合定义如下。

$$\begin{aligned} &\mathsf{id} : A \to A && f \fatsemi g\colon A \to C \\ &\mathsf{id} = \langle 0, 1, \ldots, A-1 \rangle && (f \fatsemi g)(i) \mapsto g_{f_i} \end{aligned}$$

证明。复合是良定义的：g_{f_i} 总是有定义，恰因为 $f(i) \in B$。恒等律成立，因为 $\mathsf{id}_i = i$，因此对于箭头 $f\colon A \to B$，我们有 $(\mathsf{id}_A \fatsemi f)_i = \mathsf{id}_{f_i} = f_i$，并且 $(f \fatsemi \mathsf{id}_B)_i = f_{\mathsf{id}_i} = f_i$ 对所有 i 均成立。最后，注意复合满足结合律：设 $A \xrightarrow{f} B \xrightarrow{g} C \xrightarrow{h} D$ 为箭头。那么我们有 $((f \fatsemi g) \fatsemi h)_i = h_{(f \fatsemi g)_i} = h_{g_{f_i}} = (g \fatsemi h)_{f_i} = (f \fatsemi (g \fatsemi h))_i$ □

恒等态射和态射复合可以按如下方式实现。

```
@staticmethod
def identity(n: int):
  return arange(0, n) # [0, .., n - 1]

@staticmethod
def compose(f, g):
  assert f.target == g.source
  return FiniteFunction(g.target, g.table[f.table])
```

命题 2.17（FinFun中复合的复杂度）。设 $f : A \to B$ 和 $g : B \to C$ 是 FinFun 中的态射。计算复合态射 $f \fatsemi g$ 的顺序时间复杂度为 $O(A)$，*PRAM CREW* 时间复杂度为 $O(1)$。

证明。复合态射 $f \fatsemi g\colon A \to C$ 是一个整数数组，其中包含集合 C 中的 A 个元素。对于每个 $i \in A$，每个元素 $(f \fatsemi g)_i$ 都通过一次内存查找计算得出，总共需要 $O(A)$ 次运算。在并行（PRAM CREW）环境下，每次查找都可以并行执行，因此时间复杂度为 $O(1)$。 □

2.2.4 FinFun 作为严格幺半范畴

有限集与函数构成的范畴具有始对象和余积，因此可以将其构造成严格对称幺半范畴。

命题 2.18。0 是FinFun中的始对象。

证明。设 B 是 FinFun 中的一个对象。则存在唯一态射 $?\colon 0 \to B$：空数组 $\langle\rangle$。 □

在代码中，初始映射 `initial` $\colon 0 \to B$ 返回空数组。

```
@classmethod
def initial(cls, B):
  return FiniteFunction(B, zeros(0))
```

命题 2.19 (FinFun中的余积)。 设 $f: A_0 \to B$ 和 $g: A_1 \to B$ 是 FinFun 中的箭头。对象 A_0 和 A_1 的余积由加法 $A_0 + A_1$ 给出，其单射定义为

$$\iota_0 : A_0 \to A_0 + A_1 \qquad \iota_1 : A_1 \to A_0 + A_1$$
$$\iota_0 = \langle 0, 1, \dots A_0 - 1 \rangle \qquad \iota_1 = \langle A_0, A_0 + 1, \dots A_1 - 1 \rangle$$

于是，余积 $f + g$： $A_0 + A_1 \to B$（记作 $f + g$）由数组连接给出：

$$f + g = \langle f_0, f_1 \dots f_{A_0 - 1}, g_0, g_1 \dots g_{A_1 - 1} \rangle$$

证明。 对 $f + g$ 作此选择后，它与单射相容：

$$(\iota_0 \fatsemi (f + g)_i = (f + g)_{\iota_{0_i}} = [f_0, f_1, \dots, f_{A_0 - 1}]_i = f_i$$

and

$$(\iota_1 \fatsemi [f, g])_i = [f, g]_{\iota_{1_i}} = [g_0, g_1, \dots, g_{A_1 - 1}]_i = g_i$$

此外，这一选择必须是唯一的：如果数组 $f + g$ 中哪怕有一个元素与上述规定不符，那么该图表就不交换。 □

映射 f： $A_0 \to B$ 和 g： $A_1 \to B$ 的余积 $f + g$： $A_0 + A_1 \to B$ 实现为数组连接。

```
def coproduct(f, g):
  assert f.target == g.target
  target = f.target
  table = concatenate(f.table, g.table)
  return FiniteFunction(target, table)
```

自然地，由于 FinOrd 是余完备的，并且我们声称 FinOrd ≅ FinFun，因此我们期望 FinFun 也具有余等化子，从而同样是余完备的。事实确实如此；下面给出一个众所周知的结果。

命题 2.20 (FinFun中的余等化子)。 令 $f, g: A \to B$ 是 FinFun 中的平行箭头，并令G 为由顶点和边 $\{(f(i)\ g(i)) \mid i \in A\}$构成的图。若 Q 是 G的连通分支数，且 $q : B \to Q$ 是将顶点标记为其所属连通分支的函数，那么 $q = \mathsf{c}(f, g)$ 是 f, g的余等化子

余等化子可以直接通过连通分支算法计算。为了便于实现，我们假设存在一个原语函数 `connected_components` $: B^A \times B^A \to Q \times Q^B$，它可以根据以邻接表编码的图计算连通分支。也就是说，它的两个参数分别是边源和边目标的数组。然后，我们可以如下实现有限函数的余等化子。

```
def coequalizer(f, g)
    assert f.source == g.source
  assert f.target == g.target
   q = connected_components(f.table, g.table)
  return FiniteFunction(Q, q)
```

Pro命题 2.21 （余等化子复杂度）。计算有限函数 f, g 的余等化子 $: A \to B$ 具有 $O(A + B)$ 顺序复杂度，以及 $O(\log(A + B)$ *PRAM CRCW* 时间复杂度。

证明。显然，计算余等化子的复杂度与计算连通分支的复杂度相同。在顺序情形下，对于一个有 B 个顶点和 A 条边的图 G，可以在 $O(V + E)$ 时间内标记连通分支（例如参见 [10, Chapter 22]）；而在并行情形（PRAM CRCW）下，则可以在 $O(\log\ V)$ 时间内完成（参见 [19, p. 218] or [24]）。因此，计算 FinFun 中 $f, g: A \to B$ 的余等化子，其顺序时间复杂度为 $O(A + B)$，并行（PRAM CRCW）时间复杂度为 $O(\log(A + B))$。 □

始对象和余积赋予 FinFun 严格幺半范畴的结构。对于态射 $f\colon A_0 \to B_0$ 和 $g\colon A_1 \to B_1$，张量积 $f \otimes g\colon A_0 \otimes A_1 \to B_0 \otimes B_1$ 定义为 $(f \fatsemi \iota_0) + (g \fatsemi \iota_1)$。不过，张量积也可以用更直接的方式表示。

命题 2.22（张量积）。设 $f\colon A_0 \to B_0$ 和 $g\colon A_1 \to B_1$ 是 FinFun 中的态射。张量积 $f \otimes g$ 是数组 $\langle f_0, f_1, \ldots f_{A_0}, (B_0 + g_0), (B_0 + g_1) \ldots (B_0 + g_{A_1}) \rangle$

证明。显然，$f \fatsemi \iota_0$ 是一个数组，且其元素与 f 相同；$g \fatsemi \iota_1$ 也是一个数组，其元素为 $\langle B_0 + g_0, B_0 + g_1, \ldots B_0 + g_{B_1 - 1} \rangle$。因此，张量积就是这两个数组的余积（串接）。 □

态射的张量积可以按如下方式实现，其时间复杂度与余积相同。

```
def tensor(f, g):
  table = concatenate([f.table, g.table + f.target])
  return FiniteFunction(f.target + g.target, table)
```

命题 2.23。FinFun 是一个严格对称幺半范畴。

证明。对称性按通常方式由 $\sigma := (\iota_1 + \iota_0) \fatsemi [\iota_1, \iota_0]$ 给出，显然是自逆的。由于数组拼接严格满足结合律，且与空数组拼接即为恒等态射，因此严格性成立。 □

容易证明（例如参见 [21]），存在一个同构 FinFun $\cong$FinOrd，它也保持这两个范畴的对称幺半结构。

2.3 余跨与结构余跨

余跨（以及近来出现的 结构化 余跨 [1]）如今常用于为“开放”系统建模。尤其是，余跨被用于表示字符串图的组合表示，这一表示首次见于 [4], ，并在下文第 2.4 节中加以说明。因此，我们现在回顾这些概念。

定义 2.24。范畴 $\mathscr{C}$ 中的一个余跨度 是一对态射 $X \xrightarrow{s} W \xleftarrow{t} Y$。我们称映射 s 和 t 为该余跨度的腿，称对象 X 和 Y 为脚，并称 W 为顶点。

结构化 余跨是近期为开放系统引入的 [1] 双范畴框架。本文的工作可以解读为透过这一框架，重新审视 [4] 中字符串图表的组合刻画。

定义 2.25 (出自 [1])。设 $\mathsf{L}\colon \mathscr{C} \to \mathscr{D}$ 是一个函子，我们称之为结构函子。结构余跨是 $\mathscr{D}$ 中的一个余跨，其形式为 $\mathsf{L}(A) \xrightarrow{s} W \xleftarrow{t} \mathsf{L}(B)$。

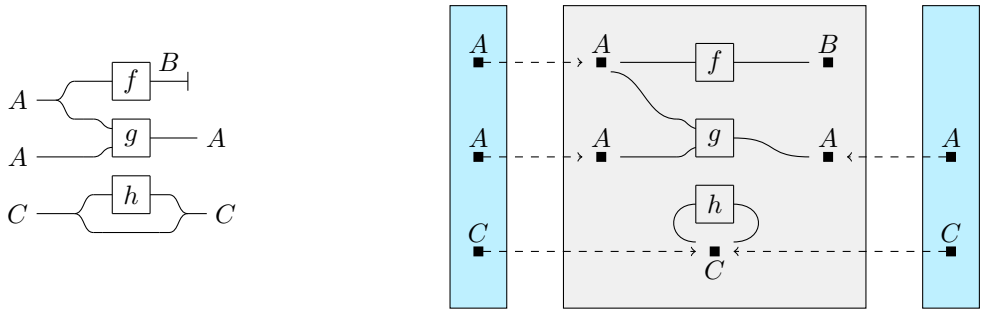
本文不要求结构余跨具备双范畴结构。不过，根据 [1,推论 3.11]，当 L 是左伴随函子且 $\mathscr{C}$ 和 $\mathscr{D}$ 具有有限余极限时，结构化余跨构成一个对称幺半范畴。该范畴中的复合通过推出实现，而张量积则是余跨各条“腿”的逐点张量积。

注 2.26。与之密切相关、但本文未采用的，是装饰余跨的构造 *[12]*。装饰余跨与结构余跨之间的关系见 *[2]*。

2.4 弦图表示的超图与二部图

[4] 的作者将弦图组合刻画为超图的余跨度。虽然本文不需要涉及技术细节，但我们在此简要概述其基本思路，以便在后续内容中参考。一个字符串图

（左）及其超图表示（右）如下图所示。

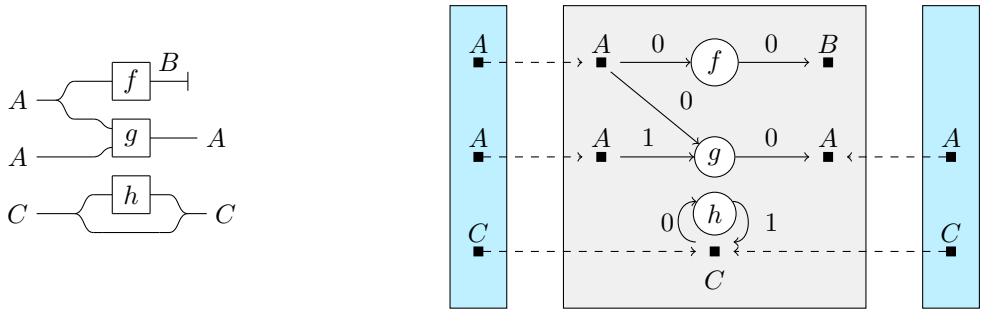


在这种表示中，超边（以白色方块表示）标记有生成元，而节点（以 ■ 表示）代表弦图中的导线。重要的是，这种表示中的超边以节点的有序列表作为其源和目标。由于这是一个余跨度，该表示实际上由三个不同的超图组成。中间的超图（灰色背景）编码了弦图的内部结构。最外层的超图（蓝色背景）是离散的，也就是说，它们只包含节点。从最外层超图到中间超图的两个态射，其映射关系以虚线箭头表示，指出了哪些节点构成该超图的左接口和右接口。这对应于弦图中的“悬空导线”，使得它们能够在左侧和右侧与其他弦图进行顺序复合。

如 [4], 所示，当弦图来自超图范畴时，已知这种表示构成范畴同构——也就是说，每个对象上都有一个特殊 Frobenius 结构。当考虑对称幺半范畴中的弦图而不附加额外结构时，只有将范围限制为所谓的单配余跨度，这种表示才是同构。单配性是对接口、节点和超边彼此相互作用方式的要求；完整定义请参阅 [4] ，我们将在第 6 节给出一个等价定义。

然而，为了在计算机上处理此类超图，必须选择如何表示其数据。例如，可以将每条超边表示为一对有序整数列表。然而，为了定义适用于例如 GPU 的并行算法，我们需要一种“扁平”的数组表示。如何高效地 编码这些超图这一问题在 [26], 中得到了探讨，作者指出，将（单偶）超图编码为二部图，可得到一种高效的稀疏矩阵表示。然而， [26] 仅限于单偶 情形，因此只适用于表示不带额外特殊 Frobenius 结构的对称幺半范畴的弦图。此外， [26]仅考虑具有单个生成对象的范畴，且所述表示并未将图表显式表示为余跨度。

我们的工作将 [26] 的方法推广到配备选定的特殊 Frobenius 结构的弦图情形。这一补充使我们能够在第10节定义一种高效的算法，用于对大型图表求反向导数。回到我们的示例，这一结构的加入使得我们能够将如下左侧所示的弦图表示为二部多重图的余跨度（如下右侧所示）。



2.5 $ACSets$

我们的主要贡献之一，是以一种能够支持 [4] 的超图结构编码方式来实现并行算法在该数据结构上运行。这得益于ACSet的构造，该构造最早在 [21]中进行了描述。简单来说，ACSet是一类表示“带属性的基于数组的数据”的范畴。例如，带有节点和边标签的图包含数据（图的邻接信息）和属性（标签）。现在，我们以非正式的方式回顾ACSet，从 [21],中的模式定义开始。

定义 2.27 （模式 [21], 非正式）。 一个模式是一个有限呈示的（小）范畴 S，它是“二部的”： S 中的对象要么属于 S_0 ，要么属于 S_1。

直观地说，模式描述了范畴中数据与属性之间的关系。具体而言， S_0 中的对象（数据）将映射到有限集，而 S_1 中的对象（属性）则映射到某种选定的类型赋值。

定义 2.28 （ACSets [21]）。 给定一个模式 S 和一个类型映射 K： $S_1 \to$ 集合，范畴 ACSet_K^S 的对象是函子 F： $S \to$ 集合，它们限制为 $F|_{S_1} = K$ ；其态射 α： $F_0 \to F_1$ 是自然变换，其中 $\alpha|_{S_1} =$id。

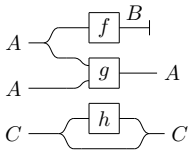
类型映射 K 定义了数据属性的类型。例如，顶点标记取自 $\mathbb{N}$ 的图，可以建模为一个ACSet，其模式有三个对象： $S_0 = \{V,\ E\}$ 和$S_1 = \{L\}$，其类型映射为 $K(L) = \mathbb{N}$。要求态射 α 必须带有 $\alpha|_{S_1} =$ id，确保 ACSets 的态射不能任意修改属性数据。

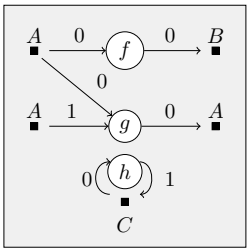
在本文中，我们只需考虑有限的 ACSets：其对象 F 限制为函子 $F|_{S_0} \to$ FinFun。请注意，这意味着我们的数据结构由稠密整数数组组成：这对于使我们的实现适用于 GPU 等并行硬件至关重要。在下一节中，我们将看到两个构成我们数据结构基础的 ACSet 示例。

3 使用 ACSet 表示图表连线

现在，我们可以开始定义用于将弦图表示为结构化余跨的数据结构。在本节中，我们定义两个范畴 Wires 和 BipartiteMultigraph，它们对弦图的“内部连线”进行建模。这两个范畴通过一个伴随 $\mathsf{L}:$ Wires $\to$BipartiteMultigraph 相关联；在我们将弦图定义为结构化余跨时，它将充当“结构函子” $\mathsf{L}(A) \xrightarrow{s} G \xleftarrow{t} \mathsf{L}(B)$ ，具体见第 4节。 Wires 的对象将充当余跨的脚，并且类似于 [4]中的离散超图。

为了更清楚地说明这两个范畴的作用，我们先来看几个例子。下方左侧展示了一个字符串图，右侧则是与其“内部连线”对应的二部多重图。



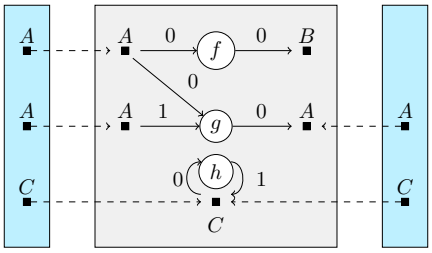


二部结构意味着节点分为两类。 ■-节点表示字符串图中的*Wires*，并以 Σ_0 的生成对象标记。与此同时，∘-节点表示运算，并以 Σ_1 中的对象标记。

图的边以自然数标记：标记为 i 的边 $\blacksquare \to \circ$ 表示一条给定导线连接到某个运算的第 i^{th} 个源端口。同样，$\circ \to \blacksquare$ 表示一条导线连接到某个运算的给定目标端口。

然而，缺少了这项信息。也就是说，并未说明哪些 ■-节点对应于字符串图的边界；换言之，哪些是左侧和右侧的“悬垂

导线”。这正是 Wires 范畴的用途，其中的对象可视为带标签有限集，而其态射是“保标签映射”。下图展示了一个 L-结构余跨。

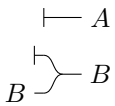


中央的灰色框表示余跨的顶点：一个二部多重图。两端是位于蓝色框中的两个 Wires 对象。它们之间的虚线箭头对应结构余跨的腿，也可以视为 Wires 的态射。

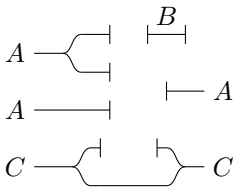
为了形式化这一构造，下面给出 Wires 和 d BipartiteMultigraph 作为 ACSets 的精确定义，以及结构函子 L。

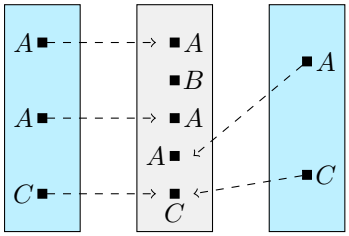
3.1 Wires：多类型有限函数的范畴

下面介绍 Wires，即“多类型”有限集与函数的范畴。与 FinOrd 一样，这些函数由 $\mathbf{CMon}(\Sigma_0)$ 表示。唯一的区别在于，生成对象 Σ_0 现在可以是任意的，而不再局限于单个对象。例如，给定生成对象 A 和 B，我们可以构造如下所示的态射。



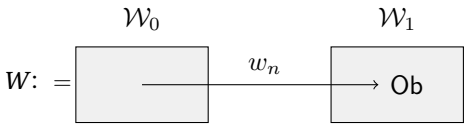
这类态射的余跨 （右下）对应于带有**Frob** 生成元的字符串图，类似于左下方所示的图。





请特别注意，现在余跨度的腿也保标记。也就是说，每个虚线箭头的源和目标具有相同的标签。这一性质是通过将 Wires 定义为标签属于属性的 ACSets 范畴而得到的。

定义 3.1. 设 Σ为任意幺半群签名。“带标签的线”的范畴记作 Wires_Σ，定义为范畴 $\mathsf{ACSet}^{\mathcal{W}}_{K_\Sigma}$，其模式 W 定义如下：



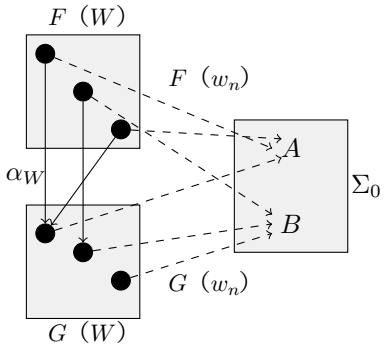
其类型映射 K_Σ 定义为

$$K_\Sigma(\mathsf{Ob}) := \Sigma_0$$

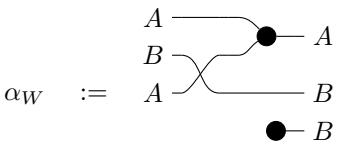
给定一个“标记函数” l： $B \to \Sigma_0$，我们用 $W(l)$ 表示 Wires 中的对象，其中 $W(l)(W) = B$且 $W(l)(w_n) = l$。给定一个有限函数 f： $A \to B$，我们用 $W(f, l)$ 表示 Wires 中的一个态射f： $W(f \fatsemi l) \to W(l)$

备注 3.2。 *Wires* 中的态射 $\alpha : F_0 \to F_1$ 是只有一个分量的自然变换，α_W。为减轻记号负担，我们有时会省略这个下标。不过，应将这些映射视为带标签有限集之间的“保标签有限函数”。

例 3.3。设 F和 G是 $Wires_\Sigma$中的对象。具体而言， F由集合 $F(W)$和态射 $F(w_n): F(W) \to \Sigma_0$构成。类似地， G是集合 $G(W)$和态射 $G(w_n): G(W) \to \Sigma_0$。 F 和 G 之间的保标签映射 α 是只有一个非恒等分量的自然变换， α_W。下面给出了 $F(W) = 3$和 $G(W) = 3$之间保标签映射的一个例子。



注意，有限函数 α_W 可以形式化地表示为下方的字符串图，其中入射导线根据 $F(w_n)$ 标记，出射导线根据 $G(w_n)$ 标记。



此示例说明，Wires 的态射确实对应于任意生成对象集合 Σ_0 上的交换幺半群理论。

我们也可以将 Wires 的对象视为离散顶点标记图。当我们定义结构函子 L 时，Wires 中的一个对象 $A \in$将映射到仅含 ■-节点的离散 二部多重图 $\mathsf{L}(A) \in$BipartiteMultigraph。

事实上，Wires 中的态射将在任意超图范畴中构成“Frobenius 半蜘蛛”。

命题 3.4. 设 $\mathscr{C}$ 是由签名 Σ呈现的超图范畴。存在唯一的严格对称幺半恒对象函子 $\mathsf{S} : \mathsf{Wires}_\Sigma \to \mathscr{C}$， 它将 **CMon** 的幺半结构映射到**Frob**的幺半结构。

证明。 回忆一下，FinOrd 由 **CMon**呈现，而 Wires 的态射是 FinOrd 中仅有一个非恒等分量的自然变换。因此，Wires 的态射也由 **CMon** 及生成对象集合 Σ_0呈现。由此，S 在所有生成元、复合和张量积上的作用均已确定，且是唯一的。 □

我们将此类态射称为“半蜘蛛”。

定义 3.5 （半蜘蛛）。 设 $\mathscr{C}$ 是由签名 Σ表示的超图范畴。$\mathscr{C}$ 中的一个***Frobenius*** 半蜘蛛 是 $\mathsf{S} : \mathsf{Wires}_\Sigma \to \mathscr{C}$的像中的一个态射。我们有时将 $\mathsf{S}(f)$ 写作一个从 $f \in \mathsf{FinFun}$ 出发的箭头，以表示 $\mathsf{S}(W(f, l))$，其中 l 由上下文可知。

或者，半蜘蛛是超图范畴中由张量积和复合从生成元 , , , 和 构造出的那些态射。此外，回顾一下，所有超图范畴都有一个伴随[13, 第 1.3 节。3]。因此，还存在第二个函子，将 $\mathsf{Free}_{\mathbf{CMon}}$ 中的态射映射到超图范畴的 余幺半群 结构： $f \mapsto \mathsf{S}(f)^\dagger$。于是，这些“伴随半蜘蛛”是由 , , , 和 构造出的态射。

将这两个函子结合起来，就刻画了超图范畴中的 Frobenius 蜘蛛。也就是说，Frobenius 蜘蛛将是形如 $\mathsf{S}(f) \fatsemi \mathsf{S}(g)^\dagger$的那些态射。这将在命题 4.7 中作出更正式的表述。

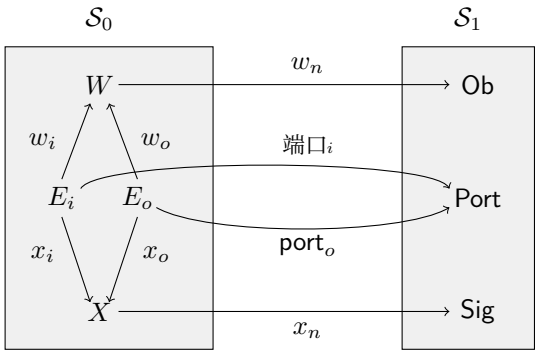
现在我们将描述二部多重图，并说明表示字符串图中运算 所需的额外结构。

3.2 二部多重图

二部多重图可以看作是附加了额外数据的 Wires 对象。更正式地说，它们由以下 ACSet 给出。

定义 3.6。 二部多重图是范畴 BipartiteMultigraph $:= \mathrm{ACSet}_K^{\mathcal{S}}$ 中的一个对象，其中 $\mathcal{S}$ 是下文定义的模式函子

$\mathcal{S} :=$

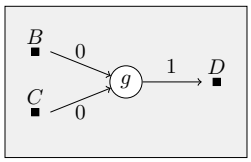


并且类型映射 $K:\ S_1 \to$ 集合 定义为

$$K(\mathsf{Ob}) := \Sigma_0 \qquad K(\mathsf{Sig}) := \Sigma_1 \qquad K(\mathsf{Port}) := \mathbb{N}$$

并非每个二部多重图都表示字符串图的有效“内部连线”。例如请注意，上述模式并未阻止这样的二部多重图：某个生成元有两条边带有相同的“端口标签”，且这两条边从生成元 ■ 连接到一个 ∘ 节点。

例 3.7。 下面的二部多重图有两处“不良构造”。



首先，有两条标记为 0 的 ■ → ∘ 边。这与将边标签解释为用于定义生成元的哪个“端口”连接到导线相冲突。其次，没有输出边

∘ → ■ 标记为 0：我们需要确保 g 的每个输出端口都有对应项。

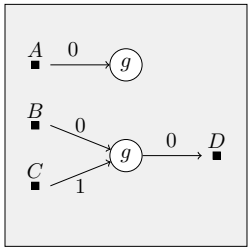
为了排除上述这类对象，我们需要讨论“良构”二部多重图。

定义 *3.8*（良构二部多重图）。对于签名 Σ，二部多重图 G 是良构的，当且仅当存在选定的类型赋值 $(a:\ X \to \Sigma_0^*, b:\ X \to \Sigma_0^*)$，使得对于每个 $x \in G(X)$，都有 $(a(x)\ b(x)) \in \tau(G(w_n)(x))$，并且 hx_i、$port_i i$ 和 hx_o、$port_o i$ 都是单射，且

$$\forall e \in E_i \quad w_n(w_i(e)) = a(x_i(e))_{\mathsf{port}_i(e)} \qquad \forall e \in E_o \quad w_n(w_o(e)) = b(x_o(e))_{\mathsf{port}_o(e)}$$

