Contents

1 Monoidal Category		noidal Category	1
	1.1	Symmetric Monoidal Category	1
	1.2	Symmetric Monoidal Closed Category	2
2	2.1	nodial Functor Symmetric Monoidal Functor	
3 Monodial Adjuctions		6	
4	Mo	nodial Natural Transformations	6

1 Monoidal Category

1.1 Symmetric Monoidal Category

Definition 1. A symmetric monoidal catergory (SMC), $(\mathbb{C}, \bullet, 1, \alpha, \lambda, \rho, \gamma)$, is a category \mathbb{C} equippped with a bifunctor $\bullet : \mathbb{C} \times \mathbb{C} \to \mathbb{C}$ with a neutral element 1 and natural isomorphisms α, λ, ρ , and γ :

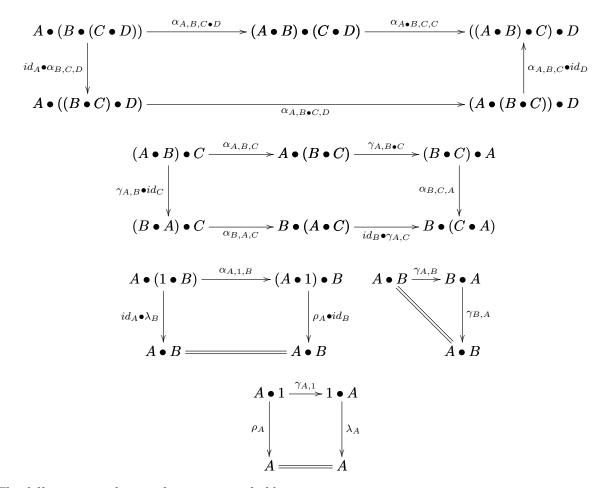
1.
$$\alpha_{A,B,C}: A \bullet (B \bullet C) \xrightarrow{\sim} (A \bullet B) \bullet C$$

2.
$$\lambda_A: 1 \bullet A \xrightarrow{\sim} A$$

3.
$$\rho_A: A \bullet 1 \xrightarrow{\sim} A$$

4.
$$\gamma_{A,B}: A \bullet B \xrightarrow{\sim} B \bullet A$$

which make the following 'coherence' diagrams commute.



The following equality is also require to hold:

$$\lambda_1 = \rho_1 : 1 \bullet 1 \to 1$$

1.2 Symmetric Monoidal Closed Category

Definition 2. A symmetric monoidal closed category (SMCC), $(\mathbb{C}, \bullet, \multimap, 1, \alpha, \lambda, \rho, \gamma)$, is a SMC such that for all objects A in \mathbb{C} , the functor $-\otimes A$ has a specified right adjoint $A \multimap -$.

Let \mathbb{C} be a SMC $(\mathbb{C}, \bullet, 1, \alpha, \lambda, \rho, \gamma)$. A structure, M, in \mathbb{C} for a given signature S_g is specified by giving an object $\llbracket \sigma \rrbracket$ in \mathbb{C} for each type σ , and a morphism $\llbracket f \rrbracket : \llbracket \sigma_1 \rrbracket \bullet ... \bullet \llbracket \sigma_n \rrbracket \to \llbracket \tau \rrbracket$ in \mathbb{C} for each function symbol $f: \sigma_1, ..., \sigma_n \to \tau$. In the case where n = 0 then the structure assigns a morphism $\llbracket c \rrbracket : 1 \to \llbracket \tau \rrbracket$ to a constant $c: \tau$.

Given a context $\Gamma = [x_1 : \sigma_1, ..., x_n : \sigma_n]$ we define $\llbracket \Gamma \rrbracket$ to be the product $\llbracket \sigma_1 \rrbracket \bullet ... \bullet \llbracket \sigma_n \rrbracket$. We represent the empty context with the neutral element 1. We need to define the bracketing convention. It shall be assumed that the tensor product is left associative, i.e. $A_1 \bullet A_2 \bullet ... \bullet A_n$ will be taken to mean $(...(A_1 \bullet A_2) \bullet ...) \bullet A_n$. We find it useful to define two 'book-keeping' functions,

$$Split(\Gamma, \Delta) : \llbracket \Gamma, \Delta \rrbracket \to \llbracket \Gamma \rrbracket \bullet \llbracket \Delta \rrbracket$$

$$Split(\Gamma, \Delta) \stackrel{\text{def}}{=} \begin{cases} \lambda_{\Delta}^{-1} & \text{If Γ empty} \\ \rho_{\Gamma}^{-1} & \text{If Δ empty} \\ id_{\Gamma \bullet A} & \text{If $\Delta = A$} \\ Split(\Gamma, \Delta') \bullet id_A; \alpha_{\Gamma, \Delta', A}^{-1} & \text{If $\Delta = \Delta'$, A} \end{cases}$$

$$Join(\Gamma,\Delta): \llbracket \Gamma,\Delta \rrbracket \to \llbracket \Gamma \rrbracket \bullet \llbracket \Delta \rrbracket$$

$$Join(\Gamma, \Delta) \stackrel{\text{def}}{=} \begin{cases} \lambda_{\Delta} & \textit{If Γ empty} \\ \rho_{\Gamma} & \textit{If Δ empty} \\ id_{\Gamma \bullet A} & \textit{If $\Delta = A$} \\ \alpha_{\Gamma, \Delta', A}; Join(\Gamma, \Delta') \bullet id_{A} & \textit{If $\Delta = \Delta', A$} \end{cases}$$