注 3.9。 定义 *3.8* 的一个推论是，在良构二部多重图 G，$G(E_i)$ 和 $G(E_o)$ 分别等于生成元 $G(X)$的总元数和余元数。

注 3.10。 上述良构性的定义明确允许多态生成元。更确切地说，由于运算 $g \in \Sigma_1$ 具有类型关系，例如，我们可以得到如下所示的二部多重图。



请注意，有两个标记为 g 且类型不同的 $\circ$-节点：一个是 $g:\ A \to I$，另一个是$g:\ B \otimes C \to D$。只要这两种类型都存在于类型关系中，这样的图表就被视为良构。

在良构二部多重图的结构余跨上定义算法时，我们需要以下保持定理。这将保证由良构图表构成的结构余跨的复合仍然是良构的。

命题 3.11（良构性的保持）。如果一个二部多重图的态射$\alpha:\ F_0 \to F_1$ 除 α_W之外的所有分量都是置换，且 F_0 是良构的，那么 F_1 也是良构的。

证明。注意，边、生成元及其属性数据的数量都得以保持。此外，根据自然性，$\blacksquare$-节点的属性也得以保持，因此 F_1 是良构的。 □

最后，在将弦图定义为结构余跨之前，我们首先将范畴 Wires 与 BipartiteMultigraph 之间的关系形式化为一个伴随。注意，Wires 是 BipartiteMultigraph 的一个子范畴，因此可以嵌入其中。

定义 3.12。将L: Wires $\to$ BipartiteMultigraph 记为（对象上恒等的）包含函子。

如 [21, 第18], 页所述，L 是遗忘函子的左伴随。

命题 3.13 (L 是左伴随)。设R : BipartiteMultigraph $\to$ Wires 为遗忘函子，它将二部多重图映射到其导线集合及其标记。则L 是R的左伴随。

证明。设 α : L(A) $\to G$ 是 BipartiteMultigraph 中的一个箭头，并设 ϵ_G : L(R(G)) $\to G$ 是一个自然变换，其分量定义如下。

$$\epsilon_{G_Y} = \begin{cases} \mathsf{id}_W & \text{if } Y = W \\ ? & \text{otherwise} \end{cases}$$

其中 ? $=$ 是初始映射。于是存在唯一的 f 使得 L(f) $\fatsemi \epsilon_G = \alpha$。由于 BipartiteMultigraph 中的态射复合是逐点进行的，对于每个分量 Y，必有 (L(f)$\fatsemi\epsilon_G$)$_Y = \alpha_Y$。因此必有 L(f)$_W = \alpha_W$，且 L(f)$_Y$ $=$?$:\ 0 \to 0$，所以根据自然性有$f = \mathcal{W}(\alpha_W, G(w_n))$。 □

左伴随性L 表明，任意形如 α : L(A) $\to G$ 的二部多重图态射，都完全由其 W 分量 α_W : L(A)(W) $\to G(W)$决定。在第4节定义弦图时，这一事实将使我们能够用FinFun的余等化子来定义复合。此外，L 像中的二部多重图在下一节中尤为重要。因此，我们引入以下记号。

定义 3.14（离散二部多重图）。给定一个标记 $w:\ A \to \Sigma_0$，对于 w 的离散二部多重图记作 D(w):= L(W(w))。给定一个二部多重图 G，我们沿用相同的记号，写作 D(G) :=L(R(G))。

有了这些定义，我们终于可以将弦图定义为结构化余跨。

4 弦图作为结构化余跨

现在，我们可以将弦图范畴定义为结构化余跨。

定义 4.1 (图表)。固定一个幺半群签名 Σ。关于 Σ 的图表 是一个 L-结构余跨 L(A) $\xrightarrow{s} G \xleftarrow{t}$ L(B)，其中 G 关于 Σ 是良构的。我们称此类图表的类型 为 $A \to B$。

请注意，如 [1, 推论 3.11] 所述，图表构成一个对称幺半范畴。这是因为 L 是左伴随，因此保持余极限。

定义 4.2。 图表的对称幺半范畴记为 $\mathsf{Diagram}_{\Sigma+\mathbf{Frob}}$，定义为 $_{\mathsf{L}}\mathsf{Csp}(\mathsf{BipartiteMultigraph})$ 中定义于 *[1, 推论 3。11]* 其对象为 Wires 的对象，其态射为图表的同构类。

为了定义 图表$_{\Sigma+\mathbf{Frob}}$ 作为一个 子范畴，重要的是验证复合和张量积能够保持图表的良构性。这由命题 3.11推出，我们将在推论 4.19中给出证明。

结构化余跨的复合通过推出实现，而张量积是逐点的；这两者的数据并行算法将分别在命题 4.16 和4.11中给出。事实上，本节的其余部分专门用于展示：（1）如何高效地表示图表；（2）如何构造恒等态射和对称性等“基本原语”图表；以及（3）张量积和复合的数据并行算法。此外，本节还旨在作为实现所述算法的指南。因此，在若干情况下，我们还列出了多个伪 Python 代码清单。

4.1 高效表示图表

定义的朴素表示 4.1需要为余跨度的每个脚都构造一个完整的二部多重图，并为腿构造一个完整的二部多重图态射。然而，为了表示大型图表，节省数据至关重要。因此，我们利用 L 是左伴随这一性质，证明其中许多数据都是冗余的。这也将简化本文其余部分中若干算法的定义。

命题 4.3。 图表与三元组 (s, t, G) 之间存在双射对应，其中 $A \xrightarrow{s} G(W) \xleftarrow{t} B$ 是 FinFun 中的一个余跨度。

证明。我们首先证明，任意图表都唯一确定这样的三元组。设 $\mathsf{L}(A) \xrightarrow{\sigma} G \xleftarrow{\tau} \mathsf{L}(B)$ 是一个图表。由于 L 是左伴随（命题 3.13），存在唯一的态射 s, t of Wires_Σ，使得 $\mathsf{L}(s) \fatsemi \epsilon_G = \sigma$ 且 $\mathsf{L}(t) \fatsemi \epsilon_G = \tau$。由于它们是 Wires_Σ 的态射，s 和 t 是自然变换，且只有一个定义在 W 上的非恒等分量。因此，我们可以将它们视为 FinFun 的态射。

反过来，假设 (s, t, G) 是一个三元组，其中 $A \xrightarrow{s} G(W) \xleftarrow{t} B$ 是 FinFun 中的一个余跨度。以下述方式定义图表 $\mathsf{L}(A) \xrightarrow{\sigma} G \xleftarrow{\tau} \mathsf{L}(B)$。我们必须取$\sigma = \mathsf{L}(\mathcal{W}(s, G((w_n)))) \fatsemi \epsilon_G$ 和 $\tau = \mathsf{L}(\mathcal{W}(t, G(w_n))) \fatsemi \epsilon_G$ □

这项对应关系意味着，图表可以用代码写成如下的 `Diagram`类。

```
class Diagram:
  s: FiniteFunction
  t: FiniteFunction
  G: BipartiteMultigraph
```

现在我们将使用这种替代表示来定义基本图表，以及张量积和复合的算法。请注意，本节余下部分我们将采用 2-范畴的视角，考察具体的图表（即 1-胞腔），而不是图表的同构类。从使用图表的程序员的角度来看，这种做法更为自然，因为人们通常使用同构类的一个代表，而非同构类本身。

4.2 基本图表

命题 4.4。 给定一个对象 $\mathcal{W}(w) \in \mathsf{Wires}$，其 恒等图表 由三元组表示 $(\mathsf{id}, \mathsf{id}, \mathsf{D}(w))$ 其中 $\mathsf{D}(w)$ 是离散二部多重图（定义 *3.14*）。

证明。 对于 Wires 的对象 $A = \mathcal{W}(w)$，恒等结构余跨是 $\mathsf{L}(A) \xrightarrow{\mathsf{id}} \mathsf{L}(A) \xleftarrow{\mathsf{id}} \mathsf{L}(A)$。因此我们必须有 $s = t = \mathsf{id}$，而根据定义，$\mathsf{L}(A) = \mathsf{D}(\mathsf{L}(A))$。 □

在代码中，恒等图表的构造方式如下。

```
@classmethod
def identity(cls, wn: FiniteFunction):
    s = FiniteFunction.identity(wn.source)
    t = FiniteFunction.identity(wn.source)
    G = BipartiteMultigraph.discrete(wn)
    return Diagram(s, t, G)
```

图表Σ+Frob 的对称性如下所示。

命题 4.5 (对称/扭转图)。 设 $a\colon A \to \Sigma_0$ 和 $b\colon B \to \Sigma_0$ 为标记，则对称图由 $(\mathsf{id}_{A+B}, \sigma_{A,B}, \mathsf{D}(a+b))$给出。

证明。 对称结构余展由 $\mathsf{L}(A+B) \xrightarrow{\mathsf{id}} \mathsf{L}(A+B) \xleftarrow{\sigma_{A,B}} \mathsf{L}(B+A)$给出，因此结论如命题 4.4所示。 □

图表Σ+Frob 也是一个超图范畴，这意味着每个对象都配备了一个特殊弗罗贝尼乌斯幺半群。这就是我们将范畴记作图表Σ+Frob的原因。在第 6 节中，我们将看到，如何利用 单配性 这一组合条件（在 [4] 中引入），无需这种额外结构即可定义范畴 图表Σ。

命题 4.6。 图表Σ+Frob 是一个超图范畴

证明。 L 是左伴随，因此保余极限。根据 [1, , DiagramΣ+Frob 因此是一个超图范畴。定理 3.12] □

Frobenius 蜘蛛（定义2.8）在 DiagramΣ+Frob 中可以明确地刻画为顶点是没有运算的离散超图的余跨度。也就是说，形如$(s,\ t,\ \mathsf{L}(A))$。

命题 4.7。 图表 $d = (s, t, G)$ 是 *Frobenius* 蜘蛛，当且仅当 $G = \mathsf{L}(A)$ 对某个 A 成立 .

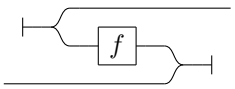
证明。 证明见附录 C □

在代码中，我们提供了如下构造 Frobenius 蜘蛛的构造器。

```
def spider(s, t, w):
  G = BipartiteMultigraph.discrete(w)
  return Diagram(s, t, G)
```

作为超图范畴，DiagramΣ+Frob 按如下方式配备了伴随 .

命题 4.8. 超图范畴具有如下伴随：



证明. 参见 [13, 第 1.3 节。3]。 □

图表的伴随形式尤为简洁：它交换余跨度的两条腿。也就是说，$(s,\ t,\ G)^\dagger = (t,\ s,\ G)$。我们将在命题 C.5中证明这一事实，不过这里先指出，它带来了如下简单的实现。

```
def dagger(self):
  return Diagram(self.t, self.s, self.G)
```

我们需要定义的最后一个“基本原语”图表是单例，它表示单个生成元 or of Σ_1.

定义 4.9 (单例图表)。 设 $x\ :\ 1 \to \Sigma_1$ 是一个运算，且 $a\ :\ A \to \Sigma_0$ 和$b\colon B \to \Sigma_0$是 x 的类型赋值。 x 的 单例图表记作 $\texttt{singleton}(a, b, x)\colon = (\iota_0, \iota_1, G)$，其中

$$G(W) = A + B \qquad G(E_i) = A \qquad G(E_o) = B \qquad G(X) = 1$$

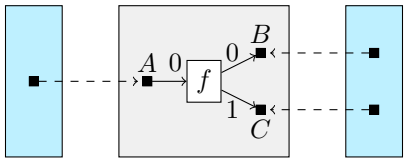
and

$$G(w_i) = \iota_0 \qquad G(w_o) = \iota_1 \qquad G(x_i) = !_A \qquad G(x_o) = !_B$$

$$G(\mathsf{port}_i) = \mathsf{id}_A \qquad G(\mathsf{port}_o) = \mathsf{id}_B \qquad G(w_n) = a + b \qquad G(x_n) = x$$

若已假定选定了一种类型赋值，我们有时只写作 $\texttt{singleton}(x)$ 。

示例 4.10. 给定一个生成元 $f\colon A \to B \otimes C$，表示为元素 $x = \hbar f i$，并且类型赋值为$a = \hbar A i$ 和 $b = \hbar B, C i$，单例图表 $\texttt{singleton}(x)$ 如下图所示。



此定义可直接转换为如下代码：

```
@classmethod
def singleton(cls, a: FiniteFunction, b: FiniteFunction, xn: FiniteFunction):
    G = BipartiteMultigraph(
        wi = F.inj0(a.source, b.source),
    wo = F.inj1(a.source, b.source),
    xi = F.terminal(a.source),
    xo = F.terminal(b.source),wn = a + b,
    pi = F.identity(a.source),
    po = F.identity(b.source),xn = xn)
    return Diagram(
    s=FiniteFunction.inj0(a.source, b.source),
    t=FiniteFunction.inj1(a.source, b.source),G=G)
```

4.3 图表的张量积

命题 4.11 (图的张量积)。 图表 $c_0 := (s_0, t_0, G_0)$ 与 $c_1 := (s_1, t_1, G_1)$ 的张量积是图表 $c_0 \otimes c_1 := (s_0 \otimes s_1, t_0 \otimes t_1, G_0 \otimes G_1)$

证明。 设 $\mathsf{L}(A_0) \xrightarrow{\sigma_0} G_0 \xleftarrow{\tau_0} \mathsf{L}(B_0)$ 和 $\mathsf{L}(A_1) \xrightarrow{\sigma_1} G_1 \xleftarrow{\tau_1} \mathsf{L}(B_1)$ 分别是与 c_0 和 c_1 相对应的结构化余跨，根据命题 4.3。那么，张量积 $c_0 \otimes c_1$ 是如下结构化余跨。

$$\mathsf{L}(A_0 \otimes A_1) \xrightarrow{\sigma_0 \otimes \sigma_1} G_0 \otimes G_1 \xleftarrow{\tau_0 \otimes \tau_1} \mathsf{L}(B_0 \otimes B_1)$$

由于余极限是逐点的，我们有 $(\sigma_0 \otimes \sigma_1)_W = s_0 \otimes s_1$ 和 $(\tau_0 \otimes \tau_1)_W = t_0 \otimes t_1$ ，因此，张量积 $c_0 \otimes c_1$ 的表示为 $(s_0 \otimes s_1, t_0 \otimes t_1, G_0 \otimes G_1)$ □

再一次，这给出了一个简单直接的实现：

```
def tensor(c0, c1):
return Diagram(
s = c0.s @ c1.s,
t = c0.t @ c1.t,
G = c0.G @ c1.G)
```

张量积的顺序时间复杂度随图表分量线性增长（顺序），随图表分量对数增长（并行）。

命题 4.12。设 $d_i = (s_i, t_i, G_i)$ 是类型为 $A_i \to B_i$ 的图表，其中 $i \in \{0,1\}$。计算张量积 $(s, t, G) = d_0 \otimes d_1$ 的顺序时间复杂度为

$$O(A_0(W) + A_1(W)) + O(G(W)) + O(G(E_i)) + O(G(E_o)) + O(G(X)) + O(B_0(W) + B_1(W))$$

并行随机存取机时间复杂度为 $O(1)$。

证明。计算二部多重图的余积，其顺序时间复杂度为 $O(G(W))+O(G(E_i)) + O(G(E_o)) + O(G(X))$，因为本质上只需拼接数组。计算同一余积的并行随机存取机时间复杂度为 $O(1)$。接着，计算 $s_0 + s_1$ 的顺序时间复杂度为 $O(A_0(W) + A_1(W))$，并行随机存取机时间复杂度为 $O(1)$；$t_0 + t_1$的情况也类似。□

之后，我们会需要一种高效的方法来对大量运算进行张量积。特别是，简单地重复二元张量运算会导致较差的复杂度：顺序情形下为二次，平行情形下为线性。因此，我们给出以下定理，它为 n 个运算的张量积提供了更简单的闭式。

命题 *4.13.* 设 $x : N \to \Sigma_1$ 是一个由 N 个运算组成的列表，并且对每个运算 $x(i)$，选定一个类型赋值 $(a(i)\ b(i)) \in \tau(x)$，使得 $a, b\colon N \to \Sigma_0^*$。分别令 $|a(i)|$ 和 $|b(i)|$ 表示 $x(i)$ 的元数和余元数。那么，运算 x 的 N 重张量积同构于图表 (ι_0, ι_1, G)，其中

$$G(W) = K_i + K_o \qquad G(E_i) = K_i \qquad G(E_o) = K_o \qquad G(X) = N$$

and

$$G(w_i) = \iota_0 \qquad G(w_o) = \iota_1 \qquad G(x_i) = \bigotimes_{i \in N} !_{|a(i)|} \qquad G(x_o) = \bigotimes_{i \in N} !_{|b(i)|} \tag{8}$$

$$G(\mathsf{port}_i) = \left(\sum_{i \in N} (\iota_0 : |a(i)| \to K_i)\right) \fatsemi \iota \qquad G(\mathsf{port}_i) = \left(\sum_{i \in N} (\iota_0 : |b(i)| \to K_o)\right) \fatsemi \iota$$

$$G(w_n) = \sum_{i \in N} (j \mapsto a(i)_j) + \sum_{i \in N} (j \mapsto b(i)_j) \qquad G(x_n) = x$$

其中：

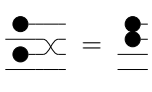
- $K_i = \sum_{i \in N} |a(i)|$ 是所有生成元的总元数
- $K_o = \sum_{i \in N} |b(i)|$ 是所有生成元的总余元数
- $!_k : k \to 1$ 是唯一的终对象映射。
- $\iota_0 \colon A \to A + B$ 是 FinFun 中余积的第一个单射
- ι 表示有限集到 $\mathbb{N}$的典范包含

证明。归纳法。空（0 重）图表和单例（1 重）图表已经具有上述形式，因此只需检查归纳步骤。

设 $c = (\iota_0,\ \iota_1,\ G)$ 是单例图表 $x\colon N \to \Sigma_1$ 的 N 重张量积。根据归纳假设，$c \cong \bigotimes_{i \in N} \mathtt{singleton}(x(i))$。现在设 $s = (\iota_0,\ \iota_1,\ H)$ 是一个单例图表，其类型赋值为 $(a' : 1 \to \Sigma_0^*,\ b' : 1 \to \Sigma_0^*)$。只需构造结构余跨的同构 $\alpha\colon c \otimes s \leftrightarrow d$，其中 $d = (\iota_0,\ \iota_1,\ J)$ 具有所需形式。

定义 α 很直接。令 $\alpha_W := =$，并将所有其他分量设为恒等态射。然后可以计算出，w_i 和 w_o 的像是单射，进一步可知 α 是结构余跨的一个同构。例如，

$$\alpha_{E_o} \fatsemi (G(w_o) + H(w_o)) \fatsemi \alpha_W =$$



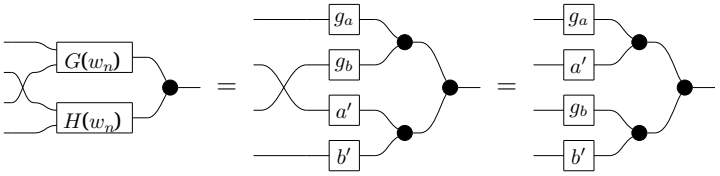
$$= \iota_1$$

其他情形同理。

最后，根据自然性，我们必须有 $J(w_n) = \alpha_W^{-1} \fatsemi G(w_n) \otimes H(w_n)$，因此只需验证这个态射具有所需的形式。

注意，$G(w_n)$ 和 $H(w_n)$ 都是映射的余积：根据定义，$H(w_n) = a' + b'$ 和 $G(w_n) = g_a + g_b$，其中 $g_a = \sum_{i \in N}(j \mapsto a(i)j)$ 和 $g_b = \sum_{i \in N}(j \mapsto b(i)j)$。对于某个 a，$b\colon N \to \Sigma_0^*$。然后利用结合律和交换公理进行如下计算。

$G'(w_n) = \alpha_W^{-1} \fatsemi (G(w_n) + H(w_n)) =$



于是我们有 $g_a + a' = \sum_{i \in N}(j \mapsto a(i)j) + (j \mapsto a'(0)j)$ 和 $g_b + b' = \sum_{i \in N}(j \mapsto b(i)j) + (j \mapsto b'(0)j)$，从而 $J(w_n) = \sum_{i \in N+1}(j \mapsto (a + a')(i)j) + \sum_{i \in N+1}(j \mapsto (b + b')(i)j)$，如所需。 □

计算运算的 N 重张量积，其所需时间相对于结果图表的大小为线性（顺序）和对数级（并行），如下述命题所示。

命题 4.14. 设 $x\colon N \to \Sigma_1$ 是运算，$d = (\iota_0, \iota_1, G)$ 是根据命题 *4.13* 定义的它们的 N 重张量积。计算 d 的顺序时间复杂度为 $O(G(W)) + O(G(E_i)) + O(G(E_i)) + O(G(X))$，而 *PRAM CREW* 时间复杂度为 $O(\log G(X))$

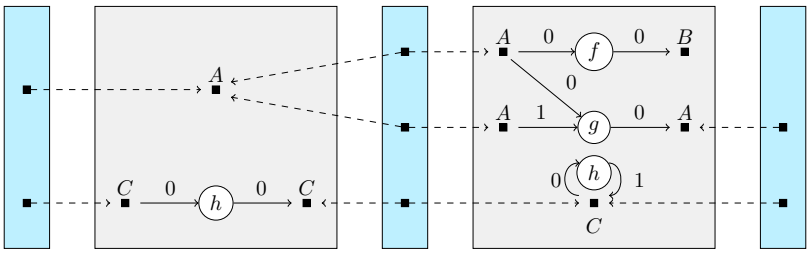
证明。计算 ι_0 和 ι_1 在顺序情形下分别需要 $O(G(E_i))$ 和 $O(G(E_o))$，在并行情形下则为常数时间。随后，在顺序情形下计算 G 本质上只需拼接数组，因此其复杂度相对于各分量的大小为线性，即 $O(G(W)) + O(G(E_i)) + O(G(E_o)) + O(G(X))$。注意，无需计算 $G(x_n)$ 分量，因为它是已知的。

在并行情形下，假设类型赋值 a 和 b 的数据以分段数组的形式提供，就可以使用 `repeat` 和 `segmented arange` 函数计算分量，从而得到 $O(\log N) = O(\log G(X))$ 的时间复杂度。 □

4.4 图表的复合

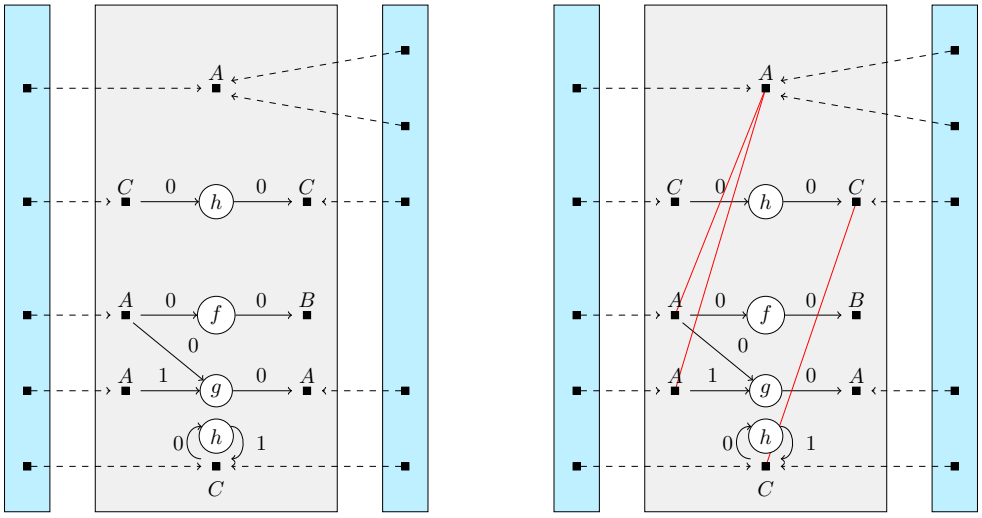
在给出图表复合的算法之前，我们先通过一个例子来说明。图表Σ+**Frob** 中的复合通过推出实现，具体由余等化子计算得出。具体来说，复合图表 $d_0 \fatsemi d_1$时，基本思路是先取它们的张量积，再对与共享边界对应的导线取余等化子。

例4.15。设 $d_0 = (s_0, t_0, G_0)$ 和 $d_1 = (s_1, t_1, G_1)$ 为下方以余跨度表示的图表。

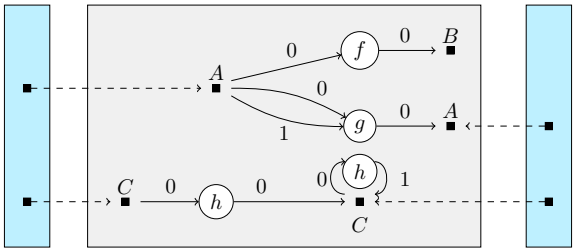


The 复合过程通过以下步骤说明。我们先对 d_0 和 d_1 取张量积（左 *t)*

然后将它们之间由红色箭头连接的 ■节点识别为同一节点（右侧）。



最后，我们将 ■-节点图中的所有连通分支取商，并取 d_0 的源映射以及 d_1 的目标。



这相当于对映射 $t_0 \fatsemi \iota_0$ 和 $s_1 \fatsemi \iota_1$ 进行余等化。

现在我们来形式化地描述这一过程。

命题 *4.16*（图的复合）。设 $d_0 := (s_0, t_0, G_0) : A_0 \to A_1$ 和 $d_1 := (s_1, t_1, G_1) : A_1 \to A_2$ 为图表，并将余等化子记为 $q := \mathrm{c}\left(\begin{smallmatrix} t_0 \\ \bullet \end{smallmatrix}, \begin{smallmatrix} \bullet \\ s_1 \end{smallmatrix} \right)$，使得 $q : G_0(W) + G_1(W) \to Q$。那么复合 $d_0 \fatsemi d_1 = (s, t, G)$ 定义为

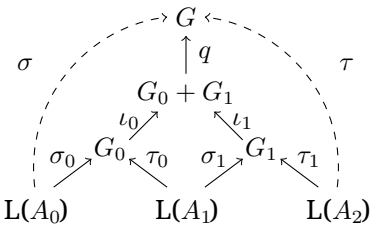
$$s = \quad \quad t = \quad \quad G = \mathsf{target}(\alpha)$$

其中 $\alpha : \ G_0 + G_1 \to G$ 是余等化子 W 分量为 q、其余所有分量均为 恒等的 ACSet，因此 G 是具有以下数据的函子

$$\begin{array}{ccccc}
 & & \Sigma_0 & & \\
 & & \uparrow{\scriptstyle G(w_n) = u} & & \\
 & & G(W) = Q & & \\
 & \nearrow{\scriptstyle (G_0(w_i) \otimes G_1(w_i)) \fatsemi q = G(w_i)} & & \nwarrow{\scriptstyle G(w_o) = (G_0(w_o) \otimes G_1(w_o)) \fatsemi q} & \\
\mathbb{N} \xleftarrow{G_0(\mathsf{port}_i) + G_1(\mathsf{port}_i)} & G_0(E_i) + G_1(E_i) & & G_0(E_o) + G_1(E_o) & \xrightarrow{G_0(\mathsf{port}_o) + G_1(\mathsf{port}_o)} \mathbb{N} \\
 & \searrow{\scriptstyle G_0(x_i) \otimes G_1(x_i) = G(x_i)} & & \swarrow{\scriptstyle G(x_o) = G_0(x_o) \otimes G_1(x_o)} & \\
 & & G(X) = G_0(X) + G_1(X) & & \\
 & & \downarrow{\scriptstyle G(x_n) = G_0(x_n) + G_1(x_n)} & & \\
 & & \Sigma_1 & &
\end{array} \tag{9}$$

并且 $G(w_n) = u$ 是余等化子 q的典范态射，即满足以下定义的唯一函数： $u(q(i)) = (G_0(w_n) + G_1(w_n))(i)$。

证明。 令 $\mathsf{L}(A_0) \xrightarrow{\sigma_0} G_0 \xleftarrow{\tau_0} \mathsf{L}(A_1)$ 和 $\mathsf{L}(A_1) \xrightarrow{\sigma_1} G_1 \xleftarrow{\tau_1} \mathsf{L}(A_2)$ 分别是由命题 d_0 和 d_1 4.3 给出的对应结构余跨4.3。余跨的复合 $d := d_0 \fatsemi d_1$ 由推出给出，我们通过余等化子 $q = \mathsf{c}(\tau_0 \fatsemi \iota_0, \sigma_1 \fatsemi \iota_1)$ 如下计算。

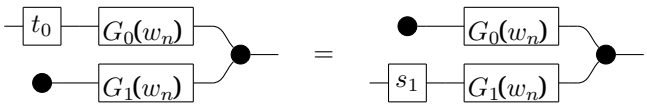


注意，这使得 $G = G_0 +_{\mathsf{L}(A_1)} G_1$。回忆一下，ACSets 中的复合和余极限是逐点的 [21,推论 4],，因此我们可以计算 $\sigma_W = (\sigma_0 \fatsemi \iota_0 \fatsemi \alpha)_W = s_0 \fatsemi \iota_0 \fatsemi \mathsf{c}(t_0 \fatsemi \iota_0)$。此外，由于 L 是左伴随，我们可以分解 $\sigma = \mathsf{L}(s) \fatsemi \epsilon_G$，并且由于 $\epsilon_{G_W} = \mathsf{id}$，我们必有 $s = s_0 \fatsemi \iota_0 \fatsemi \mathsf{c}(t_0, s_1)$。通过类似的论证，我们有 $\tau = \mathsf{L}(t) \fatsemi \epsilon_G$，其中 $t = t_1 \fatsemi \iota_1 \fatsemi \mathsf{c}(t_0, s_1)$。我们还可以看到， α 确实是余等化子：

$$\alpha_W = \mathsf{c}(\tau_0 \fatsemi \iota_0, \sigma_1 \fatsemi \iota_1)_W = \mathsf{c}(t_0 \fatsemi \iota_0, s_1 \fatsemi \iota_1) = q$$
$$\alpha_Y = \mathsf{c}(\tau_0 \fatsemi \iota_0, \sigma_1 \fatsemi \iota_1)_Y = \mathsf{id}$$

前者成立是因为 ACSets 的余极限是逐点计算的；后者成立是因为对于每个 $Y \neq W$，根据唯一性，$(\tau_0 \fatsemi \iota_0)_Y$ 和 $(\sigma_1 \fatsemi \iota_1)_Y$ 都必须等于初始映射，而初始映射的余等化子就是恒等态射。

最后，我们必须验证 u 确实是余等化子的典范态射（因此定义良好）。为此，我们必须满足 $q' = G_0(w_n) + G_1(w_n)$ 构成一个余叉：



该等式由余单位律成立；又因为 s 和 t 是保标签自然变换 σ 和 τ的 W 分量，因此 $t_0 \fatsemi G_0(w_n) = A_1(w_n) = s_1 \fatsemi G_1(w_n)$。于是，根据泛性质，存在唯一态射 $u : G(W) \to \Sigma_0$，使得 $q \fatsemi u = G_0(w_n) + G_1(w_n)$。由于 u 是唯一的，因此我们可以将其定义为函数 $u(q(i)) := q'(i)$。更直接地，如果 $q(i) = q(j)$，那么由唯一 u 的存在性可知，必有 $q'(i) = q'(j)$，因此 u 是良定义的。

□

注 4.17. 上述命题指出，在复合的情况下，只需考虑图表的导线，就能计算出余等化子（因而也能计算出推出）。也就是说，总可以选择一个余等化子，它只对二部图中的导线节点取商，而其他数据保持不变。

计算图表复合的代码很简单，

```
def compose(f, g):
    assert f.type[1] == g.type[0]
    h = f @ g
    q = f.t.inject0(g.G.W).coequalizer(g.s.inject1(f.G.W))

    return Diagram(
      s = f.s.inject0(g.G.W) >> q,
      t = g.t.inject1(f.G.W) >> q,
      G = h.G.coequalize_wires(q))
```

我们省略了 `coequalize_wires` 上文所述的实现，其计算方式如(9)所示。不过，我们现在给出计算泛同构 u的显式数据并行实现。

```
def universal(q: FiniteFunction, f: FiniteFunction):
  u = zeros(q.target)    # preallocate the table of u
  u[q.table] = f.table   # make the table of u such that q ; u = f
  return FiniteFunction(f.target, u)
```

请注意，在上述代码清单中，我们假设采用 PRAM CRCW 模型，以实现 $O(1)$ 并行时间复杂度。具体而言，赋值 `u[q.table] = f.table` 允许对同一记忆位置进行多次冲突写入，其中任意一次写入都可能成功。不过请注意，由于 q 和 f 都必须是保标签映射，所有"冲突"都必须写入相同的值，从而确保实现正确。

正如预期，复合的复杂度与图表大小呈线性关系（顺序）和对数关系（并行）。

命题 4.18。 设 $d_i = \mathsf{L}(A_i) \xrightarrow{s_i} G_i \xleftarrow{t_i} \mathsf{L}(A_{i+1})$ 是 $i \in \{0,1\}$ 的图表。设 $(s,t,G) = d_0 \otimes d_1$，则计算复合 $d_0 \fatsemi d_1$ 的顺序时间复杂度为

$$O(A_0(W)) + O(A_1(W)) + O(A_2(W)) + O(G(W)) + O(G(E_i)) + O(G(E_o)) + O(G(X))$$

以及 *PRAM CRCW* 时间复杂度 $O(\log A_1(W) + G(W))$

证明。时间复杂度为张量积的时间复杂度，加上计算余等化子并将其应用于 G 的额外开销。在顺序情形下，这部分额外开销为 $O(A_1) + O(G(W))$. 在 PRAM CRCW 情形下，为 $O(\log A_1(W)) + O(\log G(W))$.

□

最后，我们验证具有良构顶点的图表的张量积和复合也同样良构。这确保了 Diagram $_{\Sigma+\mathbf{Frob}}$ 确实构成一个（子）范畴。

推论 4.19。 如果图表 d_0 和 d_1 具有良构的顶点，那么张量积 $d_0 \otimes d_1$和复合 $d_0 \fatsemi d_1$也具有良构的顶点。

证明。张量积的情形是显然的，因为 d_0 和 d_1 的边数据都包含于 $d_0 \otimes d_1$。对于复合 $d_0 \fatsemi d_1$，注意余等化子 $\alpha : G_0 + G_1 \to G$ 除 W 之外的所有分量都等于恒等态射。因此根据命题 3.11，我们可以得出 G 也是良构的。

□

5 图表作为组合语法

现在我们构造一个显式同构图表$_{\Sigma+\mathbf{Frob}} \cong$ 自由$_{\Sigma+\mathbf{Frob}}$。这将证明我们的论断： 图表$_{\Sigma+\mathbf{Frob}}$ 的态射是签名 $\Sigma + \mathbf{Frob}$上的弦图。具体而言，这将保证 $(\Sigma + \mathbf{Frob})$-项的等价类与图表的等价类相同。有了这个同构，我们便可以将图表视为一种以组合方式表示语法的结构。在第 6节，我们将把单配性这一概念引入我们的框架，从而将其扩展为同构图表$_{\Sigma} \cong$自由$_{\Sigma}$； 单配性这一概念最初由 [4] 提出。

我们构造同构的方法如下：

- 定义将 Σ-项映射为图表的函子$\mathcal{S}$ 。
- 定义图表的"Frobenius 分解"
- 定义一个函子$\mathcal{D}$ ，使用 Frobenius 分解将图表映射到 Σ-项。
- 证明$\mathcal{S}$ 和$\mathcal{D}$ 构成一个同构

值得特别注意的是我们即将定义的"Frobenius 分解"。这种分解不仅有助于构造同构，也有助于将函子应用于图表。第9节将介绍一种基于这种分解的算法。

将 Σ项映射到图表中非常简单。我们只需将生成元 Σ_1 映射到相应的单例图表（定义 4.9），再归纳地构造图表。更正式地说，我们可以构造如下函子。

定义 5.1（$\mathcal{S}$）。$\mathcal{S}$：自由$_{\Sigma+\mathbf{Frob}}$ → 图表$_{\Sigma+\mathbf{Frob}}$ 是一个在对象上为恒等映射的严格对称幺半超图函子，其在运算 $g \in \Sigma_1$ 上的定义由归纳给出，形式为 `singleton`（g）。

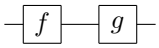
请注意，$\mathcal{S}$ 的计算复杂度较差。尽管张量积和复合的各个运算相对于所得图表的大小而言都是线性的，但上述归纳定义的函子具有 $O(n^2)$ 复杂度。这是因为我们不断将一个小图表“追加”到累加器中，这类似于通过反复追加来构造一个长度为n 的数组。我们将在第7节解决这个问题。在此之前，我们先构造函子$\mathcal{S}$的一个逆函子；该函子将图表映射到 Σ-项。

5.1 图表到 Σ-项

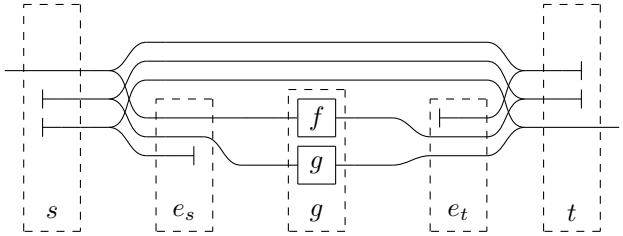
现在定义函子$\mathcal{D}$： Diagram$_{\Sigma+\mathbf{Frob}}$ → Free$_{\Sigma+\mathbf{Frob}}$，将图表映射到 Σ-项。为此，我们首先说明每个图表都可以因式分解为我们称之为“Frobenius 分解”的形式。这一复合依赖于超图范畴结构，该结构存在于Diagram$_{\Sigma+\mathbf{Frob}}$ 中，正如 [1,定理 312] 通过命题4.6所见。

Frobenius 分解的主要思想是将字符串图中的导线结构与生成元分离开来。为说明这一点，我们先来看一个例子。

例 5.2. 设我们有一个超图范畴中的态射 $f \fatsemi g$。

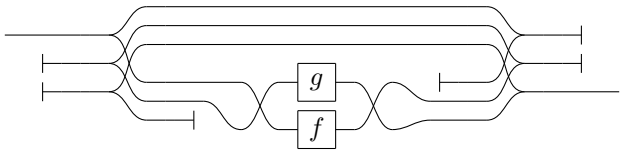


这个态射的一种 ***Frobenius*** 分解 是如下图所示的图表。



其思路是将图表顶部的所有导线画成一条“总线”，其宽度等于图表中内部线的数量。随后，可以通过映射 e_s 和 e_t 将导线连接到运算 g，它们分别将导线映射到运算的输入和输出。类似地，s 和 t 映射分别指定哪些“内部”导线出现在左边界和右边界上。

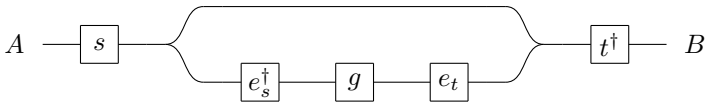
不过要注意，这种分解并不唯一。例如，我们可以交换中心处生成元的顺序，从而得到一个同构的图表。



注 5.3. *Frobenius* 分解的目的是将图表中的元素分解为 *Frobenius* 蜘蛛和运算。这样一来，定义超图函子会更加容易，因为它对 *Frobenius* 蜘蛛的作用是固定的，因此只需定义它对分解中 g 部分的作用即可。

事实上，任意超图范畴中的每个态射都有一个 Frobenius 分解，下面的命题将对此作出正式陈述。

命题 5.4（Frobenius 分解）。设 f： $A \to B$ 是超图范畴中的一个态射。那么，f 有一个分解，称为 f 的***Frobenius*** 分解，其形式如下。

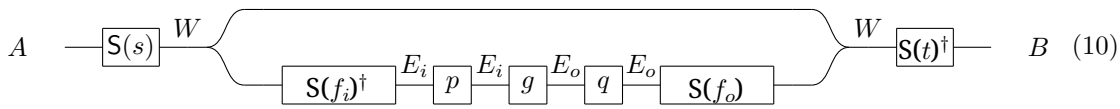


其中 σ、τ、e_s 和 e_t 是 *Frobenius* 蜘蛛，而 g 是生成元的 n 重张量积。

证明。 归纳法。完整证明见附录B.1 。 □

由于根据 [1, 定理 3.12], , Diagram$_{\Sigma+\mathbf{Frob}}$ 是一个超图范畴，每个态射都必须有一个 Frobenius 分解。为了定义函子$\mathcal{D}$，给出一个具体的Frobenius 分解会很有帮助，函子将以此为基础定义。这也将用于说明 Diagram$_{\Sigma+\mathbf{Frob}}$ 由 $\Sigma+\mathbf{Frob}$ 的生成元表示。

命题 5.5. 设 $d=\mathsf{L}(A)\ \mathsf{L}(\xrightarrow{s\ ⨾\epsilon_G} G\ )\mathsf{L}(\xleftarrow{t\ ⨾\epsilon_G})\mathsf{L}(B)$ 是签名 Σ 上的一个图表。那么，下图是 d 的 *Frobenius* 分解



(10)

其中：

- $W=\mathcal{W}(w_n)$、 $E_i=\mathcal{W}(G(w_i ⨾ w_n))$和 $E_o=\mathcal{W}(G(w_o ⨾ w_n))$
- $f_i=\mathcal{W}(G(w_i)\ G(w_n))$ 和 $f_o=\mathcal{W}(G(w_o)\ G(w_n))$
- $p=$ *(sorth*$_{G(x_i)}$ $_{G}$*(port*$_i$*)i,id,L(*E_i*))* 和 $q=$ *(sorth*$_{G(x_o)}$ $_{G}$*(port*$_o$*)i, id, L(*E_o*))*（其中 *sort* 在命题 *A.4* 中定义）
- $g=\bigotimes_i^X x_n(i)$ 是由 X 个单例图表的张量积构成的图表

证明。完整证明留待附录 B.1，这里先给出概要。显然，式 (10) 中所示的复合形式正确：S(s) 和 S(t)† 分别直接对应命题 5.4 中的 s 和 t，而复合 S($G(w_i)$)† ⨾ p 和 q ⨾S($G(w_o)$) 则分别对应 $e_s^\dagger$ 和 e_t。因此，剩下要证明的是，这种分解确实等于 (s,t,G)。这是因为存在一种余等化子的选取，使得 $p⨾g⨾q$ 同构于 G 中生成元的张量积 $g'=(\iota_0,\iota_1,G')$。然后用 g' 替换 $p⨾g⨾q$，便可再次选取余等化子，使整个复合等于 (s,t,G)。

□

备注 5.6. 注意，排序置换 sort$_{\langle x_i,\text{端口}_i\rangle}$ 和 sort$_{\langle x_o,\text{端口}_o\rangle}$ *(命题 A.4)* 是唯一的，因为它们的“排序键”是单态的；这是因为 G 是良构的。此外，无须实际使用排序算法计算排序置换，也可以通过算术运算在（顺序）线性时间和（并行）常数时间内计算它们。

Frobenius 分解使我们能够定义一个从图表到 Σ-项的函子。我们只需将分解的每个分量映射到一个代表性的 Σ-项。

命题 5.7. 存在一个严格对称对象恒等幺半图超图函子 D： Diagram$_{\Sigma+\mathbf{Frob}}\to$ Free$_{\Sigma+\mathbf{Frob}}$， 它将态射映射到其 *Frobenius* 分解。

证明。 在附录B.1 中，我们证明 $\mathcal{D}$ 是良定义的（命题B.3）、是一个函子（命题 B.4），并且是严格幺半的（命题 B.5）。最后，根据定义，它保持对称性和特殊 Frobenius 幺半群结构，因为 Frobenius 蜘蛛被映射到 Free$_{\Sigma+\mathbf{Frob}}$ 中对应的蜘蛛。 □

5.2 同构

现在我们可以证明 $\mathcal{S}$ 和$\mathcal{D}$ 构成一个同构。我们将使用 [26, B.1]，中的“基变换”引理，它给出了两个范畴同构的条件。我们先证明其中一个条件。

命题 5.8。 图表由 $\Sigma+\mathbf{Frob}$ 的运算生成。也就是说，每个图表都可以通过对id、 σ以及 $\Sigma+\mathbf{Frob}$ 的生成元进行张量积和复合来构造。

证明。 根据命题 5.5，每个图表都有 Frobenius 分解。由于分解中的每个元素要么是 Frobenius 蜘蛛，要么是运算的张量积，因此完整的图表可以通过生成元的张量积和复合来构造。 □

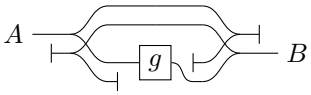
现在我们可以给出主要结果。

命题 5.9. *S* 和 *D* 构成一个同构。

证明。 来自基变换引理的 [26, B。1] 指出，若满足以下条件，则 $\mathcal{S}$和 $\mathcal{D}$ 构成一个同构。

- $\mathcal{S}$ 和 $\mathcal{D}$ 是严格对称幺半范畴中对象上恒等的超图函子（定义 5.1，命题B.5）
- $\mathbf{Free}_{\Sigma+\mathbf{Frob}}$ 由$\mathbf{Frob}$ 的运算生成（根据定义）
- 图表$_{\Sigma+\mathbf{Frob}}$ 由 $\mathbf{Frob}$ 中的运算的单例图表生成（命题5.8）
- $\mathcal{D}(\mathcal{S}(g)) \cong g$ 对于 $g \in \Sigma_1$。

因此，只需证明 $\mathcal{D} \circ \mathcal{S}$ 在生成元 $g \in \Sigma_1$ 上是逆。这很容易推出：设 $\texttt{singleton}(g) : A \to B$ 是任意生成元，则根据下图应用特殊 Frobenius 幺半群的单位公理和余单位公理即可得出结论。

$g \quad = \quad$ 

□

6 对称幺半范畴

现在我们考察对称幺半范畴的情形。具体而言，我们将得出结论：Diagram$_\Sigma$ 是 Diagram$_{\Sigma+\mathbf{Frob}}$ 的一个子范畴，且与 Free$_\Sigma$同构。

我们的方法基于 [4], 的工作。他们用超图来刻画 $\Sigma + \mathbf{Frob}$ 上的弦图。通过对超图施加一个额外的组合条件——“单配无环性”——他们重新得到 Σ上的弦图。也就是说，单配无环超图对应于不带额外超图结构的$\mathbf{Frob}$弦图。

S由于我们在定义4.1 中描述的数据结构本质上是一种编码.....的超图 r-graphs 在 [4], 中，我们现在只需将这一条件转换为适用于我们当前情形的形式。

定义 6.1 (单配性 [4])。 一个图表 $d = \mathsf{L}(A) \xrightarrow{s} G \xleftarrow{t} \mathsf{L}(B)$ 称为单配的，当 s和 t 是单射，并且以下条件对所有 ■-节点 i均成立。

$$\text{indegree}(i) = \begin{cases} 0 & \text{if } \exists j.s(j) = i \\ 1 & \text{otherwise} \end{cases} \qquad \text{outdegree}(i) = \begin{cases} 0 & \text{if } \exists j.t(j) = i \\ 1 & \text{otherwise} \end{cases}$$

其中，我们将 入度$(i) = |G(w_i)^{-1}[i]|$ 和 出度$(i) = |G(w_o)^{-1}[i]|$记作

定义 6.2 (无环性)。 当二部多重图的底层图是无环图时，该二部多重图就是无环的。

现在让我们验证，这些条件确实刻画了 Free$_\Sigma$ 中的态射。我们首先给出一个引理，该引理指出，单配无环的“Frobenius 蜘蛛”实际上就是置换。

命题 6.3。 如果 $d = (s, t, \mathsf{L}(Y))$ 是一个单配无环 *Frobenius* 蜘蛛，那么 s和 t 都是置换。

证明。 根据定义 6.1， s 和 t 都是单态射（即单射）。接着注意，$\mathsf{L}(Y)$ 是一个离散图，因此每个 ■-节点的入度和出度都必须为 0。要满足这一点，每个节点都必须同时属于 s 和 t的像，因此它们是满射。由此立即可得，s 和 t 是双射。 □

现在检查 Free$_\Sigma$ 中的这些态射经 ToDiagram 映射后是否成为单配无环图表。

命题 6.4。 如果 f 是Free$_\Sigma$的一个态射，那么ToDiagram(f) 是单配无环的 *ic.*

证明。 归纳法。

基例：生成元 根据命题 6.3 生成元 id 和 σ 是单配无环的。随后可以直接验证，单例图表（定义 4.9）是单配无环的，因此对于每个 $g \in \Sigma_1$ 都有 $\mathsf{ToDiagram}(g)$ 是单配无环的。

归纳步骤：张量积 令 $d_i = (s_i, t_i, G_i)$ 为$i \in \{0, 1\}$上的单配无环图。根据定义，$(s, t, G) := d_0 \otimes d_1 = (s_0 \otimes s_1, t_0 \otimes t_1, G_0 \otimes G_1)$。$G$ 是无环的，因为它是无环图的不交并；s 和 t 是单射，因为 s_i 和 t_i是单射。接下来只需检查入度和出度条件。为简洁起见，我们验证前者，省略后者。

> 假设 i 是 ■中的一个节点。那么，如果$\mathsf{入度}(i) = 1$，根据假设，i 不在s_0的像中，因此也不在 $s_0 \otimes s_1$ 的像中。此外，如果$\mathsf{入度}(i) = 0$，那么存在某个 j，使得 $s_0(j) = i$，于是 $(s_0 \otimes s_1)(j) = i$。
>
> 现在假设 i 是 G_1 中的一个 ■-节点，并记 $m := G_0(W)$。如果 $\mathsf{indegree}(i) = 1$，那么根据假设，i 不在 s_1 的像中，且 $m + i$ 不在 $s_0 \otimes s_1$ 的像中。此外，如果 $\mathsf{indegree}(i) = 0$，则存在某个 j，使得 $s_1(j) = i$，于是 $(s_0 \otimes s_1)(m + j) = i$。

归纳步骤：复合。设 $d_i = \mathrm{L}(A_i)\, \mathrm{L}(\xrightarrow{s_i \fatsemi \epsilon_{G_i}} G_i)\, \mathrm{L}(\xleftarrow{t_i \fatsemi \epsilon_{G_i}})\, \mathrm{L}(A_{i+1})$ 是$i \in \{0, 1\}$ 的图表。它们的复合由余等化子 $q = \mathsf{c}(t_0 \fatsemi \iota_0, s_1 \fatsemi \iota_1)$ 给出。由于$t_0 \fatsemi \iota_0$ 和 $s_1 \fatsemi \iota_1$ 是单射，若 $q(i) = q(j)$ 对于 $i \neq j$ 成立，那么存在唯一的 $a \in A_1$，使得 $(t_0 \fatsemi \iota_0)(a) = i$ 且 $(s_1 \fatsemi \iota_1)(a) = j$（反之亦然）。但由于 d_0 和 d_1 是单配无环的，因此 $\mathsf{outdegree}(i) = 0$，且 $\mathsf{indegree}(j) = 0$；所以 $\mathsf{indegree}(q(i)) = \mathsf{indegree}(i)$，且 $\mathsf{outdegree}(q(i)) = \mathsf{outdegree}(j)$，商图也同样是单配的。

> 最后，注意到无环性也基于同样的原因得以保持：如果 G_0 和 G_1 是无环的，那么从 $q(i)$ 只能到达 G_1 中的 ■-节点，因此不可能出现环。

□

现在我们来验证反向命题：所有单配无环图都对应于 Free_Σ 的态射。这个事实最早在 [4]，中得到证明，因此并不意外。此外，我们的论证与 [26，命题 B.3]，基本相同，所以这里只对证明作简要说明。

命题 6.5。 如果 d 是一个单配无环图表，那么 $D(d)$ 同构于一个 Free_Σ 的态射。

证明。令 $d = (s、t、G)$ 为一个图表。利用无环性，d可以分解为形式 $s \fatsemi \mathcal{S}\left(\square\right) \fatsemi d'$。然后，由于 s 是单射，按照生成元 d的数量进行归纳即可得出结果。

□

由这两个引理可知，存在一个单配无环图范畴，其态射与同一签名上的自由对称幺范畴的态射同构。

Pro命题 6.6。 存在一个子范畴 图表$_\Sigma$，它属于 图表$_{\Sigma+\mathbf{Frob}}$，其态射与 *re*mon配无环图表，且 图表自由。

证明。正如命题 6.4 和 6.5所保证的，单配无环性在张量积和复合下保持闭合。此外，同构 图表$_\Sigma \cong$自由$_\Sigma$ 由函子 $\mathcal{S}$ 和 $\mathcal{D}$ 限制在单配无环图表上而得到。

□

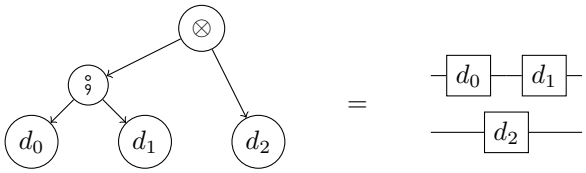
注记 6.7。 请注意，函子 D 并不直接给出一个 Σ项，而“仅仅”给出一个与之等价的（$\Sigma+\mathbf{Frob}$）项。如果希望从图表中恢复一个真正的 Σ项，可以直接使用基于例如拓扑排序的分解，具体如 *[26, 命题 B.13]*中所述。

7 更快的函子

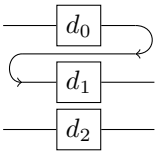
在上一节中，我们给出了张量积与图表复合的算法，其顺序复杂度为 $O(n)$，PRAM 复杂度为 $O(\log n)$。这使我们能够定义一个函子$\mathcal{S}$：$\mathsf{Free}_{\Sigma+\mathbf{Frob}} \to \mathsf{Diagram}_{\Sigma+\mathbf{Frob}}$。然而，朴素地应用这个函子的复杂度为 $O(n^2)$。

现在我们给出一个更快的函子$\mathcal{F}$： $\mathrm{Free}_{\Sigma+\mathbf{Frob}} \to \mathrm{Diagram}_{\Sigma+\mathbf{Frob}}$ ，它可以直接计算得到的图表。重要的是，这里介绍的算法在（顺序）时间复杂度上为线性，在 PRAM CREW 时间复杂度上为对数级，均相对于所得图表的大小而言。给定一个 Σ-项t，基本思路是先对 t 的所有生成元取张量积，再一步将它们全部连接起来。我们用一个例子来说明这个过程。

例 7.1。 考虑下面的 Σ-项（表示为一棵树）及其对应的字符串图。



本节的思路是，先将所有图表$d_0 \otimes d_1 \otimes d_2$张量积起来，再一步完成图表的“布线”，从而构造右侧的字符串图。更直观地说，我们可以将其想象成下面的字符串图，其中 d_0 的目标被“连接”到 d_1的源。



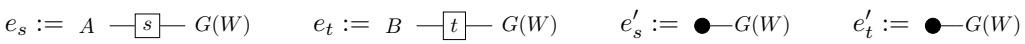
7.1 更快的函子

为了“一步完成图表的布线”，我们必须指定哪些 ■-节点要合并为商，以及哪些节点将位于边界上。这一指定由一个四元组给出，我们称之为“布线映射”。

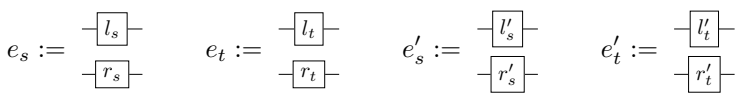
定义 7.2 （布线映射）。 设 e 是一棵二叉树，其 n 个叶子对应图表 $d_i =$ （s_i、t_i、G_i），且其节点标记为 $;$ 或 $\otimes$。 e 的四个布线映射 类型如下

$$e_s : A_S \to G(W) \qquad e_t : A_T \to G(W) \qquad e'_s : A_I \to G(W) \qquad e'_t : A_I \to G(W) \tag{11}$$

其中 $G = \bigotimes G_i(W)$。 A_S、 A_T 和 A_I按如下方式递归计算。当 e 是一棵由单个叶子 $e = \mathsf{L}(A)$ $\mathsf{L}(\xrightarrow{s\ ;\epsilon_G} G)\mathsf{L}(\xleftarrow{t\ ;\epsilon_G})\mathsf{L}(B)$ 构成的二叉树时，布线映射是下文给出的 Wires 态射。

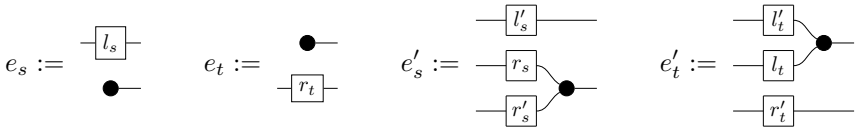


树 $e =$ $\otimes$ 的布线映射由 l r 给出



(12)

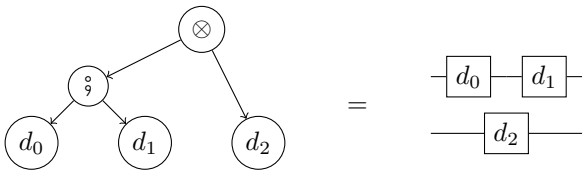
$e =$ $;$ 的布线映射则定义为 l r



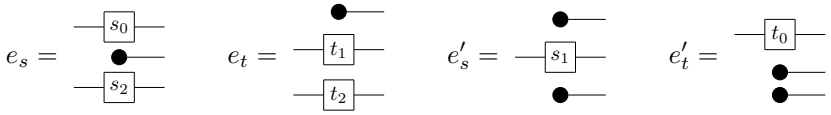
(13)

映射 e'_s 和 e'_t 共同用于确定需要合并取商的内部 ■-节点；最终图表通过对这些映射进行余等化得到。同时，映射 e_s 和 e_t 用于确定与图表边界相对应的■-节点。为了说明这一点，我们现在给出.....