We shall also refer to indexed variants of these; for example

$$Split_n(\Gamma_1,...,\Gamma_n): \llbracket \Gamma_1,...,\Gamma_n \rrbracket \to \llbracket \Gamma_1 \rrbracket \bullet ... \bullet \llbracket \Gamma_n \rrbracket$$

which is defined in the obvious way.

The semantics of a term in context is then specified by a structural induction on the term.

$$[\![x:\sigma\rhd x:\sigma]\!]\ \stackrel{\mathrm{def}}{=}\ id_\sigma$$

$$\llbracket \Gamma_1,...,\Gamma_n \rhd f(M_1,...,M_n) : \tau \rrbracket \ \stackrel{\mathsf{def}}{=} \ Split_n(\Gamma_1,...,\Gamma_n); \llbracket \Gamma_1 \rhd M_1 : \sigma_1 \rrbracket \bullet ... \bullet \llbracket \Gamma_n \rhd M_1 : \sigma_n 0 \rrbracket; \llbracket f \rrbracket$$

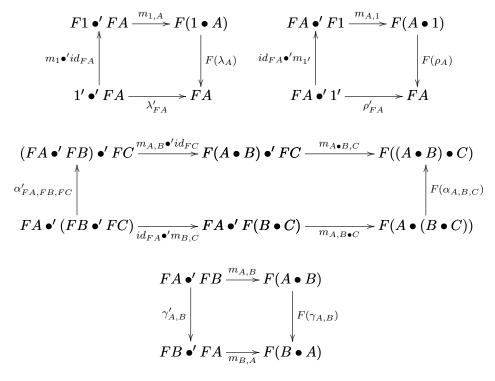
2 Monodial Functor

2.1 Symmetric Monoidal Functor

Definition 3. A symmetric monoidal functor between SMCs $(\mathbb{C}, \bullet, 1, \alpha, \lambda, \rho, \gamma)$ and $(\mathbb{C}', \bullet', 1', \alpha', \lambda', \rho', \gamma')$ is a functor $F : \mathbb{C} \to \mathbb{C}'$ equipped with

- 1. A morphism $m_{1'}: 1' \to F1$.
- 2. For any two objects A and B in \mathbb{C} , a natural transformation $m_{A,B}: F(A) \bullet' F(B) \to F(A \bullet B)$

These must satisfy the following diagrams:



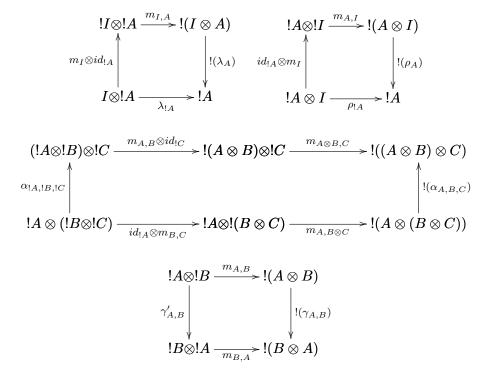
However in this particular case, assuming that ! is a symmetric monoidal (endo) functor means that ! comes equipped with a natural transformation

$$m_{A,B}: !A \otimes !B \rightarrow !(A \otimes B)$$

 $and\ a\ morphism$

$$m_I:I\to !I$$

(where m_I is just the nullary version of the natural transformation.) The diagrams given in the above definition become the following:

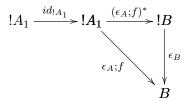


2.2 Symmetric Monoidal Functor (Strict and Strong)

Definition 4. A symmetric monoidal functor, $(F, m_{A,B}, m_{1'}) : \mathbb{C} \to \mathbb{C}'$, is said to be

- 1. Strict if $m_{A,B}$ and $m_{1'}$ are identities.
- 2. Strong if $m_{A,B}$ and $m_{1'}$ are natural isomorphisms.

The equation in context for Dereliction gives us that



commutes, or in other words

$$\begin{array}{c|c} !A & \xrightarrow{!f} & !B \\ & \downarrow & & \downarrow \\ & \downarrow & & \downarrow \\ A & \xrightarrow{f} & B \end{array}$$

commutes. Given that we have made the assumption that ! is a (symmetric) monoidal functor, this diagram suggests that ϵ is a monoidal natural transformation. We shall make this assumption and write ϵ for the monoidal natural transformation $\epsilon : ! \to Id$.

- 3 Monodial Adjuctions
- 4 Monodial Natural Transformations