例 7.3。 回顾下面的示例 Σ-项及其对应的字符串图。



记 s_i 和 t_i 为 d_i的源映射和目标映射，我们有



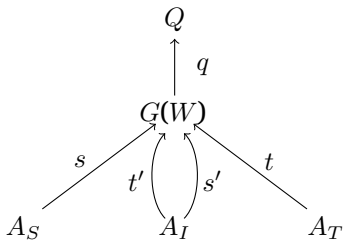
注意，对“内部布线映射” e'_s 和 e'_t 进行余等化，将使 d_0 的目标与 d_1的源相等。

利用布线映射，现在可以定义“快速”函子$\mathcal{F}$。

定义 7.4. F :Free$_{\Sigma+\mathbf{Frob}}$ $\to$ Diagram$^{\Sigma+\mathbf{Frob}}$ 是一个对象上恒等的函子，其在箭头（Σ-项）上的定义如下。设 e 为一棵二叉树，它由将一个 Σ-项的叶节点替换为相应的单例图表得到，并令

$$s : A_S \to G(W) \qquad t : A_T \to G(W) \qquad s' : A_I \to G(W) \qquad t' : A_I \to G(W)$$

为 e的布线映射。定义 $q := c(t', s')$ 为内部布线映射的余等化子。



(14)

如命题 *4.16*所示， q 提升为一个余等化子 α ，它位于 BipartiteMultigraph 中，其 W 分量为 q，所有其他分量均为 id。 $\mathcal{F}(e)$ 于是对应于图表 $(s \fatsemi q,\ t \fatsemi q,\ \text{目标}(\alpha))$。

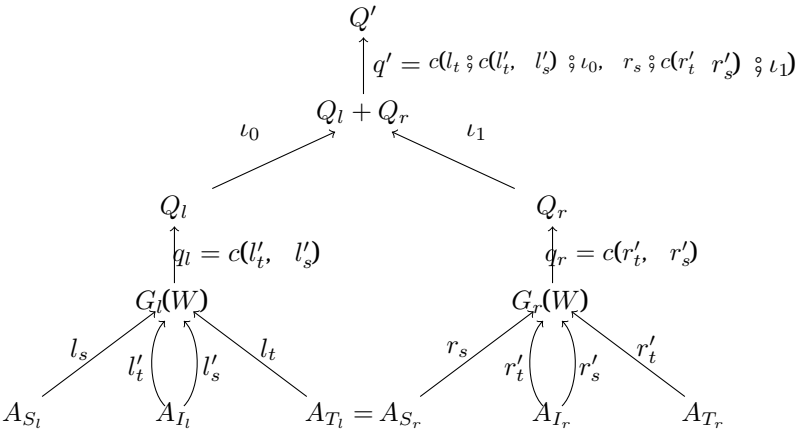
注 7.5。 请注意，F 不仅定义了一个从 Free$^{\Sigma+\mathbf{Frob}}$ 出发的函子，还可以用来任意地“连接”已有的图表，因为它可以作用于叶子为图表的任意树，而不仅仅是单例图表。

在命题7.7中，我们将验证 $\mathcal{F}$ 和 $\mathcal{S}$ 作用于同一表达式树时会得到同构图表。这也将用于验证定义 7.4的良定义性。本证明中的一个关键引理是 $\mathcal{F}$的函子性，下面我们来证明这一点。

命题 7.6 （$\mathcal{F}$ 的函子性）。 $\mathcal{F}(\mathsf{id}) \cong \mathsf{id}$ 且 $\mathcal{F}\left(\begin{smallmatrix} & \fatsemi & \\ l & & r \end{smallmatrix}\right) \cong \mathcal{F}(l) \fatsemi \mathcal{F}(r)$

证明。 我们有 $\mathcal{F}(\mathsf{id}) = \mathsf{id}$ 根据定义，因此还需证明 $\mathcal{F}\left(\begin{smallmatrix} & \fatsemi & \\ l & & r \end{smallmatrix}\right) \cong \mathcal{F}(l) \fatsemi \mathcal{F}(r)$。这意味着需要检查 Q 在（14）中与 Q' 在（15）中同构，并且该同构扩展到

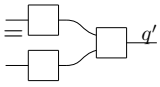
余锥的同构。



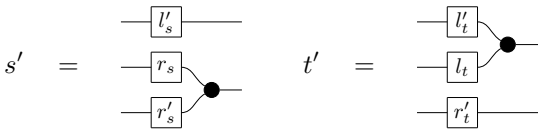
(15)

证明分为四个部分。(1) 构造“正向”映射 $u : Q \to Q'$，(2) 构造“逆向”映射 $u' : Q' \to Q$，(3) 证明它们构成一个同构，以及 (4) 验证这扩展为一个余锥同构。

(1) 前向首先，通过证明存在一个映射 $x : G(W) \to Q'$，它余等化 s' 和 t'，来构造唯一态射 $u: Q \to Q'$。

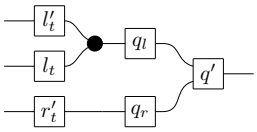
q_l根据定义，$G(W) = G_l(W) \otimes G_r(W)$，因此我们可以定义 $x :$ 

。现在我们q_r将验证 $s' \fatsemi x = t' \fatsemi x$。首先，回顾定义7.2，我们有

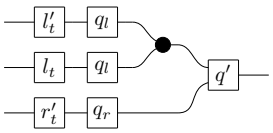


因此我们可以计算

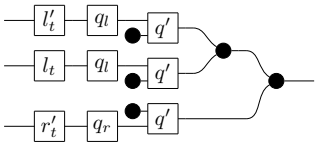
$t' \fatsemi x$ $=$



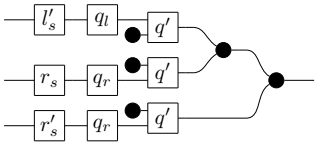
$=$



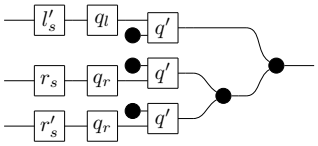
$=$



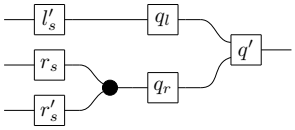
$=$



$=$



$=$

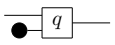


$=$ $s' \fatsemi x$

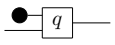
因此，由于 x 余等化了 t' 和 s'，根据余等化子的泛性质，存在唯一态射 $u\colon\ Q \to Q'$，使得 $q \fatsemi u = x$。

(2) 逆 现在我们来构造逆态射 $u'\ :\ \ Q' \to Q$。首先定义映射 y_l :$G_l(W) \to Q$ 和 $y_r\colon\ G_r(W) \to Q$ 如下。

$y_l \quad :=$

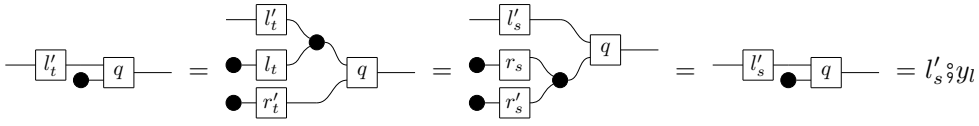


$y_r \quad :=$



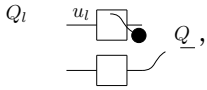
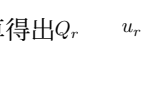
现在计算可得，y_l 余等化了 l'_t 和 l'_s。

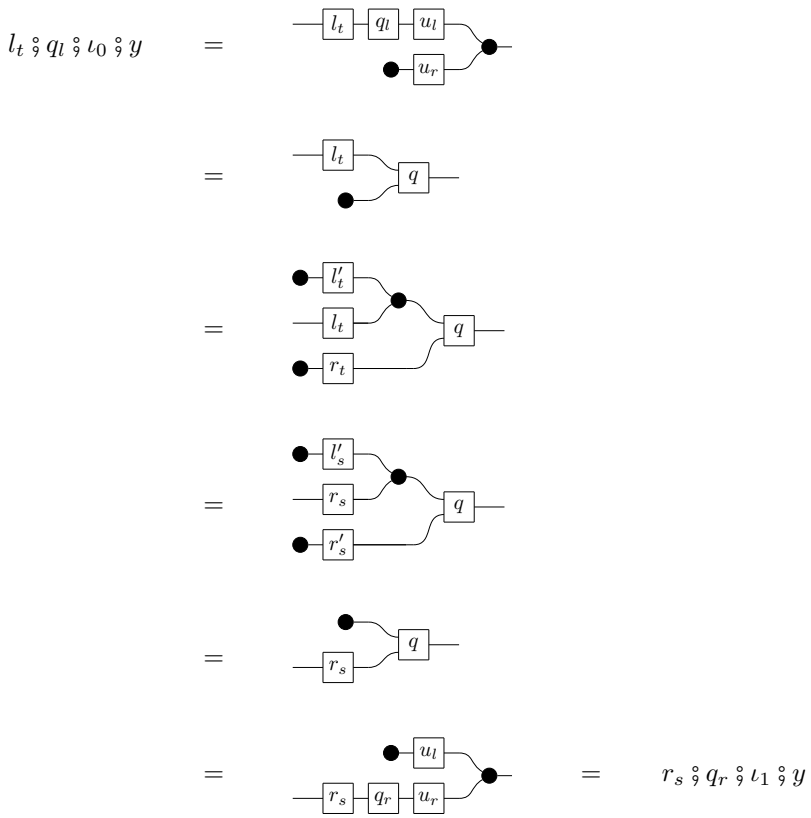
$l'_t \fatsemi y_l =$



$= l'_s \fatsemi y_l$

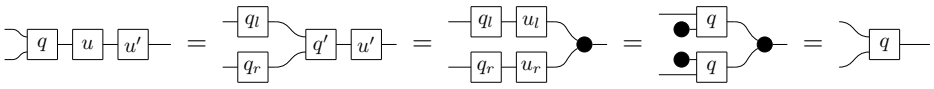
类似计算可得 $r'_t \fatsemi y_r = r'_s \fatsemi y_r$，因此根据泛性质，必定存在唯一态射 $u_l\colon\ Q_l \to Q$ 和 $u_r\colon\ Q_r \to Q$，使得 $y_l = q_l \fatsemi u_l$ 和 $y_r = q_r \fatsemi u_r$。

利用 u_l 和 u_r，我们可以定义 y:=  ，并计算得出 

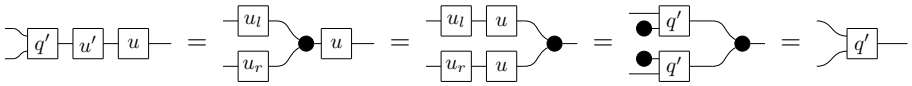


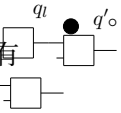
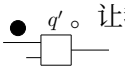
因此，由于 y 余等化了 $l_t \fatsemi q_l \fatsemi \iota_0$ 和 $r_s \fatsemi q_r \fatsemi \iota_1$，必定存在唯一的 $u' \colon Q' \to Q$，使得 $y = q' \fatsemi u'$。

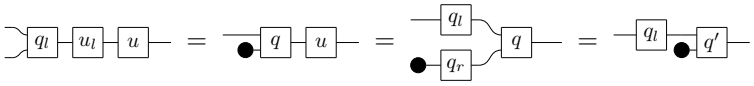
(3) 同构 我们已经构造了态射 $u \colon Q \to Q'$ 和 $u' \colon Q' \to Q$，因此接下来只需验证它们确实构成一个同构。我们先验证 $u \fatsemi u' = \mathsf{id}$。根据定义，$t' \fatsemi q = s' \fatsemi q$。因此，根据泛性质，存在唯一态射 v，使得$q \fatsemi v = q$。显然，$v = \mathsf{id}$ 满足这一条件，所以如果 $q \fatsemi u \fatsemi u' = q$，那么根据唯一性，$u \fatsemi u' = \mathsf{id}$。证明如下。



同样地，如果我们能证明 $q' \fatsemi u' \fatsemi u = q'$，那么根据唯一性可得 $u' \fatsemi u = \mathsf{id}$。

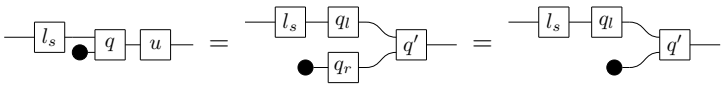


注意，在第三步中，我们使用了 $u_l \fatsemi u =$  和 $u_r \fatsemi u =$  。根据泛性质即可再次得到这些结论；我们如下证明前者。回忆一下，根据定义，q_l 是 l'_t 和 l'_s 的余等化子，因此显然  也必须将它们余等化。根据泛性质，存在唯一态射 v 使得 $q_l \fatsemi v =$  。显然，$v =$  满足这个等式，因此若 $q_l \fatsemi u_l \fatsemi u =$  ，则根据唯一性有 $u_l \fatsemi u =$  。让我们计算：



因此，所求的 $u_l \fatsemi u$ 与 q' 成立。类似的论证可得 $u_r \fatsemi u$ 与 q，从而完成了 u 和 u' 构成同构的证明。

(4) 余锥同构 现在我们证明 u 可扩展为余锥的同构。下面的计算通过泛映射u将（14）与（15）的余锥源腿对应起来。



类似的计算可得反方向的结果；对目标映射应用对称论证，即可完成证明。

□

现在我们证明$\mathcal{F}$ 与$\mathcal{S}$之间的同构。

P命题 7.7。 设 e 是一个 Σ-项。则存在图表同构 $F(e) \cong S(\quad e)$.

证明。 我们采用归纳法。当 $e \in \{\mathrm{id}\ \sigma\} \cup \Sigma_1$ 时，这是基例，此时$\mathcal{F}(e) = \mathcal{S}(e)$。对于归纳步骤，假设存在同构 $u_l : \mathcal{F}(l) \cong \mathcal{S}(l)$和 $u_r : \mathcal{F}(r) \cong \mathcal{S}(r)$。现在我们必须证明，对于情形 $e =$ 和 $e =$ ，有$\mathcal{F}(e) \cong \mathcal{S}(e)$。对于前一种情形，注意到$l \quad r \qquad l \quad r$:

$$\mathcal{F}\left(\begin{array}{ccc} & \otimes & \\ l & & r \end{array}\right) = \mathcal{F}(l) \otimes \mathcal{F}(r) \cong \mathcal{S}(l) \otimes \mathcal{S}(r) = \mathcal{S}(l \otimes r)$$

其中的同构 $u_l \otimes u_r$ 由归纳假设给出。在后一种情况下，我们推导如下。

$$\mathcal{F}\left(\begin{array}{ccc} & \fatsemi & \\ l & & r \end{array}\right) \cong \mathcal{F}(l) \fatsemi \mathcal{F}(r) \cong \mathcal{S}(l) \fatsemi \mathcal{S}(r) = \mathcal{S}(l \fatsemi r)$$

该推导的前两个步骤使用了 $\mathcal{F}$ 的函子性（命题7.6）以及图表复合的良定义性 [1, 命题 3。11]分别。

□

Cor推论 7.8。 设 e 是一个具有 N 个叶子的 Σ-项（二叉树）。计算 $F(e)$ 具有 $O(N)$ （顺序 n-执行）以及 $O(\log N)$ （*PRAM CRCW*）时间复杂度。

证明。 由于 Σ 是固定的，我们认为图表中的导线数和边数与 N 成正比（参见 [26, 命题 6.1, 有界稀疏性]）。叶子的 N重张量积根据命题 4.14具有给定的复杂度，接下来只需计算连通分支；这同样是 $O(N)$顺序的，并且是 $O(\log N)$并行的。

□

8 快速布线

在第7 节中，我们展示了如何使用余等化子和布线映射（定义7.2）一次性完成图表的“布线”。现在，我们给出一种用于计算布线映射的并行算法。该算法分为两部分。第一部分（第8.1节）将计算布线映射的问题转化为寻找“祖先映射”。对于给定 Σ项中的所有节点，祖先映射都会寻找一个标记为 ;的最近祖先（如果存在），使给定节点位于其特定的（左侧或右侧）子树中。算法的第二部分（第8.2节）是一种数据并行方法，用于高效计算这些祖先映射。

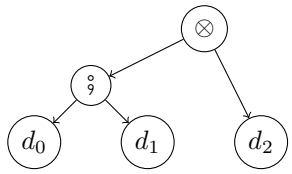
8.1 将问题归约为树谓词

现在，我们给出布线映射的一种替代性“适合并行”的定义，该定义在命题8.5中被证明是等价的。通过这一替代定义高效地并行计算布线映射，将取决于计算下文定义的“祖先映射”。

定义 8.1（左侧和右侧 $\wp$-祖先）。 在二叉树 e 中，节点 i 若位于任意标记为 $\wp$ 的节点 j 的左侧 $\wp$子树中，则称其具有右侧 $\wp$-祖先。相应地， i 若位于任意标记为 $\wp$ 的节点 j 的右侧 子树中，则称其具有左侧 $\wp$-祖先。

计算节点的左/右祖先可以确定如何将图表中的导线连接起来。我们通过以下示例对此进行说明。

示例 8.2。 下图展示了一个呈现为树的 Σ-表达式，并在下方列出了每个节点最近的左（分别为右）$\wp$-祖先，其中 · 表示节点没有左（分别为右） $\wp$-祖先。



$node$	0	1	2	3	4
$Left$	·	·	1	·	·
正确	1	·	·	·	·

图表（叶节点） d_0 和 d_1 由节点 1“绑定”，表示 d_0 的输出将“连接”到 d_1的输入。

定义 8.3（祖先映射）。 设 e 是一棵具有 n 个叶子和 $n-1$ 个（非叶子）节点的二叉树。 e 的左祖先映射和右祖先映射 如下。

$$a_L : \ n \to na_L : = \begin{cases} 0 \text{ 如果 } i \text{ 没有左侧 } \wp\text{-祖先，则取最大的 } j \text{，使得} \\ j \text{ 是 } i \text{ 的左侧 } \wp\text{- 祖先 } a_R : \ n \to na_R : = \begin{cases} j \text{ 取最小的} \\ j \quad j \quad \wp \quad i \text{，使其} \end{cases} \end{cases}$$

n 是 i 的右侧 $\wp$-祖先，如果 没有左侧 $\wp$-祖先。

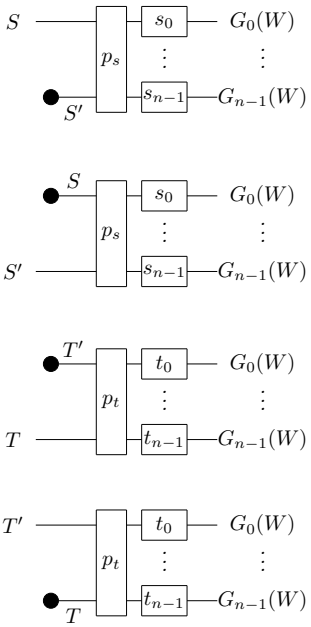
注释 8.4。 左祖先映射 a_L 返回给定节点位于右子树中的“最近”祖先的索引——这意味着在中序遍历中，该祖先位于该节点的“左侧”。

我们将在第 8.2 节展示如何高效计算祖先映射。

命题 8.5（布线映射的另一种定义）。 设 e 是一棵有 n 个叶子的树，图表

d_i： $A_i \to B_i$ 对于 $\overline{i} \in n$。 e 的布线映射也可以等价地定义如下。

$$
\begin{aligned}
e_s &= \iota_0^{S,S'} \fatsemi p_s \fatsemi \bigotimes_{i\in\overline{n}} s_i = \\
e'_s &= \iota_1^{S,S'} \fatsemi p_s \fatsemi \bigotimes_{i\in\overline{n}} s_i = \\
e_t &= \iota_1^{T',T} \fatsemi p_t \fatsemi \bigotimes_{i\in\overline{n}} t_i = \\
e'_t &= \iota_0^{T',T} \fatsemi p_t \fatsemi \bigotimes_{i\in\overline{n}} t_i =
\end{aligned}
$$



(16)

其中 p_s 和 p_t 是由稳定排序置换定义的自然变换 $\mathsf{sort}_{\langle a_L,\ \mathsf{id}\rangle}$ 和 $\mathsf{sort}_{\langle a_R,\ \mathsf{id}\rangle}$，分别[2]，而对象 S、S'、 T'和 T 定义如下。

$$
S = \bigotimes_{\{i\in\overline{n} \mid a_L\ (i)\ =0\}} A_i \qquad S' = \bigotimes_{\{i\in\overline{n} \mid a_L(i)>0\}} A_i
$$

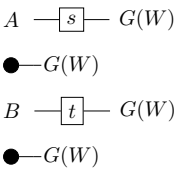
$$
T = \bigotimes_{\{i\in\overline{n} \mid a_R(i)=n-1\}} B_i \qquad T' = \bigotimes_{\{i\in\overline{n} \mid a_R(i)<n-1\}} B_i
$$

注 8.6。 对象 S 和 S' 分别指未绑定和已绑定的源映射。如果某个图表的源映射在构造图表时不会与其他导线取商，那么该源映射就是“未绑定”的。注意，张量积 S 中的对象恰好是没有左侧 ⨟祖先的那些源对象。因此，这些源映射将构成所得图表中的左侧悬垂导线。与此同时， S' 表示确实有左侧 ⨟祖先的源映射，因此属于复合——它们将指出图中需要在结果中取商的 ■节点。类似地，T 和 T' 分别是未绑定和已绑定的目标映射。

证明。 归纳法。

基例：单叶 令 e 为一棵仅由一片叶子构成的树，图表 $d = (s,\ t,\ G)$ 的类型为 $A \to B$。那么 $a_L = \mathsf{id}$，因此 $p_s = \mathsf{id}$；且 $a_R = \mathsf{id}$，因此 $p_t = \mathsf{id}$。此外， $S = A$、 $S' = I$、$T = B$ 和 $T' = I$，可按如下方式计算。

$$
\begin{aligned}
e_s &= \iota_0^{A,I} \fatsemi \mathsf{id} \fatsemi s = \\
e'_s &= \iota_1^{A,I} \fatsemi \mathsf{id} \fatsemi s = \\
e_t &= \iota_1^{I,B} \fatsemi \mathsf{id} \fatsemi t = \\
e'_t &= \iota_0^{I,B} \fatsemi \mathsf{id} \fatsemi t =
\end{aligned}
$$

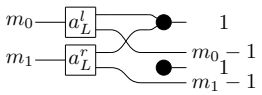
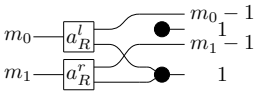


⊗**归纳步骤：张量积** 令 $e =$ 为一棵有 $n = m_0 + m_1$ 片叶子的树。根据归纳假设-l r

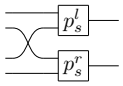
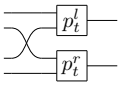
esis, (16) 等同于定义 7.2 对于 l 和 r。然后，用 a^l 和 a^r 表示祖先

[2]我们在定义A.5中定义“稳定”排序，并省略这些自然变换的下标，以减少记号负担。

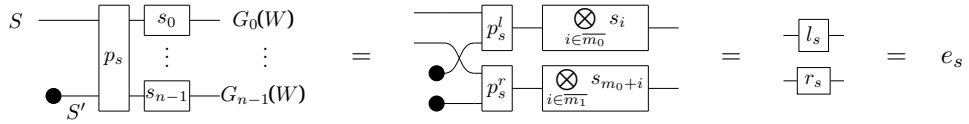
l 和 r 的映射，我们有

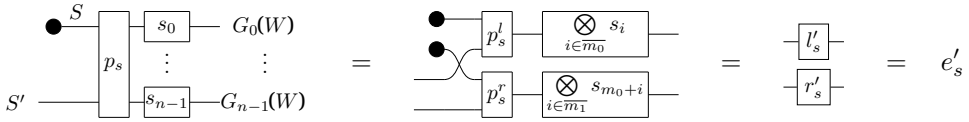
$a_L =$  and $a_R =$ 

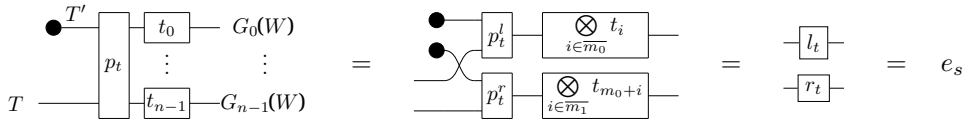
等等。

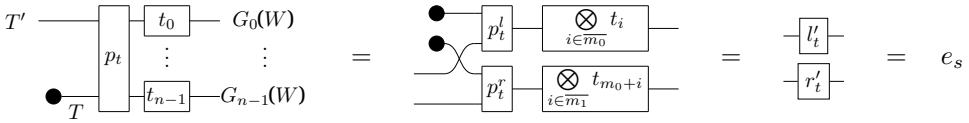
$p_s =$  $p_t =$ 

由此我们可以推导出



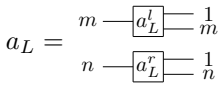
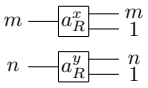






如所需。

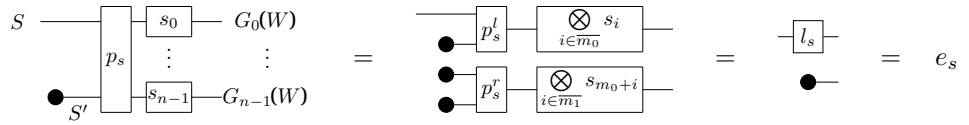
归纳步骤：复合 设 $e =$ 根据归纳假设，(16) 等于定义 x y 7.2对 l 和 r。接下来，使用 a^l 和 a^r 表示 l 和 r的祖先映射，我们有

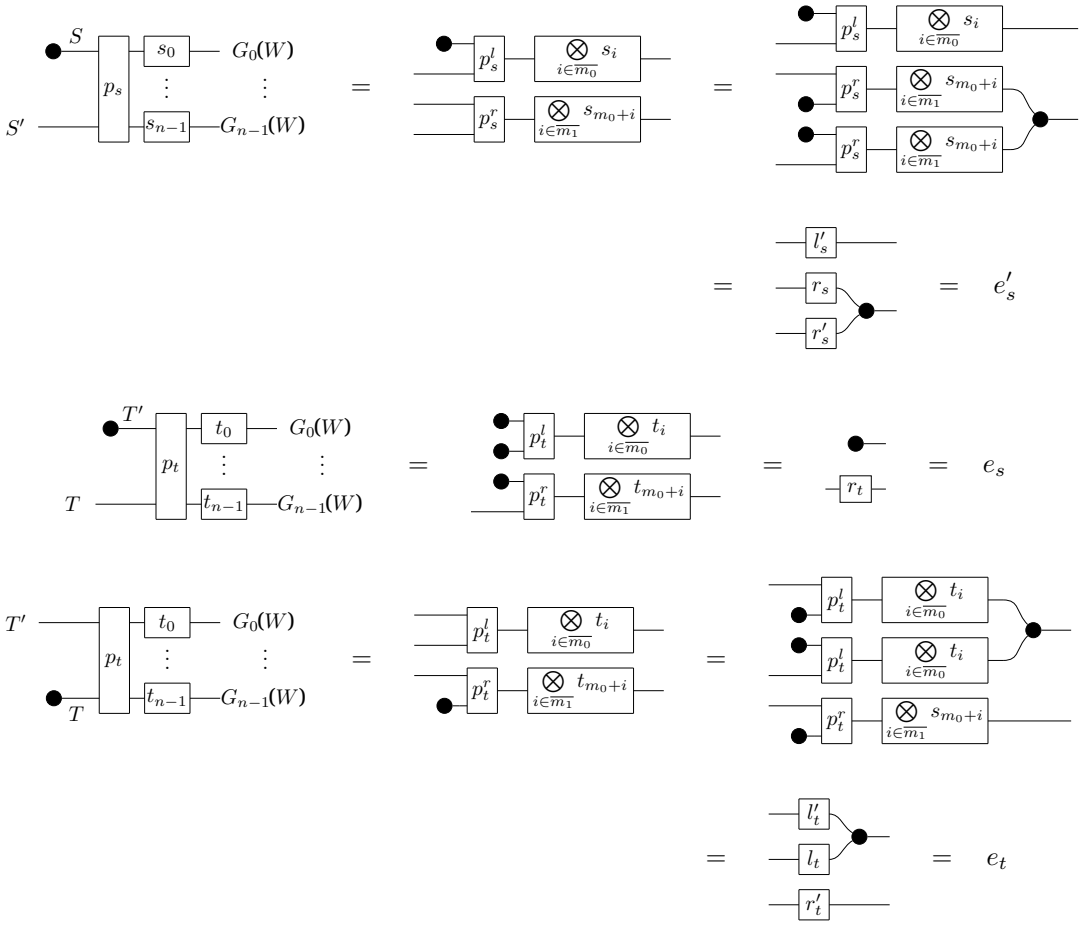
$a_L =$  $a_R =$ 

因此

$p_s =$  $p_t =$ 

由此我们可以推导出





从而完成证明。 □

命题 8.7。 设 e 是一棵包含 n 个节点的二叉树，并令 a_L 和 a_R 表示 e 的祖先映射。计算 e 的布线映射需要 $O(n)$ 顺序时间，以及 $O(\log n)$ 并行时间。

证明。 假设已给出祖先映射 a_L 和 a_R。注意， p_s 和 p_t 可通过对 a_L 和 a_R 的数组数据进行稳定的整数排序得到。其顺序时间复杂度为 $O(n)$，并行随机存取机时间复杂度为 $O(\log n)$：顺序情形可使用计数排序，并行情形可使用基数排序。根据有限函数复合的线性（顺序）时间复杂度和 $O(1)$（并行）时间复杂度，即可得到结论。 □

8.2 计算左侧和右侧匹配祖先

在第8.1节中，我们将计算布线映射（定义 7.2）的问题归约为树上的一个简单函数——计算“祖先映射”（定义 8.3）。尽管在顺序情形下有一种直接的自顶向下算法，但它不容易并行化。

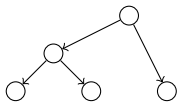
在本节中，我们给出一种数据并行算法，用于计算满足某个谓词、且其所在节点位于左子树或右子树中的最近祖先。这是对计算祖先映射问题的推广。我们的算法具有 $O(\log n)$ PRAM CREW 时间复杂度，我们将在推论 8.16 中证明这一点。

首先需要以一种“适合并行”的方式表示树。存在许多合适的、基于数组的树表示方法（例如，[10, 第 128] 页），它们可以在常数时间内计算给定节点的父节点和子节点索引。因此，我们不固定某一种特定的基于数组的树表示方法，而是假设可以在线性（顺序）时间和常数（并行）时间内计算以下有限函数。

定义 8.8。 设 e 为一棵大小为 $n = 2m - 1$ 的二叉树（即， e 有 m 个叶子和 $m - 1$ 个节点。）记 `parent`： $n \to n+1$ 为将节点映射到其父节点的有限函数；若不存在父节点，则映射到 n。

我们用 isRight： $n \to 2$ 表示一个谓词函数：如果某个节点是一个节点的右子节点，则返回1，否则返回0。类似地，用 isLeft： $n \to 2$ 表示一个谓词：如果某个节点是一个节点的左子节点，则返回1。

例 8.9. parent、 isRight 和 isLeft 的作用如下例所示，其中树的节点标记为 0.....4。



node	0	1	2	3	4
parent	1	3	1	5	3
isRight	0	0	1	0	1
isLeft	1	1	0	0	0

回顾一下，祖先映射（定义 8.3）旨在计算树中每个叶子的、标记为 ；的最近祖先。这里给出的算法解决了一个稍微一般化的问题：计算满足谓词 P 的最近祖先，其中谓词判断节点是否位于左子树或右子树中。这些函数的归纳定义如下。

定义 8.10（左/右 P祖先函数）。 设 e 是大小为 n的二叉树，且P： $n \to 2$ 是定义在节点上的谓词。左 P祖先 函数记为 $\texttt{ancestor}_L$，定义如下。

$$\texttt{ancestor}_L : n \to n+1$$

$$\texttt{ancestor}_L(i) = \begin{cases} n & \texttt{root}(i) \\ \texttt{parent}(i) & P(\texttt{parent}(i)) \vee \texttt{isRight}(i) \\ \texttt{ancestor}_L(\texttt{parent}(i)) + \texttt{isRight}(i) & \textit{otherwise} \end{cases}$$

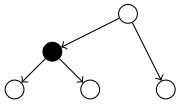
类似地，右 P祖先函数可表示为

$$\texttt{ancestor}_R : n \to n+1$$

$$\texttt{ancestor}_R(i) = \begin{cases} n & \texttt{root}(i) \\ \texttt{parent}(i) & P(\texttt{parent}(i)) \vee \texttt{isLeft}(i) \\ \texttt{ancestor}_R(\texttt{parent}(i)) + \texttt{isLeft}(i) & \textit{otherwise} \end{cases}$$

以下示例说明了 $\texttt{ancestor}_L$ 映射的作用。

例 8.11。 下图展示了一棵树，其中每个节点 i 在 P（i） $= 1$ 时绘制为黑色，否则绘制为白色；每个节点的 $\texttt{ancestor}_L$（i）值也标示在图中。

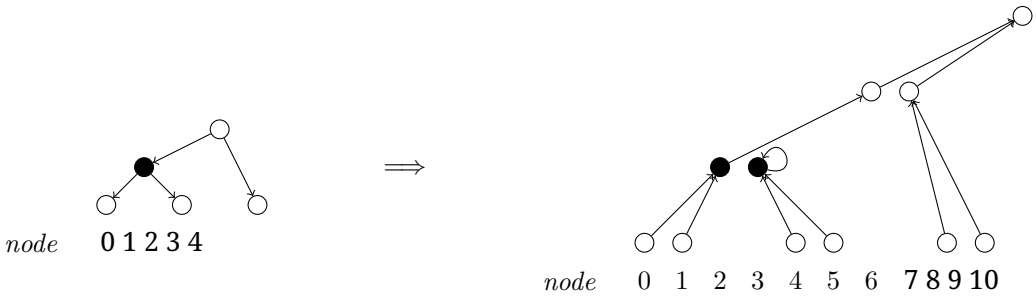


node	0	1	2	3	4
P	0	1	0	0	0
$\texttt{ancestor}_L$	5	5	1	5	5

我们采用“自底向上”的方法来高效计算 $\texttt{ancestor}_L$ 和 $\texttt{ancestor}_R$。首先，将大小为 n的树转换为包含 $2n+1$ 个节点的函数图，然后反复应用其邻接函数，直到结果收敛。下面通过一个示例来说明这一过程。

例 8.12。 下图展示了一棵包含 $n = 5$ 个节点的树，以及它的“左祖先图”，其中

$2n+1=11$ 节点。



对于树中的每个节点 i，图中都有两个节点，分别为 $2i$ 和 $2i+1$。图的边表示父节点关系，并分为两种不同情况：当子节点是其父节点的左 子节点或右 子节点时。节点 i 的左子节点与图节点 $2i$ 相邻，而右子节点与 $2i+1$ 相邻。当 $P(i)$ 时，其对应的图节点 $2i+1$ 指向自身，而不是指向其父节点。因此，在遍历邻接关系时，最终所有右侧后代都会被其最近的左侧祖先“捕获”。

下面将更正式地定义此图。

定义 8.13 (祖先图)。 设 e 为一棵具有 m 个叶节点和 $n=2m-1$ 个节点的树，并且 P：$n\to 2$ 是定义在节点上的谓词。 e 的左祖先图 具有 $2n+1$ 个顶点，其边由以下邻接函数给出。

$$r_L : 2n+1 \to 2n+1$$

$$r_L(j)=\begin{cases} j & j=2n \\ j & \texttt{odd}(j)\wedge P\left(\left\lfloor \frac{j}{2}\right\rfloor\right) \\ 2\cdot \texttt{parent}\left(\left\lfloor \frac{j}{2}\right\rfloor\right)+\texttt{isRight}\left(\left\lfloor \frac{j}{2}\right\rfloor\right) & \textit{otherwise}\end{cases}$$

对称地，右祖先图 的邻接关系为

$$r_R : 2n+1 \to 2n+1$$

$$r_R(j)=\begin{cases} j & j=2n \\ j & \texttt{even}(j)\wedge P\left(\left\lfloor \frac{j}{2}\right\rfloor\right) \\ 2\cdot \texttt{parent}\left(\left\lfloor \frac{j}{2}\right\rfloor\right)+\texttt{isRight}\left(\left\lfloor \frac{j}{2}\right\rfloor\right) & \textit{otherwise}\end{cases}$$

现在我们可以将祖先图与函数 $\texttt{ancestor}_L$ 和 $\texttt{ancestor}_R$联系起来。

命题 8.14。 对于树 e中深度为 $\texttt{depth}(i)$ 的所有节点 i

$$\texttt{ancestor}_L(i)=\left\lfloor \frac{r_L^k(2i)}{2}\right\rfloor \qquad \texttt{ancestor}_R(i)=\left\lfloor \frac{r_R^k(2i)}{2}\right\rfloor$$

对于 $k>\texttt{depth}(i)$，其中 r 是祖先图的邻接函数。

证明。 按深度进行自顶向下归纳。记 r 为 r_L，只证明前一种情况，省略关于 $\texttt{ancestor}_R$ 函数的对称证明。

基例 假设 $\texttt{root}(i)$，则 $\texttt{depth}(i)=0$。我们有 $r(2i)=2\cdot\texttt{parent}(i)+\texttt{isRight}(i)=2\cdot\texttt{parent}(i)=2n$，因此

$$\left\lfloor \frac{r(2i)}{2}\right\rfloor = n = \texttt{ancestor}_L(i)$$

此外， $2n$ 是 r的不动点，因此对于任何满足 $k\geq 1$的 r^k，这一结论都成立。

归纳情形 1 在第二种情形中，我们有 $P(\texttt{parent}(i))\wedge\texttt{isRight}(i)$。假设 $\texttt{depth}(i)=m$，其中 $m\geq 1$，因为 i 不是根节点。我们有

$$r(2i)=2\cdot\texttt{parent}(i)+\texttt{isRight}(i)=2\cdot\texttt{parent}(i)+1$$

现在令 $k \geq 1$，使得 $m + k > \texttt{depth}(i)$。然后

$$\left\lfloor \frac{r^{m+k}(2i)}{2} \right\rfloor = \left\lfloor \frac{r^{m+k-1}(2 \cdot \texttt{parent}(i) + 1)}{2} \right\rfloor = \left\lfloor \frac{2 \cdot \texttt{parent}(i) + 1}{2} \right\rfloor = \texttt{parent}(i) = \texttt{ancestor}_L(i)$$

按要求。

归纳情形 2 在最后一种情形中，设 i 为一个节点，且 $k > \texttt{depth}(i)$，使得 $k - 1 > \texttt{depth}(\mathit{parent}(i))$，归纳假设如下。

$$\left\lfloor \frac{r^{k-1}(2 \cdot \texttt{parent}(i))}{2} \right\rfloor = \texttt{ancestor}_L(\texttt{parent}(i))$$

现在进一步分为两种情况：(a) 当 $\neg\texttt{isRight}(i)$ 时，以及 (b) 当 $\neg P(\texttt{parent}(i)) \wedge \texttt{isRight}(i)$ 时。

情况 (a) 注意到 $r(2i) = 2 \cdot \texttt{parent}(i) + \texttt{isRight}(i) = 2 \cdot \texttt{parent}(i)$。然后

$$r^k(2i) = r^{k-1}(r(2i)) = r^{k-1}(2 \cdot \texttt{parent}(i))$$

由此我们可以推导出

$$\left\lfloor \frac{r^k(2i)}{2} \right\rfloor = \left\lfloor \frac{r^{k-1}(2 \cdot \texttt{parent}(i))}{2} \right\rfloor = \texttt{ancestor}_L(\texttt{parent}(i)) = \texttt{ancestor}_L(i)$$

情况 (b) 当 $\neg P(\texttt{parent}(i)) \wedge \texttt{isRight}(i)$ 时，我们有 $r(2i) = 2 \cdot \texttt{parent}(i) + \texttt{isRight}(i) = 2 \cdot \texttt{parent}(i) + 1$ 且

$$\begin{aligned} r(2 \cdot \texttt{parent}(i) + 1) &= 2 \cdot \texttt{parent}\left(\left\lfloor \frac{2 \cdot \texttt{parent}(i) + 1}{2} \right\rfloor\right) + \texttt{isRight}\left(\left\lfloor \frac{2 \cdot \texttt{parent}(i) + 1}{2} \right\rfloor\right) \\ &= 2 \cdot \texttt{parent}(\texttt{parent}(i)) + \texttt{isRight}(\texttt{parent}(i)) \\ &= r(2 \cdot \texttt{parent}(i)) \end{aligned}$$

其中，我们在第一步使用了 $\neg P(\texttt{parent}(i))$。然后可以立即得出如下结果。

$$\begin{aligned} \left\lfloor \frac{r^k(2i)}{2} \right\rfloor = \left\lfloor \frac{r^{k-1}(2 \cdot \texttt{parent}(i) + 1)}{2} \right\rfloor = \left\lfloor \frac{r^{k-1}(2 \cdot \texttt{parent}(i))}{2} \right\rfloor &= \texttt{ancestor}_L(\texttt{parent}(i)) \\ &= \texttt{ancestor}_L(i) \end{aligned}$$

□

推论 8.15. 设 e 是一棵大小为 n 的树。那么

$$\texttt{ancestor}_L(i) = \left\lfloor \frac{r^n(2i)}{2} \right\rfloor$$

证明。具有 n 个节点的树中，叶子的最大深度为 $n-1$。因此，对于所有节点 $n > \texttt{depth}(i)$，都有 i，故根据命题 8.14，等式成立。 □

推论 8.16。 设 e 为一棵大小为 n 的树。计算 e 的有限函数 $\texttt{ancestor}_L$ 和 $\texttt{ancestor}_R$ 的 *PRAM CREW* 时间复杂度为 $O(\log n)$。

证明。令 r 表示 r_L 或 r_R。计算 r 的时间复杂度为 $O(1)$，因为底层数组的每个元素都可以在常数时间内并行计算。复合 $r \fatsemi r$ 需要常数并行随机存取机 PRAM CREW 时间，因为结果数组的每个元素都只需进行一次查找，并且每个元素可在 $O(2n+1)$ 个处理器上并行计算。因此，通过“重复平方”计算 r^n 的时间复杂度为 $O(\log n)$。也就是说，令 $f^0 = r$、$f^{n+1} = f^n \fatsemi f^n$，则 $f^{\log n} = r^n$。 □

注 8.17。 在并行计算文献中，对函数图进行“重复平方”的技术称为“指针跳跃”。例如参见 *[19, 2, 2]*。还需注意，尽管其时间复杂度为对数级别，但此处给出的算法并非工作高效，需要 $O(n\log n)$ 次运算。我们预计可以将其改进为 $O(n)$，但将其留作未来工作。

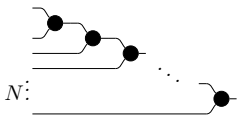
最后请注意，定义 a_L 和 a_R 中的祖先映射8.3 与定义 8.10中的函数并不完全相同。不过，只需进行简单的逐元素算术运算即可在这两个有限函数之间转换，因此我们省略细节。

9 将函子应用于图表

现在我们给出一种高效的并行算法，用于将严格对称幺半超图函子应用于图表。下文中我们将简称其为“函子”，并假定具有严格对称幺半超图结构。请注意，在下文中，我们假设签名 Σ 和 Ω 是任意的，并且可以选择包含所选的特殊 Frobenius 结构**Frob**。

这里给出的算法无需先将图表分解为一个 Σ-项。下面的例子说明了这一点对于 $\mathbf{Diagram}_\Sigma$ 中未配备超图结构的图表的重要性。

例 9.1。 假设我们在范畴 $\mathbf{Free}_{\mathbf{CMon}}$ 中工作。终对象映射 $!_N : N \to 1$表示为如下字符串图。



将其组合地表示为图表后，它具有 $O(N)$ 条内部线、生成元和边。然而，假设我们通过取“垂直切片”来分解它，那么分解中的项数为 $O(N^2)$。第一个“切片”是 $A \otimes A \to A \otimes$ 恒等态射 $\otimes \ldots \otimes$ 恒等态射，包含 $N-2$ 个恒等态射的副本。第二个“切片”包含 $N-3$ 个副本，依此类推，因此每个切片的平均规模为 $N/2$，共 N 个切片。

因此，若天真地将一个项分解为一个 Σ-项，表示规模可能会受到二次惩罚。我们的方法是完全避免这种转换。也就是说，我们直接将函子应用于图表，而不通过 Σ-项进行“往返转换”。此外，在第 10 节中，我们将展示如何利用这一方法，快速计算反向导数（更一般地说，如何利用超图结构，将图表映射为由 光学态射 构成的图表）。

我们的算法基于命题 5.4中定义的 Frobenius 分解。这些分解将每个态射化为由 Frobenius 蜘蛛和生成元的张量积组成的形式。给定图表范畴之间的一个严格对称幺半超图函子F： 图表$_\Sigma \to$ 图表$_\Omega$，我们的方法是将 图表$_\Sigma$ 中的 Frobenius 分解映射为 图表$_\Omega$中的 Frobenius 分解。因此，要给出 F作用的算法，只需定义两项内容：（1）对蜘蛛的作用；（2）对生成元（张量积）的作用。

本节结构如下。第 9.1 节定义一种辅助数据结构（“分段有限函数”）。我们用它来编码 F 的对象映射，并给出一个对 F 的 Frobenius 蜘蛛进行作用的算法图表$_\Sigma$。第 9.2节接着展示如何用同一数据结构编码 箭头映射 F，并类似地给出一个将 F 作用于生成元张量积的算法。最后，我们证明，应用编码后的函子 F 具有线性（顺序）和对数（并行）时间复杂度。

9.1 对象、（半）蜘蛛与分段 FiniteFunction

给定签名 Σ 和 Ω 以及函子 $F : \mathbf{Diagram}_\Sigma \to \mathbf{Diagram}_\Omega$，我们必须对$F$ 的数据进行编码，使其对图表的应用能够并行化。我们做出两项简化假设：(1) 签名 Σ 和 Ω 是有限的；(2) 二者的类型关系都是一个

函数。（后一项假设在一定程度上比所需条件更严格；我们将在第 11 节讨论一个较弱的条件。）

函子 $F : \mathbf{Diagram}_\Sigma \to \mathbf{Diagram}_\Omega$ 由两个映射组成。一个作用于对象 F_0 的函数 $\Sigma_0^* \to \Omega_0^*$，以及一个作用于态射 F_1 的函数 $\mathbf{Diagram}_\Sigma(A, B) \to \mathbf{Diagram}_\Omega(F(A)\ F(B))$。在本节中，我们将介绍一种辅助数据结构——“分段有限函数”，它将使我们能够以适合并行应用的方式对这些映射进行编码。

为说明这一点，考虑对象映射 F_0。由于 F 是范畴图表之间的严格 对称幺半图函子，因此我们有如下结论。

$$F_0\left(\sum_{i\in N} A_i\right) = \sum_{i\in N} F_0(A_i)$$

因此，我们可以认为 F 的对象映射具有类型 $F_0 : \Sigma_0 \to \Omega^*$，因为（根据严格性）它完全由其在生成对象上的作用确定。类似地，我们认为箭头上的映射具有类型 $F_1 : \Sigma_1 \to \mathbf{Diagram}_\Omega$.[3]

现在，对于每个 $i \in \Sigma_0$，我们可以将 $F_0(i) : \Omega_0^*$ 视为一个有限函数 $F_0(i) : s(i) \to \Omega_0$，其中$s : \Sigma_0 \to K$ 是一个函数，用于表示每个列表 $s(i) = |F_0(i)|$的长度（源），而 K 是这些长度中的最大值。对对象映射 F_0 进行编码，意味着将这些有限函数的数据存储在一个“扁平”的数组结构中，其中数组 s 编码了它们的源。

定义9.2。 令 $f_i : A_i \to B_i$ 为一组 N 有限函数。一个分段有限函数 将它们的源、目标和数组数据编码为以下三个映射。

$$\mathsf{sources} : N \to \max_{i\in N} \mathsf{source}(f_i) \qquad \mathsf{targets} : N \to \max_{i\in N} \mathsf{target}(f_i)$$

$$\mathsf{sources}(i) = \mathsf{source}(f_i) \qquad \text{目标}(i) = \text{目标}(f_i)$$

$$\mathsf{values} : \sum_{i\in N} \mathsf{sources}(i) \to \max_{i\in N} \mathsf{target}(f_i)$$

$$\mathsf{values} = \sum_{i\in N} (f_i \fatsemi \iota_0)$$

注9.3。 在并行编程术语中，值 是一个分段数组，其分段大小由源给出。请注意，我们还存储了每个态射的目标，这对于进行张量积是必要的。

现在，给定一个索引函数，我们就可以按任意顺序取函数 f_i 的余积，如下所示。

命题9.4。 设 $f_i : A_i \to B$ 是一组共用同一陪域的有限函数，并且$x : X \to N$ 是一个有限函数。那么 x-索引余积的计算如下。

$$\sum_{i\in X} f_{x(i)} : \sum_{i\in X} A_{x(i)} \to B$$

$$\sum_{i\in X} f_{x(i)} = \left(\sum_{i\in X} \iota_{f(i)}\right) \fatsemi \mathsf{values}$$

证明。 我们可以计算 $\left(\sum_{i\in X} \iota_{x(i)}\right) \fatsemi$值 $= \sum_{i\in X} f_{x(i)} \fatsemi \iota_0 = \sum_{i\in X} f_{x(i)}$，因为 $\iota_0 = \mathsf{id}$，因为所有 f_i 的目标都相等。还要注意，源 $\sum_{i\in X} A_{x(i)}$ 可以通过数组中各项的求和来计算。

□

这种索引余积正是我们将函子对象映射 F_0 应用于*Frobenius* 蜘蛛所需要的。我们先精确定义对象映射的编码方式。

[3]注意，严格来说，$\mathbf{Diagram}_\Omega$ 的态射是等价类，但映射 F_1 将生成元映射到特定的图表。为避免引入新的记号，我们略过这一细节。

定义 9.5。 给定有限签名 Σ 和 Ω 以及严格对称幺半超图函子 F : 图表$_{\Sigma}$ $\to$ 图表$_{\Omega}$, F 的对象映射编码 是一个分段有限函数，其源 为 $= s$，值 为 $= v$，且目标(i) $=$ Ω，其中

$$s : \Sigma_0 \to K \qquad\qquad v : \sum_{i \in \Sigma_0} s(i) \to \Omega_0$$

$$s(i) = |F_0(i)| \qquad\qquad v = \sum_{i \in \Sigma_0} F_0(i)$$

$$v \qquad\qquad \iota_x \quad s \; x \; \to$$

在上文中，余积源($\sum_{i \in \Sigma_0} s \; i$)

$= \sum_{i \in \Sigma_0} s(i)$ 的单射具有如下类型：( ) ( )。根据余积的定义，我们有如下结果。

$$\iota_x \fatsemi v : s(x) \to \Omega_0$$
$$\iota_x \fatsemi v = F_0(x)$$

换句话说，将值的分段数组与一个单射进行复合，相当于取出数组中对应特定“分段”的切片。

将对象映射 F_0 以适当方式表示后，我们现在可以说明如何把 Frobenius 蜘蛛 f =(s, t,L(B)) 在图表Σ 中的数据映射到图表Ω中对应的蜘蛛。回顾一下，任何 Frobenius 蜘蛛都可以看作两个半蜘蛛的复合（定义 3.5）：f =(s, id, L(B)) ⨟(id t, L(B))。因此，只需确定 F 对半蜘蛛的作用。

由于半蜘蛛属于函子 S 的像，我们可以将它们直接视为 Wires 的态射。 F 对这类态射的作用如下。

命题 9.6。 设 f : $A \to B$ 是 Wires 的一个态射，并设 F 是一个严格对称幺半超图函子，其对象编码是一个分段有限函数，其 源 为 $= s$，值 为 $= v$。则 F（S（f） =S（f'）: $A' \to B'$，其中

$$B'(w_n) : \sum_{b \in B(W)} s(B(w_n)(b)) \to \Omega_0 \qquad f' : \sum_{a \in A(W)} s(A(w_n)(a)) \to \sum_{b \in B(W)} s(B(w_n)(b))$$

$$B'(w_n) := \left(\sum_{b \in B(W)} \iota_{B(w_n)(b)} \right) \fatsemi v \qquad f' := \sum_{a \in A(W)} \iota_{f_W(a)}$$

并且 A'（w_n） $= f' \fatsemi B'$（w_n） 由自然性保持不变。

证明。 根据严格性假设，需要给出 B' 的定义。随后必须验证$A'(w_n) = f' \fatsemi B'(w_n)$，这由 f 的自然性得到：

$$\begin{aligned} f'_W \fatsemi B'(w_n) &: \sum_{a \in A(w_n)} s(A(w_n)(a)) \to \Omega_0 \\ &: \sum_{a \in A(w_n)} s(B(w_n)(f(a))) \to \Omega_0 \\ f'_W \fatsemi B'(w_n) &= \left(\sum_{a \in A(W)} \iota_{f_W(a)} \right) \fatsemi \left(\sum_{b \in B(W)} \iota_{B(w_n)(b)} \right) \fatsemi v \\ &= \left(\sum_{a \in A(W)} \iota_{B(w_n)(f(a))} \right) \fatsemi v \\ &= \left(\sum_{a \in A(W)} \iota_{A(w_n)(a)} \right) \fatsemi v \end{aligned}$$

最后，我们必须检查超图结构是否得到保留，以及 F 是否满足函子律。这通过归纳法来完成：在基例中，可以直接验证该定义对于 S 的像中的生成操作成立。张量积的归纳步骤同样如此。

显然。可以如下验证复合的情形。假设 $A \xrightarrow{f} B \xrightarrow{g} C$ 是 Wires 的态射。那么

$$f' \fatsemi g' : \sum_{a \in A(W)} s(B(w_n)(f(a))) \to \sum_{c \in C(W)} s(C(wn)(c))$$

$$f' \fatsemi g' = \left(\sum_{a \in A(W)} \iota_{f(a)} \right) \fatsemi \left(\sum_{b \in B(W)} \iota_{g(b)} \right) = \sum_{a \in A(W)} \iota_{g(f(a))} = (f \fatsemi g)'$$

最后，由于 F 是超图函子，根据 S 的唯一性，我们有 $F(\mathsf{S}(f)) = \mathsf{S}(f')$。 □

注释 9.7。 请特别注意，上述 B' 和 f' 的定义中所用的单射并不相同。B' 的定义中所用单射的索引遍历 v 的线段，而 f' 的定义中所用单射的索引则遍历 $B(W)$ 的元素。

映射后的蜘蛛中的对象 $B'(w_n)$ 恰好是值表的一个以 $B(w_n)$ 为索引的余积。类似地，f' 的值由“带索引的单射”给出。为了计算这两者，我们使用以下算法。给定有限函数 $s : N \to K$ 以及映射 $x\colon X \to N$（将其视为索引数组），`injections(s, x)` 计算单射的余积 $\iota_{x(0)} \cdots \iota_{x(A-1)}$。

```
def injections(s, x):
    p = prefix_sum(s)
    r = segmented_arange(x >> s)
    return FiniteFunction(sum(s), r.table + repeat(x >> p, x >> s).table)
```

请特别注意，上述算法中的 $+$ 运算表示整数数组的逐点求和，而不是余积。给定 F 的对象映射编码后，就可以计算 f' =`injections(s, f >> B(w_n))` 和 $B'(w_n)$ = `injections(s, B(w_n)) >> v`。

命题 9.8. 设 $s : N \to K$ 和 $x : X \to N$ 是有限函数。计算 `injections` 的顺序时间复杂度为 $O(N) + O(X) + O(\mathtt{sum}(x \fatsemi s))$，其 *PRAM CREW* 复杂度为 $O(\log N) + O(\log X)$。

证明。在顺序情形下，两次函数复合均为 $O(X)$，前缀和为 $O(N)$，其余运算为 $O(\mathtt{sum}(x \fatsemi s))$。在并行情形下，相同运算分别为 $O(\log N)$、$O(1)$ 和 $O(\log X)$。 □

请注意，使用函数 `injections` 计算 f' 和 $B'(w_n)$ 时，`sum`$(x \fatsemi s)$ 与 $B'(W)$ 成正比，因此蜘蛛计算的顺序复杂度与结果大小呈线性关系。

9.2 生成元张量积的映射

定义了 F 在 Frobenius 分解的蜘蛛 上的作用后，现在还需定义其在生成元的张量积上的作用。这种张量积是形如 $g = (\iota_0, \iota_1, G)$ 的图表，其数据由列表 $G(x_n) : G(X) \to \Sigma_1$ 在同构意义下确定。换言之，由 g 表示的图表同构类属于 F 下 Σ-项 $\bigotimes_{i \in G(X)} G(x_n)(i)$ 的像，因此根据严格性，必须满足 $F(g) \cong \bigotimes_{i \in G(X)} F_1(G(x_n)(i))$。

与对象映射一样，F 在态射上的作用完全由其在运算 Σ_1上的作用定义。因此，我们可以认为其类型为 $F_1 : \Sigma_1 \to$ 图表$_\Omega$。这使得顺序情形下的实现十分直接。将 F_1 的数据编码为一个长度为$|\Sigma_1|$ 的列表，其中第 i^{th} 个条目是图表 $F_1(i)$，那么在伪代码中，我们可以写作

```
def apply_functor(F1: List[Diagram], g: Diagram):
  diagrams = [ F1[g.G.xn(i)] for i in range(0, g.X) ]
  Diagram.tensor_list(diagrams)
```

图表的 N 重张量积是二元情形的直接推广。回想一下，图表本质上是有限映射的集合，而余积和张量积都是逐点进行的。因此，对一列图表取张量积，相当于取有限

余积，而不是二元余积。并行情形在“本质上”是相同的，但要得到恰当的对数时间算法，需要特别谨慎：对一列有限函数计算有限张量积，并不能直接推广到并行情形。下面来看一个朴素的实现。

```
def coproduct_list(fs: List[FiniteFunction]):
    return FiniteFunction(fs[0].cod, concatenate([f.table for f in fs]))
```

问题在于，构造 `concatenate` 的参数所需的时间与 `fs` 的大小成线性关系。由于 `fs` 的长度等于 $G(X)$，因此所需时间与图表中的运算数量成线性关系，所以在并行情形下无法获得加速。有限映射的张量积也存在同样的问题。

解决方案是将 F_1 的数据编码为“列表的图表”，而不是图表列表。更准确地说，我们将每个图表分量分别编码为一个分段有限函数。借助这种表示，我们能够提取任意余积和“线段”的张量积来构造结果。

F_1 的编码是一组分段有限函数：源映射和目标映射各对应一个，而每个分量 $G(f)$（$f \in \mathcal{S}$）也各对应一个。

定义 9.9。设 Σ 和 Ω 为有限签名，且 $F : \mathsf{Diagram}_\Sigma \to \mathsf{Diagram}_\Omega$ 为严格对称幺半超图函子，并用 $d_i = (s_i, t_i, G_i)$ 表示图表集合 $d_i = F_1(i)$，其中 $i \in \Sigma_1$。F 的箭头映射编码 由分段有限函数构成，分别对应于 s_i、t_i，以及模式 S 中每个分量 f 所对应的 $G_i(f)$。

为了计算图表的张量积，有必要从分段有限函数计算带索引的张量积。

命题 9.10. 设 $f_i : A_i \to B_i$ 是一组 N 有限函数，且 $x : X \to N$ 是为该集合编索引的有限函数。那么，该带索引的张量积由以下数组给出

$$\begin{aligned}\bigotimes_{i \in X} f_{x(i)} : \sum_{i \in X} A_{x(i)} &\to \sum_{i \in X} B_{x(i)} \\ &= \sum_{i \in X} f_{x(i)} \oplus \mathsf{repeat}(x \fatsemi \mathsf{sources}, p)\end{aligned}$$

其中，$\oplus$ 表示自然数的逐元素相加，p 表示陪域 $p(i) = \sum_{j \in i} \text{目标}(x(j))$ 的部分和。

证明。回顾一下，张量积可以按如下方式用余积表示。

$$\bigotimes_{i \in X} f_{x(i)} = \sum_{i \in X} (f_{x(i)} \fatsemi \iota_i)$$

这里，每个单射的类型为

$$\iota_i \quad B_{x(i)} \to \textstyle\sum_{j \in X} B_{x(j)} \qquad\qquad p\ i \quad p\ i \ +\text{：其数组数据由}$$

$$1 \ldots p\ i + B_i - 1 \qquad\qquad f_{x(i)} \fatsemi \iota_i \qquad\qquad f_{x(i)}$$

$\langle (\,)\,(\,)\,,,(\,)\,\rangle$ 给出。因此，复合的数组数据就是 再加上常数 $p(i)$，我们可以将其改写如下

$$\sum_{i \in X} (f_{x(i)} \fatsemi \iota_i) = \sum_{i \in X} f_{x(i)} \oplus \mathsf{repeat}(x \fatsemi \mathsf{sources}, p)$$

□

计算张量积 $h = (\iota_0, \iota_1, H)$ 的 $F(h)$ 十分直接。由于余积和张量积都是逐点的，每个包含 $F(h)$ 数据的有限函数，都是其对应分段有限函数的 $H(x_n)$-索引余积（或张量积）。

命题 9.11。将箭头映射应用于生成元的 N 重张量积 (ι_0, ι_1, g) 的复杂度为 $O(N)$（顺序）和 $O(\text{对数}\ N)$（并行随机存取机），以 g 的运算次数计。

证明。 与推论 7.8中一样，图表中的导线和边的数量与 N成正比。应用 F_1 需要为 G的每个分量态射计算由 $G(x_n)$ 索引的余积和张量积，因此其复杂度与这些运算相同。根据命题 $O(N)$，这些运算的复杂度为 O(对数 N)（并行随机存取机），而顺序计算的复杂度为 $O(N)$ 。参见命题 9.8。 □

推论 *9.12.* 设 $F : \mathit{Diagram}_{\Sigma} \to \mathit{Diagram}_{\Omega}$ 是一个严格对称幺半图函子，且 $d = (s, t, G)$ 是 $\mathit{Diagram}_{\Sigma}$ 中类型为 $d : A \to B$ 的一个图表。给定对象映射与态射映射编码，计算 $F(d)$ 的顺序时间复杂度与 d 的导线数量 $G(W)$、边数量 $G(E_i)$、$G(E_o)$、运算数量 $G(X)$ 以及边界数量 $A(W)$ 和 $B(W)$ 呈线性关系。并行时间复杂度与上述数量呈对数关系。

10 使用 Frobenius 结构进行光学态射复合

我们现在展示如何利用图表$_{\Sigma+\mathbf{Frob}}$ 的超图结构来编码光学。我们的方法受到 [3] 和 [15]的弦图的启发。结合第9节所描述的函子作用并行算法，我们可以定义一种将图表映射为光学图的算法。一般而言，这使我们能够对具有双向信息流的系统进行建模。这类系统的一个例子是将神经网络 [11] 视为具有反向导数 [9]的范畴中的态射。我们将专门探讨这一例子，并定义一种可扩展的方法，用于计算大型图表的反向导数。此外，这也解决了 [15]中指出的朴素反向导数计算方法所存在的效率问题。

与第 9 节一样，我们假设有任意签名 Σ 和 Ω，现在还假设它们包含额外选定的特殊 Frobenius 结构。我们先非正式地回顾一下光学态射。

定义 10.1 (非正式定义， [15, 22])。 类型为 $\binom{\overrightarrow{A}}{\overleftarrow{A}} \to \binom{\overrightarrow{B}}{\overleftarrow{B}}$ 的一个光学态射 是一个三元组

$$M \in \mathscr{C} \qquad \overrightarrow{f} : \overrightarrow{A} \to \overrightarrow{B} \otimes M \qquad \overleftarrow{f} : M \otimes \overleftarrow{B} \to \overleftarrow{A}$$

其中，我们分别将 $\overrightarrow{f}$和 $\overleftarrow{f}$ 称为正向映射和反向映射。

整个光学态射被视为一个模拟前向和反向信息流的系统。该 $\overrightarrow{f}$态射捕捉了模型的“前向”信息流，将数据 $\overrightarrow{A}$映射到 $\overrightarrow{B}$。对象 M是输入 $\overrightarrow{A}$的一些“记忆”，它会传递给“反向”映射 $\overleftarrow{f}$，后者将“类似输出”的数据和记忆映射回“类似输入”的数据。在反向导数的特定情形下（见第10.1节），“类似输入”和“类似输出”分别表示输入和输出中的变化。

请注意，严格来说，光学范畴中的态射是三元组 $(M,\ f,\ f')$ 的等价类。这不会影响我们。相反，本节的重点是如何将此类的代表元——特定的三元组 $(M,\ f,\ f')$——编码为图表。本节的主要贡献在于认识到，可以利用 Diagram$_{\Sigma+\mathbf{Frob}}$ 中存在的超图结构来“模拟”光学态射复合。为了对此进行形式化表述，首先有必要定义交错态射。

定义 10.2。 假设对每个生成元 $X \in \Sigma_0$，都有一对对象 $\overrightarrow{X} \in \Omega_0^*$ 和 $\overleftarrow{X} \in \Omega_0^*$。那么令 $A = A_0 \otimes A_1 \otimes \ldots \otimes A_{N-1}$ 为 Σ_0中的生成对象张量积。 A处的交错 是具有以下类型的置换。

$$\mathsf{interleave}_A : \bigotimes_{i \in N} \overrightarrow{A_i} \otimes \bigotimes_{i \in N} \overleftarrow{A_i} \longrightarrow \bigotimes_{i \in N} (\overrightarrow{A_i} \otimes \overleftarrow{A_i})$$

对于每个生成操作 $f :\ A \to B$，选定正向映射与反向映射后，这种交错方式使我们能够定义一个严格幺半群函子，将每个 f 映射到其选定的光学态射。

Defi定义 10.3. 设 Σ 和 Ω 为幺半群签名，并假设给定了以下数据。

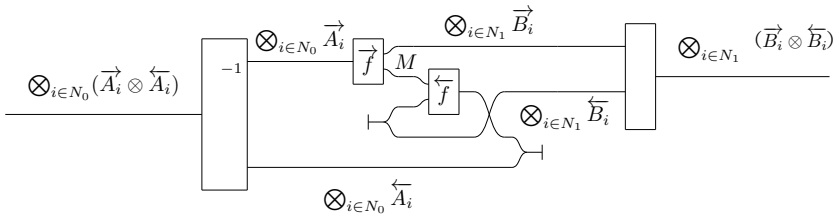
- 对于每个生成对象 $A \in \Sigma_0$，一对对象（列表） $\overrightarrow{A} \in \Omega_0^*$ 和 $\overleftarrow{A} \in \Omega_0^*$

- 对于 Σ_1 中的每个生成操作 $f:\ \bigotimes_{i\in N_0} A_i \to \bigotimes_{i\in N_1} B_i$，一对态射

  - $\overrightarrow{f} : \bigotimes_{i\in N_0} \overrightarrow{A_i} \longrightarrow \left(\bigotimes_{i\in N_1} \overrightarrow{B_i}\right) \otimes M,$
  - $\overleftarrow{f} : M \otimes \left(\bigotimes_{i\in N_1} \overleftarrow{B_i}\right) \longrightarrow \bigotimes_{i\in N_0} \overleftarrow{A_i}$

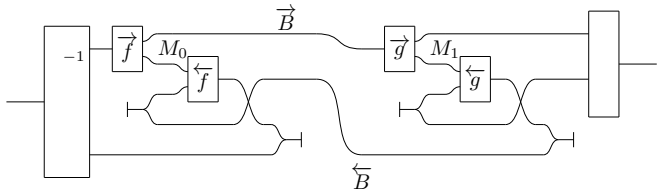
记 $\mathcal{O}:$ 图表$_\Sigma \to$ 图表$_\Omega$ 为通过归纳定义的严格对称幺半超图函子，其在生成对象上的作用为 $F(A) = \overrightarrow{A} \otimes \overleftarrow{A}$，在态射上的作用如下所示。

$F(f) :=$



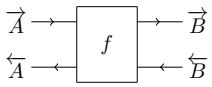
注意，$\mathcal{O}$ 的像中的图表以与光学态射相同的方式复合。

$\mathcal{O}(f \fatsemi g) = \mathcal{O}(f) \fatsemi \mathcal{O}(g) =$

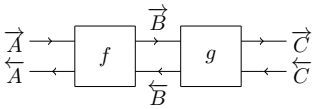


直观地说，标记为 $\overleftarrow{A}$ 和 $\overleftarrow{B}$ 的导线中的“信息流”方向是从右向左：$\overleftarrow{g}$ 的输出连接到 $\overleftarrow{f}$ 的输入，而后者又连接到左边界。

将此与 [3], 的图形语法进行比较。在该语法中，作者使用有向线为 Tambara 模构造了一种图表语言。光学则是这种语言中具有以下类型的图表。

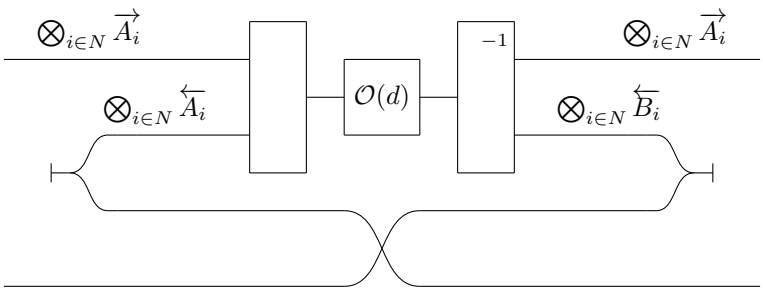


这些图表的复合方式如下所示，与此类似。



属于 $\mathcal{O}$ 像中的图表一般不具有这种类型，但通过与交错进行前复合和后复合，可以轻松地对其进行调整。

这很有用，因为对于给定的 $d:\ A \to B$ 图表$\mathcal{O}(d)$ 并非单配无环（定义6.1 和6.2），因为每个输出都 $\overleftarrow{f}$连接到左侧边界。然而，如果对于所有 $f \in \Sigma_1$，每个 $\overrightarrow{f}$和 $\overleftarrow{f}$都是单配无环的，那么就可以使用如下的交错态射将所得图表转换为单配无环图表。



(17)

这对我们的案例研究很重要，因为这意味着我们可以提取一个态射，并将其解释为对称幺半范畴中的一个函数。这意味着它可以被视为一个函数，同时计算反向导数（视为光学态射）的正向映射与反向映射。

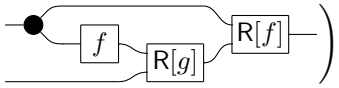
10.1 案例研究：通过光学态射组合实现高效反向导数

作为论文的结尾，我们现在讨论迄今为止给出的构造如何应用于机器学习领域。具体而言，我们将展示如何利用超图结构对光学态射进行编码，从而高效地计算大型图表的反向导数。

我们先非正式地回顾一下反向导数，它是 [11] 中所述基于梯度的学习形式化过程的一个关键分量。一个（笛卡尔的）反向导数范畴（RDC）为态射 $f\colon A \to B$ 配备了一个反向导数$\mathsf{R}[f]$：$A \times B' \to A'$，并满足若干公理。直观地说，它将输入 A 和输出的变化 B' 映射为输入 的变化 A'。

反向导数也可以看作透镜：由一对态射 $(f, \mathsf{R}[f])$ 构成，并按照下图所示的方式进行复合。

$$(f, \mathsf{R}[f]) \fatsemi (g, \mathsf{R}[g]) \quad = \quad \Big(f \fatsemi g \quad , \quad$$

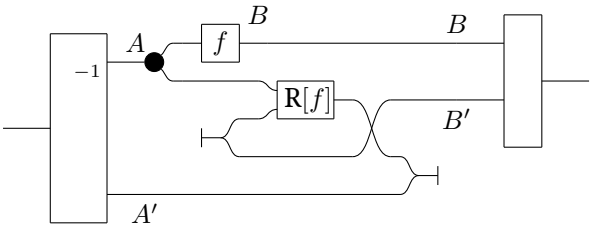


$$\Big)$$

这个定义带来了两种低效之处。首先，对于每个态射，我们都必须表示两个不同的图表。这意味着无法应用第 9 节中定义的那类函子。因此，为了将映射 f 的反向导数计算为一个透镜，似乎必须先提取出一个 Σ 项，再逐步构建一对图表。正如示例 9.1 所示，提取这样的 Σ 项可能会带来二次惩罚。第二个问题在 [15]，中被指出：透镜组合的定义造成了以空间换时间的权衡，导致透镜的正向态射出现多个冗余“副本”。

为避免这种冗余计算，我们可以利用 [15, 图 10] 所展示的观察结果，其中说明了透镜如何复合作为光学态射。这依赖于这样一个事实：透镜是基范畴为笛卡尔范畴的光学态射的特殊情形。就光学态射而言，透镜的正向映射具有$^{\prime}$ 的形式，其中为空间的对角映射。因此，我们可以将RDC中的一个态射 f 映射为如下光学态射。

$$\mathcal{O}(d) \quad =$$



完成上述步骤后，我们便可以使用(17) 来提取RDC中的一个态射，使其同时运行正向和反向过程。这带来了两个具体好处：(1) 我们可以并行地高效计算反向导数的大型符号描述；(2) 完全避免分解为 Σ项。

请注意，这里所介绍的内容比反向导数更加一般。由于我们已经给出了应用超图函子的算法，因此可以将其应用于那些迹由超图结构给出的带迹对称幺半范畴。我们推测，这将使对带有反馈的双向信息流系统进行建模成为可能。更准确地说，这里给出的算法将允许我们把带反馈的一个态射映射为一个带反馈的光学态射。

11 讨论与未来工作

本文介绍的数据结构和算法在多个方面改进了以往工作 [26]。我们处理了配备特殊 Frobenius 结构的图表，扩展到任意集合的生成对象，并消除了对底层稀疏数组实现的依赖。此外，原生支持配备特殊 Frobenius 结构的图表，使得能够表示光学图。进而，这使我们能够给出一种计算反向导数的并行算法。

然而，与 [26], 一样，仍有许多改进空间。除了匹配和重写算法之外，未来工作还应包括求值图表的算法。尽管我们预计

这项工作可以轻松完成，但内部图连线的其他表示方式可能会影响求值算法的效率。例如，我们目前将图表的运算存储在单个数组 $G(x_n) : G(X) \to \Sigma_1$中。不过，出于性能考虑，存储多个这样的数组可能更有优势，这样便能更高效地在 GPU 硬件上并行执行运算。

或许也可以放宽本文对 PRAM CRCW 模型的假设。由于在大多数情况下 CREW 模型已足够，因此只需做少量修改。例如，计算余等化子泛态射的算法需要 CRCW 假设，但或许可以通过用整数排序替代并发写入，得到一个 PRAM CREW 算法。此外，我们或许还能提高本文所介绍的一些并行算法的工作效率，尤其是祖先映射（推论 8.16）。

最后，在第 9 节定义将函子 F 应用于图表的算法时，我们要求类型关系 τ 是一个函数。然而，我们这种表示方式的优势之一，在于允许使用多态生成元。因此，一个自然的扩展方向是允许多态生成元之间的范畴间函子。为此，在映射给定运算时需要使用类型信息，因为不再能仅凭运算标签 $g \in \Sigma_1$确定唯一的图表 $F(g)$。

参考文献

[1]John C. Baez 和 Kenny Courser。《结构化余跨》。2019年。DOI: 10.48550/ARXIV.1911.04630。URL https://arxiv.org/abs/1911.04630.[2] John C. Baez、Kenny Courser 和 Christina Vasilakopoulou。《结构化余跨与装饰余跨》。《组合性》，4:3，2022年9月。DOI: 10.32408/compositionality-4-3。URL https://doi.org/10.32408%2Fcompositionality-4-3.[3] Guillaume Boisseau。《光学的弦图》，2020年。[4] Filippo Bonchi、Fabio Gadducci、Aleks Kissinger、Paweł Sobociński 和 Fabio Zanasi。《对称幺半结构模下的重写》。载于第31届 ACM/IEEE 逻辑与计算机科学年会论文集。ACM，2016年7月。DOI: 10.1145/2933575.2935316。URL https://doi.org/10.1145%2F2933575.2935316.[5] Filippo Bonchi、Pawel Sobociński 和 Fabio Zanasi。《组合信号流理论综述》。载于《推进信息与通信技术研究》，第 AICT-600 卷，第 29–56 页。2021年。DOI: 10.1007/978-3-030-81701-5_2。URLhttps://hal.inria.fr/hal-03325995。专题: TC 1: 计算机科学基础。[6] Albert Burroni。《高维字问题及其在等式逻辑中的应用》。《理论计算机科学》，115(1): 43–62，1993年。[7] Matteo Capucci、Neil Ghani、Jé rémy Ledent 和 Fredrik Nordvall Forsberg。《将扩展式博弈转换为具有主体性的开放博弈》。《理论计算机科学电子论文集》，372: 221–234，2022年11月。DOI: 10.4204/eptcs.372.16。URL https://doi.org/10.4204%2Feptcs.372.16.[8] A. Carboni 和 R.F.C. Walters。《笛卡尔双范畴 I》。《纯粹与应用代数杂志》，49(1): 11–32，1987年。ISSN 0022-4049。DOI: https://doi.org/10.1016/0022-4049(87)90121-6。URL https://www.sciencedirect.com/science/article/pii/0022404987901216.[9] Robin Cockett、Geoffrey Cruttwell、Jonathan Gallagher、Jean-Simon Pacaud Lemay、Benjamin MacAdam、Gordon Plotkin 和 Dorette Pronk。《反向导数范畴》，2019年。[10] Thomas H. Cormen、Charles E. Leiserson、Ronald L. Rivest 和 Clifford Stein。《算法导论》，第四版。麻省理工学院出版社，精装版，2022年4月。ISBN 978-0262046305。[11] G. S. H. Cruttwell、Bruno Gavranović、Neil Ghani、Paul Wilson 和 Fabio Zanasi。《基于梯度学习的范畴论基础》，2021年。URL https://arxiv.org/abs/2103.01931.[12] Brendan Fong。《装饰余跨》，2015年。[13] Brendan Fong。《开放互联系统的代数》，2016年。URL https://arxiv.org/abs/1609.05382.[14] Thomas Fox。《余代数与笛卡尔范畴》，1976年1月。URL http://dx.doi.org/10.1080/00927877608822127。

[15] Bruno Gavranović。《通过高阶范畴论研究透镜与光学的时空权衡》，2022年。[16] Fabrizio Genovese 和 Jelle Herold。《分层 Petri 网的范畴语义》，2021年。[17] Fabrizio Romano Genovese、Fosco Loregian 和 Daniele Palombi。《带有法力的网：化学反应建模框架》，2021年。[18] Dan R. Ghica、George Kaye 和 David Sprunger。《数字电路的组合理论》，2022年。[19] Joseph JaJa。《并行算法导论》。Addison Wesley，美国马萨诸塞州波士顿，1992年3月。[20] Evan Patterson 等贡献者。Algebraicjulia/catlab.jl：v0.12.2，2021年5月。URLhttps://doi.org/10.5281/zenodo.4736069.[21] Evan Patterson、Owen Lynch 和 James Fairbanks。《技术计算的范畴数据结构》。《组合性》，4:5，2022年12月。DOI：10.32408/compositionality-4-5。URL https://doi.org/10.32408%2Fcompositionality-4-5.[22] Mitchell Riley。《光学范畴》，2018年。[23] P. Selinger。《幺半范畴图形语言综述》。载于《物理学的新结构》，第289–355页。Springer Berlin Heidelberg，2010年。DOI：10.1007/978-3-642-12821-9_4。URL https://doi.org/10.1007%2F978-3-642-12821-9_4.[24] Yossi Shiloach 和 Uzi Vishkin。《一个 O(log n) 并行连通性算法》。《算法杂志》，3(1)：57–67，1982年。ISSN 0196-6774。DOI：https://doi.org/10.1016/0196-6774(82)90008-6。URL https://www.sciencedirect.com/science/article/pii/0196677482900086.[25] Toby St. Clere Smithe。《贝叶斯更新以光学方式复合》，2020年。[26] Paul Wilson 和 Fabio Zanasi。《组合性的代价：字符串图组合的高性能实现》。2021年。DOI：10.48550/ARXIV.2105.09257。URLhttps://arxiv.org/abs/2105.09257。

附录

排序置换

命题 A.1. $p:\ A \to A$ 是 FinFun 中的同构，当且仅当 p 是一个置换 .

证明。 如果 p 是一个置换，那么它显然是一个同构，因为它有逆。因此，只需证明如果 p 是一个同构，那么它就是一个置换。由于 p 是一个同构，它必定是一个双射，因而是一个置换。 □

定义 A.2. 设 $f:\ A \to B$ 是一个函数。若同构 $s:\ A \to A$ 使 A 按f排序，则称其对 A 按f 排序；若 $s \fatsemi f$是非递减的，即对所有 i、$j \in A$、$i < j \implies f(s(i)) \leq f(s(j))$

命题 A.3. 设 $f:\ A \to B$ 是单射，且 $s:\ A \to A$ 按 f排序。那么 $s \fatsemi f$ 是严格递增的，即 $i < j \Leftarrow\Rightarrow f(s(i)) < f(s(j))$。

证明。根据定义，我们有 $i < j \implies f(s(i)) \leq f(s(j))$。我们分两种情况证明。

$i < j \implies f(s(i)) < f(s(j))$ **(情况 1)** 在第一种情况下，我们有 $i < j \implies f(s(i)) \leq f(s(j))$，因为 s 按 f排序。接着假设 $f(s(i)) = f(s(j))$。f和 s 都是单态射，因此它们的复合也是单态射。因此，我们可以利用 $i \neq j \implies f(s(i) \neq f(s(j))$，从而得出 $i < j \implies f(s(i)) < f(s(j))$。

$f(s(i)) < f(s(j)) \implies i < j$（**情况 2**）在第二种情况下，我们证明逆否命题：$i \geq j \implies f(s(i)) \geq f(s(j))$，我们可以将其改写为 $j \leq i \implies f(s(j)) \leq f(s(i))$。这是成立的，因为当 $j = i$时，我们有 $f(s(j)) = f(s(j))$，而当 $j < i$时，假设条件适用。

□

命题 A.4。 设 N 为一个有限集，且 $f : N \to B$ 是一个单态射。存在唯一的同构 sort$f:\ N \to N$，它按照 f进行排序。

证明。设 $r, s\colon N \to N$是按 f排序的两个置换。由于 f是单态射，对于所有$i, j \in N$，我们有 $f(r(i)) < f(r(j)) \iff i < j \iff f(s(i)) < f(s(j))$，根据命题 A.3。于是我们可以计算得出。

$$\begin{aligned} r^{-1}(i) < r^{-1}(j) &\iff f(r(r^{-1}(i))) < f(r(r^{-1}(j))) \\ &\iff f(i) < f(j) \\ &\iff f(s(s^{-1}(i))) < f(s(s^{-1}(j))) \\ &\iff s^{-1}(i) < s^{-1}(j) \end{aligned}$$

现在我们可以用它来证明 $r^{-1} = s^{-1}$。

回想一下，$\overline{i}$ 表示小于 i 的自然数集合，因此根据定义有 $|i| = i$，进而有 $|i| = |j| \Rightarrow i = j$。此外，对于任意置换 $p\colon N \to N$ 和 $j \in N$，有$|j| = |\{i \in N \mid i < j\}| = |\{i' \in N \mid p(i') < j\}|$，因为对于每个 $i \in N$，恰好存在一个 i'，使得 $p(i') = i$，这是因为 p 是一个双射。利用这一点以及 $r^{-1}(i) < r^{-1}(j) \iff s^{-1}(i) < s^{-1}(j)$ 这一事实，我们可以计算出

$$\begin{aligned} r^{-1}(i) &= \left|\overline{r^{-1}(i)}\right| \\ &= \left|\left\{i \in \overline{N} \mid i < r^{-1}(j)\right\}\right| \\ &= \left|\left\{i \in \overline{N} \mid r^{-1}(i) < r^{-1}(j)\right\}\right| \\ &= \left|\left\{i \in \overline{N} \mid s^{-1}(i) < s^{-1}(j)\right\}\right| \\ &= \left|\left\{i \in \overline{N} \mid i < s^{-1}(j)\right\}\right| \\ &= \left|\overline{s^{-1}(i)}\right| \\ &= s^{-1}(i) \end{aligned}$$

因此，根据逆元唯一性，我们有 $r = s$。 □

定义 A.5 (稳定排序)。 由键 $f\colon A \to B$ 进行的稳定排序 是态射 sort_k，其中$k = \langle f, \mathsf{id} \rangle\colon A \to B \times A$

注记 A.6. 在上述定义中，s' 被迫通过 A 它们的“键”f 以及由恒等函数给出的“位置”来区分元素。这样，函数 $\langle f, \mathsf{id} \rangle$ 就成为一个单态射。根据命题*A.4*，存在唯一的排序态射，因此上述定义是良定义的。

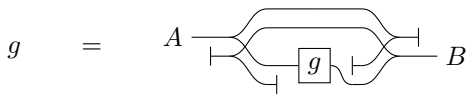
B 第 5 节的证明5

B.1 $\mathcal{D}$证明

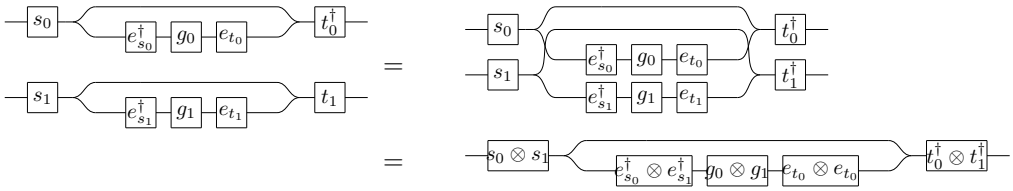
本节中，我们验证 $\mathcal{D}$ (命题 5.7) 定义了一个严格对称幺半超图函子。首先证明前文暂缓证明的结果：任意超图范畴中都存在 Frobenius 分解。

命题 *5.4*的证明。 我们用归纳法证明，超图范畴中的任意态射都有一个Frobenius 分解。

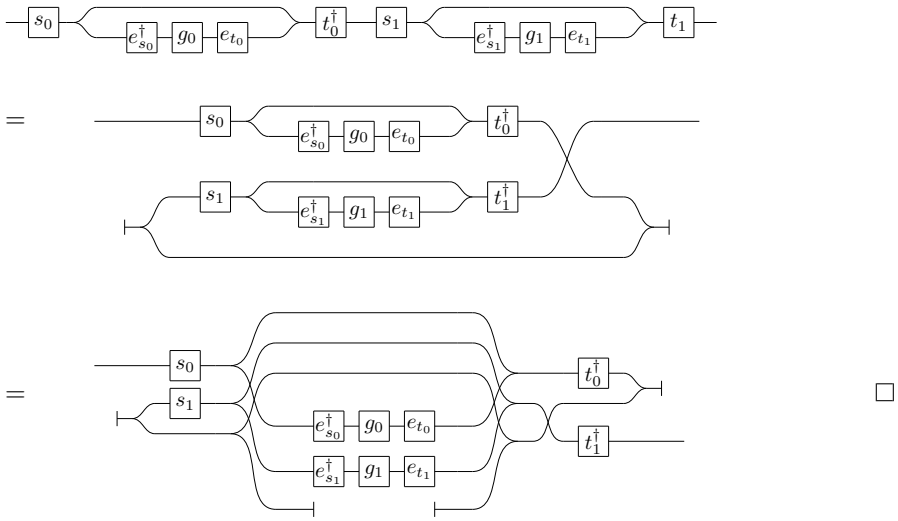
Base Case: 生成元 生成元的 Frobenius 分解 g 是其 单例图 m:



归纳步骤：张量积 $f_0 \otimes f_1$ 设 f_0 和 f_1 为具有 Frobenius 分解的态射。那么 $f_0 \otimes f_1$ 具有 Frobenius 分解。



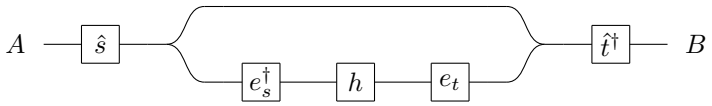
归纳步骤：复合 $f_0 \fatsemi f_1$ 设 f_0 和 f_1 是具有 Frobenius 分解的态射。那么， $f_0 \fatsemi f_1$ 具有 Frobenius 分解。



□

现在我们完成命题 5.5 的证明概要，验证所给出的 Frobenius 分解确实与原图表同构。为此需要以下引理，该引理表明，一个图表可以立即因式分解为“几乎 Frobenius”形式。

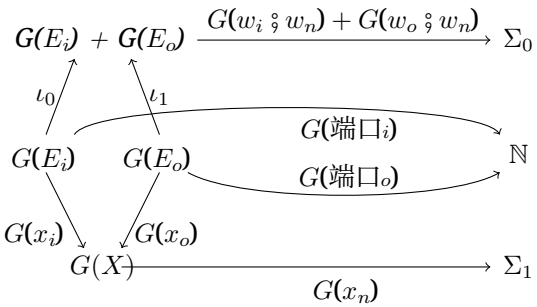
命题 B.1 （几乎 Frobenius 分解）。 设 (s, t, G) 是一个图表。那么可以选择一个余等化子，使得以下“几乎 *Frobenius*”分解与 (s, t, G) 同构。



其中

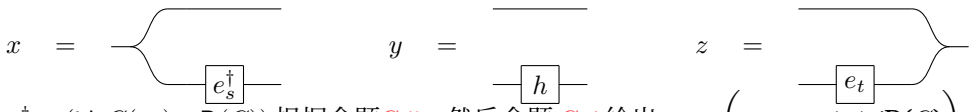
$$\hat{s} = \mathsf{S}(s) \qquad \hat{t} = \mathsf{S}(t) \qquad e_s = \mathsf{S}(G(w_i)) \qquad e_t = \mathsf{S}(G(w_o)) \qquad h = (\iota_0, \iota_1, H)$$

是带有标记 $G(w_n)$ 的半蜘蛛，而 H 根据 G 定义如下。



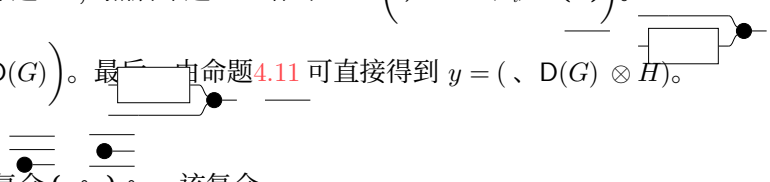
证明。 该证明通过对与适当选取的余等化子的复合进行直接计算来完成。我们将首先关注图表的内部部分，在证明结束前暂时忽略 $\hat{s}$ 和 $\hat{t}$ 。

我们首先计算与以下复合操作相对应的图表。



现在，$e_s^\dagger = (\mathsf{id}\ G(w_i)$、$\mathsf{D}(G))$ 根据命题C.5，然后命题 C.4 给出 $x = \Big($, $G\ (w_i)$ ’$\mathsf{D}(G)\Big)$。

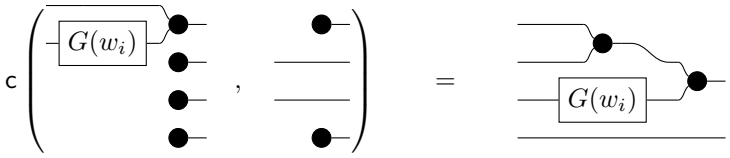
同样的命题还表明 $z = \Big(G_{(w_o)}$ 、$\mathsf{D}(G)\Big)$。最后，由命题4.11 可直接得到 $y = ($ 、$\mathsf{D}(G) \otimes H)$。



现在按照括号所指示的顺序计算复合 $(x \fatsemi y) \fatsemi z$。该复合

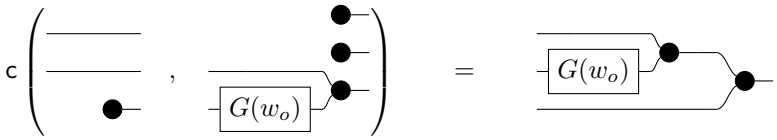
$$(x \fatsemi y) = \Big(\text{—} \ , \ \text{—}\!\boxed{t}\!\text{—} \ , \mathsf{D}(G) \Big) \fatsemi \Big(\text{—} \ , \ \text{—} \ , \mathsf{D}(G) \otimes H \Big)$$

可以通过选取以下余等化子来计算。



这给出了一个余跨度，其源映射为 ι_0，目标映射为 $\mathsf{id} \otimes \mathsf{id}$。

接下来，我们使用以下余等化子的选取来计算 $(x \fatsemi y) \fatsemi z$。



This 分别得到源映射和目标映射 $s = \mathsf{id}$ 和 $t = \mathsf{id}$，以及余跨度的顶点 n eval映到 G。

接下来只需证明，与 $\hat{s}$ 和 $\hat{t}^\dagger$ 进行前复合和后复合(s, t, G)会得到图表 $(x \fatsemi y) \fatsemi z$ 的源映射和目标映射都是恒等态射，因此由命题 C.1可得，证明完成。

□

注 B.2。 请注意，上述内容还不是 Frobenius 分解，因为 h 并非生成元的张量积：其边界未必按所需顺序排列。要得到 *Frobenius* 分解，只需将 h' 分解为形式 $p \fatsemi h \fatsemi q$，其中 h 是生成元的张量积，而 p 和 q 是置换。

由此命题可得证明：命题 5.5 中给出的特定 Frobenius 分解确实通过复合还原出原始态射。

命题 5.5 的证明。首先将图表 (s, t, G) 分解为命题 B.1 中描述的“几乎 Frobenius”形式 $(\hat{s}, e_s^\dagger, h', e_t, \hat{t}^\dagger)$。要得到真正的 Frobenius 分解，只需将图表 $h' = (\iota_0, \iota_1, H')$ 分解为复合 $p \fatsemi g \fatsemi q$，其中 p 和 q 是置换，而 g 是一个与生成元的 n 重张量积同构的图表。

为得到 g，我们只需置换 h' 的导线，得到一个图表 (p, q, h)，其中 h 是生成元的张量积。更具体地说，我们首先构造一个结构余跨的同构 $\alpha : h' \to h$，其分量定义如下。

$$\alpha_W = \mathsf{sort}_{k_i}^{-1} \otimes \mathsf{sort}_{k_o}^{-1} \qquad \alpha_{E_i} = \mathsf{sort}_{k_i}^{-1} \qquad \alpha_{E_o} = \mathsf{sort}_{k_o}^{-1} \qquad \alpha_X = \mathsf{id}$$

这里，$\mathsf{sort}f$ 表示由单射 f 排序的唯一置换（参见附录 A），而$k_i = \mathrm{h}G(x_i)$ G(端口$_i$)i：$X \times \mathbb{N}$ 和 $k_o = \mathrm{h}G(x_o)$ G(端口$_o$)i：$X \times \mathbb{N}$ 是“排序键”函数。注意，根据 H'的良构性，k_i 和 k_o 是单射。

的目标 α 是结构余跨 $h = ($类型$_{k_i} \fatsemi \iota_0$，类型$_{k_o} \fatsemi \iota_1$，$H)$，根据推论 C.7，可分解为 $p \fatsemi g \fatsemi q$，其中

- $g = (\iota_0,\ \iota_1,\ H)$
- $p =$ (排序$_{k_i}$,标识,D(H))
- $q =$ (id, 排序$_{k_o}$,D(H))

因此，剩下要证明的是 g 同构于 n 重张量积的生成元。为此，我们采用命题 4.13，由此可立即得出， g 的各条腿已经ι_0 且 ι_1 ，符合要求。因此，只需验证 H 具有式（8）所给出的形式。

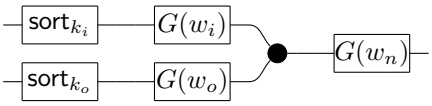
在对象上， H 由下式给出

$$H(W) = G(E_i) + G(E_o) \qquad H(E_i) = G(E_i) \qquad H(E_o) = G(E_o) \qquad H(X) = G(X)$$

而在态射上，根据 α的自然性，我们必须有如下关系。

$$H(w_i) = \iota_0 \qquad H(w_o) = \iota_1 \qquad H(x_i) = \mathsf{sort}_{k_i} \fatsemi G(x_i) \qquad H(x_o) = \mathsf{sort}_{k_o} \fatsemi G(x_o)$$

$$H(\mathsf{port}_i) = \mathsf{sort}_{k_i} \fatsemi G(\mathsf{port}_i) \qquad H(\mathsf{port}_o) = \mathsf{sort}_{k_o} \fatsemi G(\mathsf{port}_o) \qquad H(x_n) = G(x_n)$$

$H(w_n) =$ 

首先注意， $H(w_i) = \iota_0$、 $H(w_o) = \iota_1$以及 $H(x_n) = G(x_n)$均如式（8）所示。对于 H中的生成元，分别将所选的类型赋值记为 $a :\ H(x_n) \to \Sigma_0^*$ 和 $b :\ H(x_n) \to \Sigma_0^*$ 。现在将 H 像中的其余态射视为有限数组。 $H(x_i) = \mathsf{sort}_{k_i} \fatsemi G(x_i) = \mathsf{sort}_{k_i} \fatsemi hG(x_i)\ G(\mathsf{port}_i)\mathrm{i} \fatsemi \pi_0$ 是一个形如 $0^{|a(0)|}1^{|a(1)|}1, \ldots$的非递减数组，因此等于

$$\bigotimes_{i \in G(X)} !_{|a(i)|}$$

如所需。对于 $H(x_o)$，也有类似的等式。同理， $H(\mathsf{port}_i)$ 是“分段递增”的，也就是说，它是数组的串接

$$\langle 0, 1, \ldots, |a(0)| - 1\rangle \ + \ \langle 0, 1, \ldots, |a(1)| - 1\rangle \ + \ \ldots \ + \ \langle 0, 1, \ldots, |a(G(X) - 1)| - 1\rangle$$

这显然符合 (8) 中描述的形式。类似的推理也适用于 H(端口$_o$)。

最后，由于 H' 是良构的，我们有

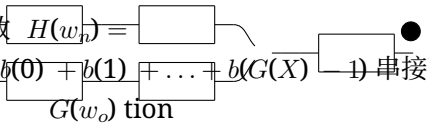
$$\forall e \in E_i \quad w_n(w_i(e)) = a(x_i(e))_{\mathsf{port}_i(e)} \qquad\qquad \forall e \in E_o \quad w_n(w_o(e)) = b(x_o(e))_{\mathsf{port}_o(e)}$$

并且由于 sort$_{k_i}$ 是双射，在 H 中我们有

$$\forall e \in E_i \quad G(w_n)(G(w_i)(\mathsf{sort}_{k_i}(e))) = a(G(x_i)(\mathsf{sort}_{k_i}(e)))_{G(\mathsf{port}_i)(\mathsf{sort}_{k_i}(e))}$$

and

$$\forall e \in E_o \quad G(w_n)(G(w_o)(\mathsf{sort}_{k_o}(e))) = b(G(x_o)(\mathsf{sort}_{k_o}(e)))_{G(\mathsf{port}_o)(\mathsf{sort}_{k_o}(e))}$$

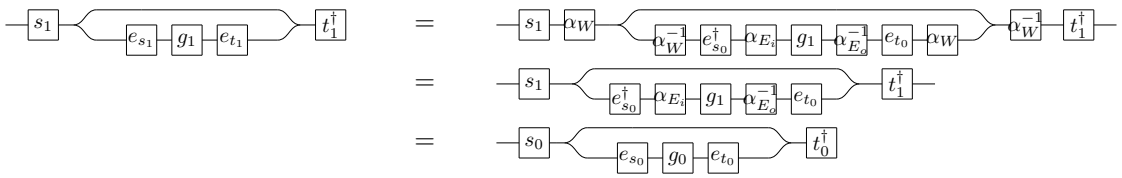
sort$_{k_i}$ $G(w_i)$，因此我们可以将函数 $H(w_n) =$  $G(w_n)$ 视为由有限数组 $a(0) + a(1) + \ldots a(G(X) - 1) + b(0) + b(1) + \ldots + b(G(X) - 1)$ 串接而成；这正如命题 4.13 所要求的那样。concatena- sort$_{k_o}$ $G(w_o)$ tion

□

We 现在可以使用 Frobenius 分解来定义函子 $\mathcal{D} : \mathsf{Diagram}_{\Sigma+\mathbf{Frob}} \to \mathsf{Free}^{\Sigma+\mathbf{Frob}}$.

命题 B.3. $\mathcal{D}$ 是良定义的。

证明。设 $d_i = (s_i, t_i, G_i)$ 是对应于 $i \in \{0, 1\}$ 的图表，且 $\alpha\colon d_0 \to d_1$ 是图表的一个同构。假设 d_0 和 d_1 的“近弗贝尼乌斯”分解（命题 B.1）是一个真正的弗罗贝尼乌斯分解。正如命题 5.5 的证明所表明的，这一假设不失一般性，因为任意图表都同构于一个具有所需形式的图表。因此，我们可以将 D 视为定义在这些代表元上。于是可以如下计算，其中我们用 x 代替 $\mathsf{S}(x)$，以避免过多的记号负担。不过，读者应当清楚，这些图表表示范畴 $\mathsf{Free}^{\Sigma+\mathbf{Frob}}$ 中的态射。



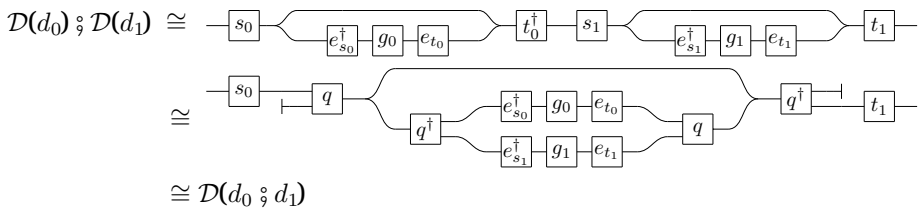
在计算的第一步中，我们使用同构 α 来用 d_0 表示 d_1 的分量。第二步由命题 D.2 得出。最后一步是证明 $\alpha_{E_i} \fatsemi g_1 \fatsemi \alpha_{E_i}^{-1} = g_0$。由于 d_0 和 d_1 同构，且 g_0 和 g_1 只是生成元的张量积，因此必定存在一个置换 π，使得 $g_1 = \pi \fatsemi g_0 \fatsemi \pi^{-1}$ 和 pi 是“分块”的，即对生成元 g_0 的输入类型而言是自然的。因此，只需证明 $\alpha_{E_i}^{-1} \fatsemi \pi = \mathsf{id}$，从而得到 $\alpha_{E_i}^{-1} = \pi$。这由良构性得出。注意，根据自然性，$G_1(x_i) = \alpha_{E_i}^{-1} \fatsemi G_0(x_i) \fatsemi \alpha_X$，我们可以将其改写为 $G_1(x_i) = \alpha_{E_i}^{-1} \fatsemi \pi \fatsemi G_0(x_i)$。由于 $G_1(x_i)$ 和 $G_0(x_i)$ 都是单调的，因此 $\alpha_{E_i} \fatsemi \pi = (p_0 \otimes p_1 \otimes \ldots \otimes p_X)$ 必定是置换的张量积。也就是说，每个 p_i 都会改变单个生成元的输入导线顺序。现在我们证明每个 $p_i = \mathsf{id}$。根据自然性，可以写为 $G_1(\mathsf{port}_i) = \alpha_{E_i}^{-1} \fatsemi \pi \fatsemi \pi^{-1} \fatsemi G_0(\mathsf{port}_i)$。由于已知 $\alpha_{E_i}^{-1} \fatsemi \pi$ 是置换的张量积，且 π 是分块的，因此必有 $G_1(\mathsf{port}_i) = (p_0 \fatsemi \iota_0) + \ldots + (p_X \fatsemi \iota_0)$。最后，我们可以利用 $(f + g) = (\mathsf{id} + h) \implies f = \mathsf{id}$，得出每个 $p_i = \mathsf{id}$，从而得到 $\alpha_{E_i} = \pi^{-1}$。

通过类似的推理可以验证，α_{E_o} 对应于生成元输出的分块置换，因此 $g_1 = \alpha_{E_i}^{-1} \fatsemi g_0 \fatsemi \alpha_{E_i}$，证明完毕。 □

命题 B.4。$\mathcal{D}$ 是一个函子。

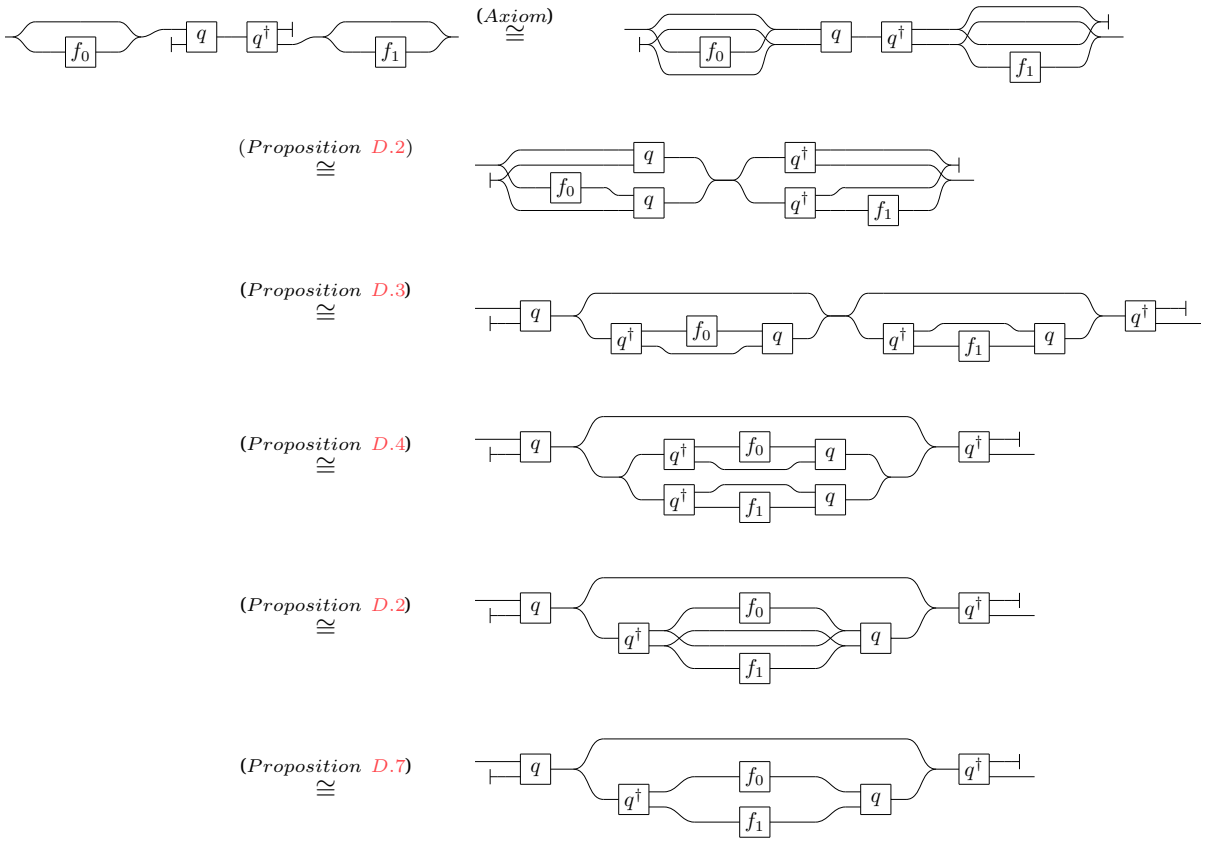
证明。$\mathcal{D}(\mathsf{id}) = \mathsf{id}$ 根据定义成立，因此只需证明 $\mathcal{D}(f \fatsemi g) = \mathcal{D}(f) \fatsemi \mathcal{D}(g)$。

设 $d_i = (s_i, t_i, G_i)$ 是 $i \in \{0, 1\}$ 的图表，且 $(s_i, e_{s_i}, g_i, e_{t_i}, t_i)$ 是命题 5.5 中指定的 d_i 的 Frobenius 分解。若 D 是超图函子，就必须保持 Frobenius 蜘蛛；此外，我们会将生成元的张量积 g_0 和 g_1 映射为 $\mathsf{Free}^{\Sigma+\mathbf{Frob}}$ 中相应的张量积。因此，我们必须证明以下等式在 $\mathsf{Free}^{\Sigma+\mathbf{Frob}}$ 中成立，其中 $q = \mathsf{c}(t_0 \fatsemi \iota_0, s_1 \fatsemi \iota_1)$。



根据 [13, 定理 1.2]，我们有 $q \quad q^\dagger \quad \cong$ 此外，由于 q 是一个余等化子，它是一个满射函数，这意味着其在 $\mathsf{Free}^{\Sigma+\mathbf{Frob}}$ 中的对应态射由生成元 、 、 和 通过张量积与复合构造而成。因此我们

证明任意箭头 f_0 和 f_1 都满足以下引理。



从而完成证明。 □

命题 B.5。 D 是一个严格幺半函子

证明。 我们必须证明 $\mathcal{D}(c_0 \otimes c_1) = \mathcal{D}(c_0) \otimes \mathcal{D}(c_1)$。这可仿照命题 5.4证明中的归纳步骤得出。 □

C Frobenius 蜘蛛作为余跨度

现在我们可以证明表示 Frobenius 蜘蛛的图表与顶点位于 L 的像中的图表之间的对应关系。请注意，该证明依赖于命题 C.5，该命题见附录C.1。

命题 4.7 的证明。我们首先证明，任何形如 $d = (s, t, \mathsf{L}(Y))$ 的图表都是一个 Frob 蜘蛛。我们有 $d = (s, \mathsf{id}, \mathsf{L}(Y)) \fatsemi (\mathsf{id}\ t, \mathsf{L}(Y)) = \mathsf{S}(s) \fatsemi \mathsf{S}(t)^\dagger$，由命题 C.5 得出。接着根据定义 3.5，$\mathsf{S}(s)$ 和 $\mathsf{S}(t)$ 是由生成元 , , , 通过张量积和复合构造而成的。

现在我们验证，每个 Frobenius 蜘蛛都是形如 $(s, t, \mathsf{L}(Y))$ 的图表。特殊 Frobenius 幺半群的生成元是 [1, 定理 3.12]，中给出的结构余跨。由此可直接得出，每个生成元的顶点都具有 $\mathsf{L}(Y)$ 的形式。张量积的顶点也具有这种形式，因为 L 保持余极限，因此只需检查复合 $(s_0, t_0, \mathsf{L}(Y_0)) \fatsemi (s_1, t_1, \mathsf{L}(Y_1))$ 的顶点是否也在 L 的像中。复合通过余等化子计算。令 $q = \mathsf{c}(\mathsf{L}(t_0) \fatsemi \iota_0, \mathsf{L}(s_1) \fatsemi \iota_1)$。由于 ACSets 的余极限是逐点计算的，因此 $q = \mathsf{c}(\mathsf{L}(t_0 \fatsemi \iota_0), \mathsf{L}(s_1 \fatsemi \iota_1))$。同理，我们有 $q_W = \mathsf{c}(t_0 \fatsemi \iota_0, s_1 \fatsemi \iota_1)$，对于分量 $Z \neq W$，有 $\mathsf{L}(x)_Z = ?: 0 \to 0$，因此 $q_Z = ?: 0 \to 0$。于是可知 $q = \mathsf{L}(\mathsf{c}(t_0 \fatsemi \iota_0, s_1 \fatsemi \iota_1))$，证明完毕。 □

C.1 余跨度的伴随

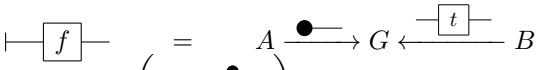
现在我们证明，图表的伴随会交换余跨的“腿”，因此 $(s,\ t,\ G)^\dagger = (t,\ s,\ G)$。我们在命题 C.5 中正式陈述这一点。为此需要若干引理，我们先证明这些引理。

请注意，本节中的证明是关于具有有限余极限和余等化子的范畴中的余跨度的一般性陈述（因此也具有所有有限余极限）。

命题 C.1. $\mathsf{c}(f,\ f) = \mathsf{id}$

证明。 显然，$f \fatsemi \mathsf{id} = f \fatsemi \mathsf{id}$，且所有 g 都构成余叉 $f \fatsemi g$，其泛态射为 $u = g$。 □

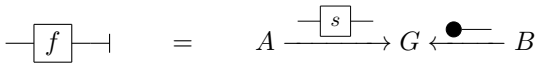
命题 C.2。 设 $f = A \xrightarrow{s} G \xleftarrow{t} B$ 是一个余跨度。那么存在一种余等化子的选取，使得



证明。 首先，注意到 是余等化子 $\mathsf{c}\left(\ ,\ \right)$。由余单位公理可立即得出，这使得余叉交换；接着，给定某个同样使余叉交换的 q，可以验证 $u =$  是泛态射。

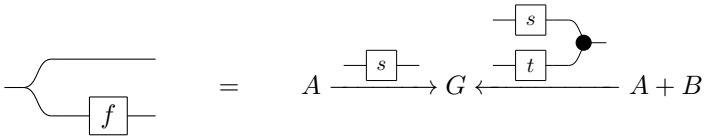
 □

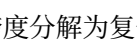
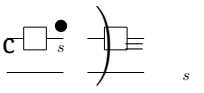
命题 C.3. 设 $f = A \xrightarrow{s} G \xleftarrow{t} B$ 是一个余跨度。那么存在一种余等化子的选取，使得



证明. 如命题C.2中所示。 □

命题 C.4. 设 $f = A \xrightarrow{s} G \xleftarrow{t} B$ 是一个余跨度。那么存在一种余等化子的选取，使得

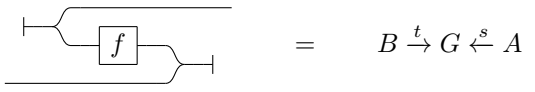


证明. 将该余跨度分解为复合  现在选取  ，根据单位公理，它是交换的。接着，给定某个同样使余叉交换的 q，可以验证存在唯一的 $u =$  。然后再次根据单位公理即可得出结论。

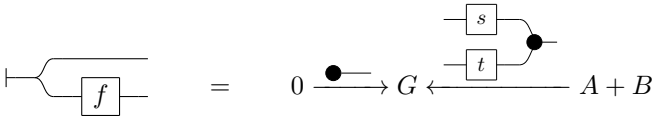
 □

最后，我们可以证明主要命题。

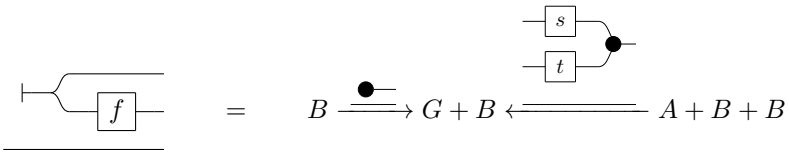
Pro命题 C.5. 设 $f = A \xrightarrow{s} G \xleftarrow{t} B$ 是一个余跨度。那么存在一种余等化子的选取，使得 at



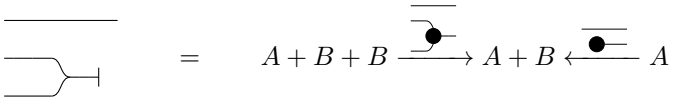
证明。 根据命题C.2和 C.4我们有



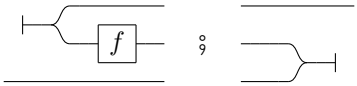
因此，根据张量积的定义，我们得到如下结果。



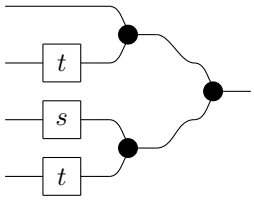
类似地，我们有



因此，只需为下列复合找到一个余等化子。



这样的余等化子由以下方式给出。



然后该结果由单位公理得出。 □

C.2 Frobenius 蜘蛛与置换

腿为复合置换的图表可以分解为与蜘蛛的复合。我们通过以下两个命题对其作形式化陈述。

命题 C.6. 设 $c_0 := (s_0, t_0, \mathsf{L}(B)) \colon A \to B$ 和 $c_1 := (s_1, t_1, G) \colon B \to C$ 是图表，其中 t_0 是一个同构。那么存在一种余等化子的选择，使得

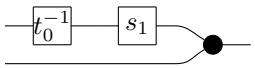
$$c_0 \fatsemi c_1 = (s_0 \fatsemi t_0^{-1} \fatsemi s_1 \ , \ t_1 \ , \ G)$$

类似地，若 $c_2 := (s_2, t_2, \mathsf{L}(C)) \colon C \to D$ 且 s_2 是一个同构，则

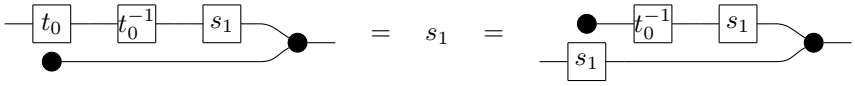
$$c_1 \fatsemi c_2 = (s_1 \ , \ t_2 \fatsemi s_2^{-1} \fatsemi t_1 \ , \ G)$$

证明。 我们证明第一种情况，即 t_0 是一个同构的情形；第二种情况可用对称论证证明，此处略去。首先，我们证明相关的余等化子具有以下闭式。

$$\mathsf{c}(t_0 \fatsemi \iota_0, s_1 \fatsemi \iota_1) =$$



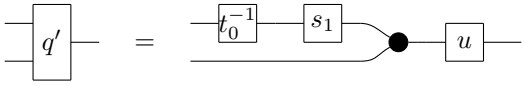
注意到余叉是交换的：



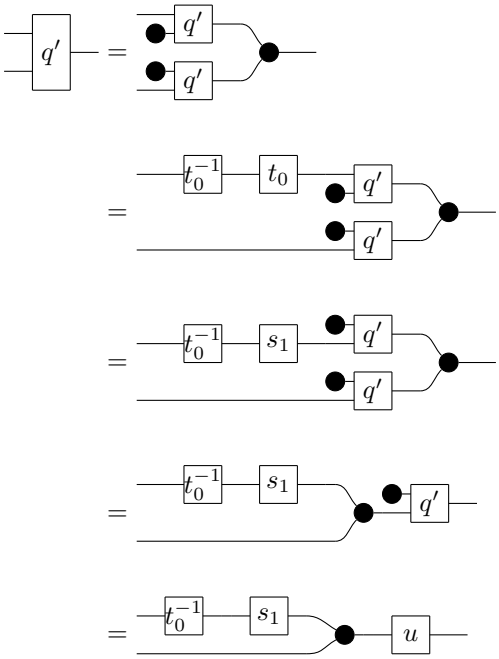
现在假设存在某个 q' 使得



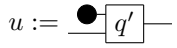
存在一个 u，使得



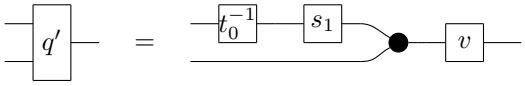
我们可以按如下方式计算：



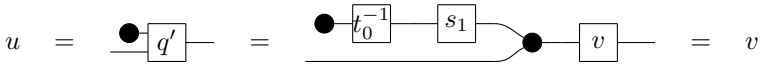
因此我们有



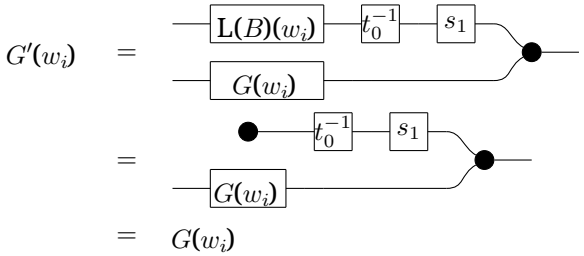
而且这个 u 是唯一的。假设存在某个 v 满足



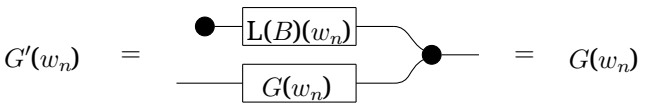
那么我们可以推导出



最后，我们还必须证明复合的顶点等于 G。根据命题 4.16，余等化子 $q : \mathsf{L}(B) + G \to G'$ 是一个 ACSet 态射，其分量除$q_W = \mathsf{c}(t_0 \fatsemi \iota_0, s_1 \fatsemi \iota_1)$外均为恒等态射。在对象上，我们有 $G'(Y) = G(Y)$，这可以分两种情况验证。第一，$(\mathsf{L}(B) + G)(Y) = G(Y) = G'(Y)$，其中对象为 $Y \neq W$，此时 q_Y是恒等态射。第二，注意到 q_W 的陪域是 $G(W)$，因此 $G'(W) = G(W)$。对于态射，只需检查源或目标为 $G'(W)$的态射，因为根据自然性，其余所有态射都必然立即相等。我们先从 $G'(w_i)$开始。根据自然性，必有$G'(w_i) = (\mathsf{L}(B)(w_i) \otimes G(w_i)) \fatsemi q$；又由于 $\mathsf{L}(B) = 0$，我们还有 $\mathsf{L}(B)(w_i) = \mathsf{L}(B)(w_o) =$ 。于是可以计算



同样地，我们有 $G'(w_o) = G(w_o)$。最后，我们利用 $G'(w_n)$ 是余等化子的典范态射，计算得



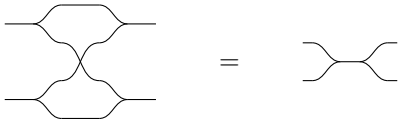
因此 $G' = G$。 □

推论 *C.7.* 设 $c := L(A)\, L(\xrightarrow{p \fatsemi s_0 \fatsemi \epsilon_G})\, G)\, L(\xleftarrow{q \fatsemi t_0 \fatsemi \epsilon_G})\, L(B)$ 是一个图表。那么 $c = \hat{}p \fatsemi c \fatsemi \hat{}q$ 其中 $\hat{}p = (p, id, L(A))$, $\hat{}q = (id\ q, L(B))$, 和 $c = (s_0, t_0, G)$。

D Frobenius 蜘蛛引理

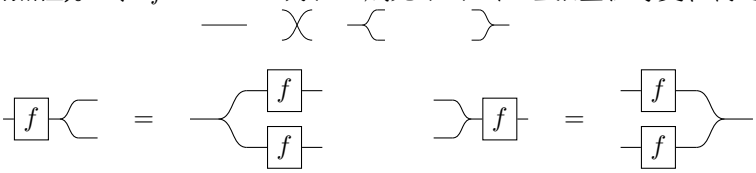
以下引理适用于任意超图范畴。我们特别使用它们来证明$\mathcal{D}$的函子性。

命题 D.1(双代数)。



证明。 由 [13, 定理 1。2]可直接得出。也可以直接利用结合律、交换律和蛇形公理推导出这个等式。 □

命题 D.2 (部分自然性)。 令 $f : A \to B$ 为由生成元 、 、 和 经张量积与复合构造的态射。那么



(18)

证明。 我们用归纳法证明第一个等式，第二种情况可对称地推出。令 f 为由给定生成元经张量积与复合构造的态射。首先检查归纳法的基例。

基例 $f =$id 显然: $=$



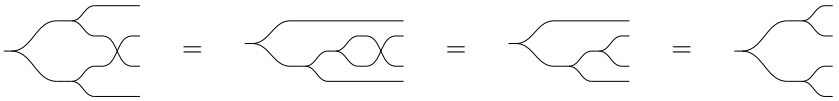
基例 $f =$



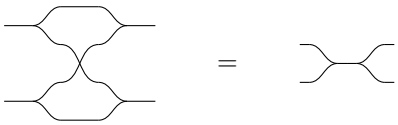


基例 $f =$

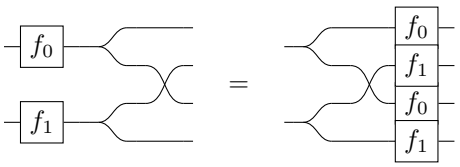




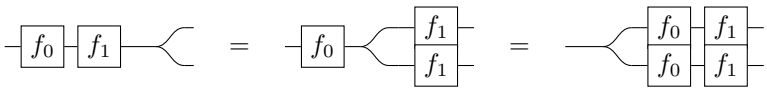
基例 $f =$ 由命题 D.1 可直接得出,



归纳步骤 现在假设 f_0 和 f_1 满足（18）。那么它们的张量积也满足该条件 :

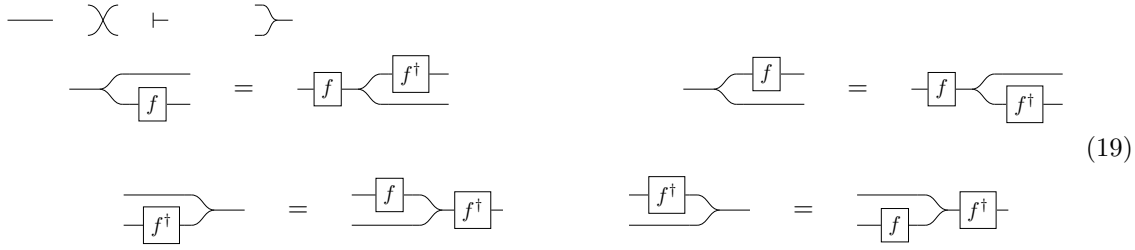


它们的复合也如此：



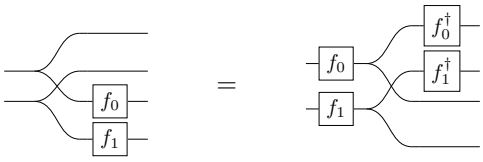
□

命题 D.3. 设 f 是由生成元 , , , 和 通过张量积和复合构成的态射。那么我们有以下等式。

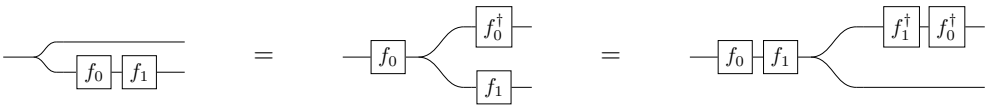


(19)

证明。 我们首先用归纳法证明（19）中的左上方等式。与命题 D.2中一样，可以直接推出基例，也可以通过 [13, 定理 1。2]观察到它们成立。接下来归纳步骤如下。假设 f_0 和 f_1 满足（19）中的左上方等式。那么



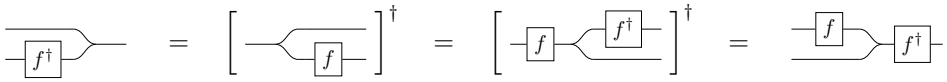
and



如所需。现在我们可以证明其余方程也成立。右上方的方程根据交换律成立。

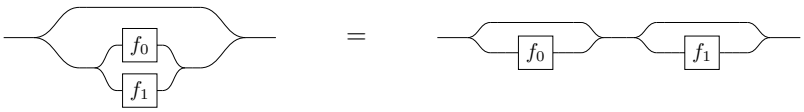


左下方的等式只是左上方等式的伴随：



and 右下方的等式依据交换律成立，理由与右上方等式的证明相同。 □

命题 D.4。



证明。 由 [13,定理 1.2]可得。 □

命题 D.5。 设 $q = \mathsf{S}(f)$ 为某个满射 f，使得 q 由生成元 、 、 通过张量积和复合构造而成。则 $q^\dagger \fatsemi q = \mathsf{id}$。

证明。 归纳法。在基例中，容易看出当 $q =$ 或$q =$ 时，该等式成立。此外，$=$ 作为公理成立。因此，剩下的任务是证明归纳步骤。设 q_0 和 q_1 为态射，使得对于 $i \in \{0,1\}$，$q_i^\dagger \fatsemi q_i = \mathsf{id}$。则该等式对它们的张量积成立

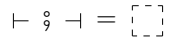


以及它们的复合。



□

请注意，在 额外的 特殊 Frobenius 幺半群理论中，命题 D.5 适用于 任意属于 S 像中的态射；满射性要求被取消了。这是因为存在额外的公理



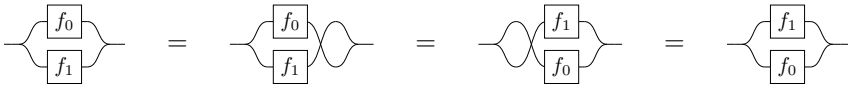
由此证明了剩余的基例。

命题 D.6。



(20)

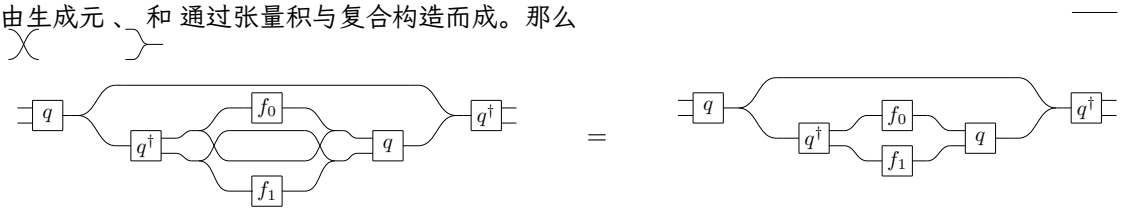
证明。



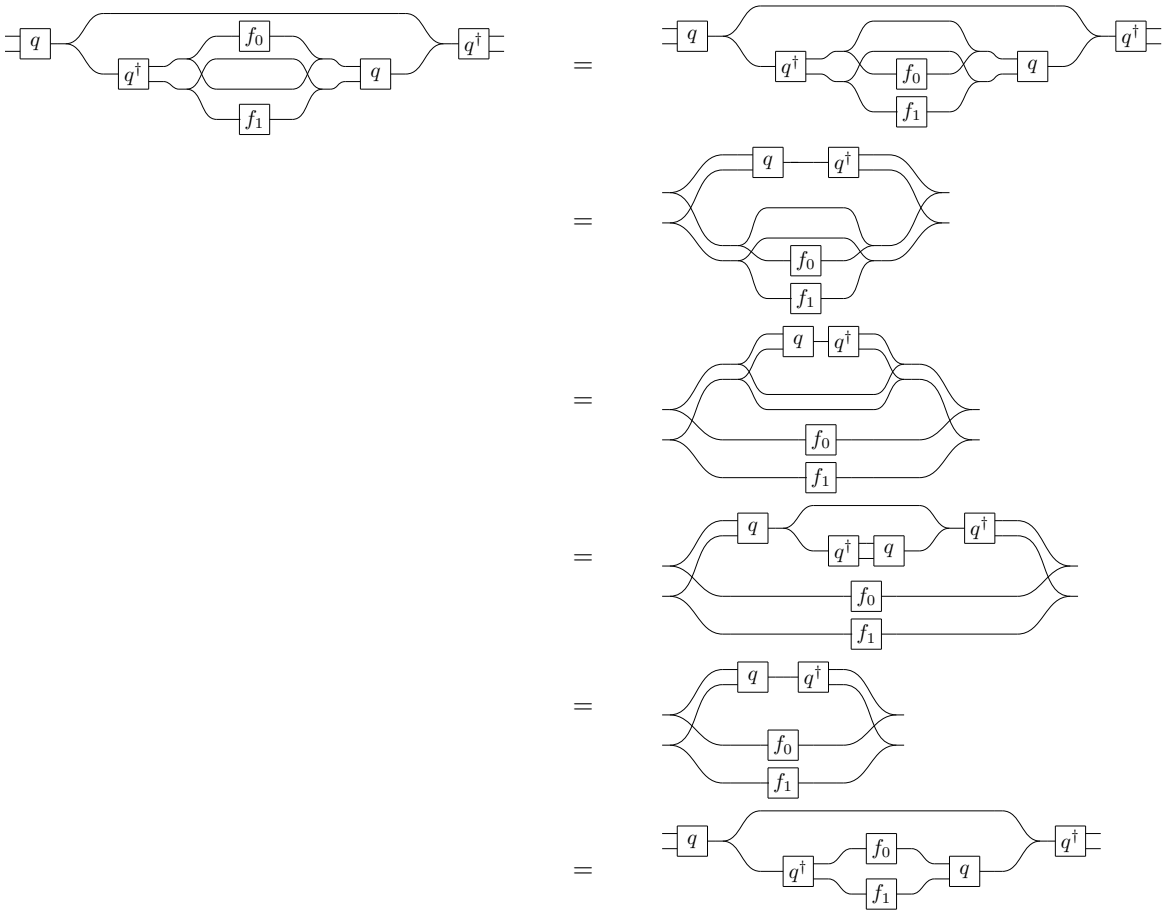
□

最后一个引理是证明$\mathcal{D}$ 具有函子性的关键步骤。不过，其推导颇为繁琐，因此我们在证明的主体部分省略了它。

命题 D.7。 设 f_0 和 f_1 是任意态射，并假设 $q = \mathsf{S}(q')$ 对某个满射 q'成立。也就是说，假设 q 由生成元 、 和 通过张量积与复合构造而成。那么



证明。 我们如下推导。



上述内容中，第一步使用命题 D.6，倒数第二步使用命题 D.5，其余所有步骤使用命题 D.3。

□