モジュールのパターンマッチ・ Focused Logic

El Pin Al

本稿は言語実装 Advent Calendar 2020 13 日目の記事です。

シークエント計算に対する focusing の概念を用いることで、プログラミング言語におけるパターンマッチを論理学的に — そして Curry-Howard 対応により型理論的に — 形式化することができます。本稿では、この考え方を ML のモジュールシステムに適用することで、モジュールのパターンマッチという概念を提案します。

1. モジュールのパターンマッチ

1.1. ワイルドカード

モジュールの評価は副作用を伴なうことがあります。たとえば、ファンクタ `F` を適用したときの副作用のみが重要で、その結果を用いない場合、次のように `_` でワイルドカードを利用できると便利です。

```
module _ = F X
```

もちろん、`F` がストラクチャを返すファンクタであった場合は次のようにすることで、同じような結果が得られます。(本稿では、`sig ... end` と `struct ... end` の両方を `{ ... }` と書きます。)

```
include F X : {}
```

しかし、ワイルドカードを用いた方法は、

- コア言語に対する普通のパターンマッチ (`val _ = print "abc"`) との類推が効く
- `F` がファンクタを返すファンクタであった場合でも使える

という点で良いです。

また、高階ファンクタ (`H`) に渡すファンクタ (`G`) を定義する際、もし `G` が引数を利用しない場合、ここでもワイルドカードがあると便利です。

```
module G (_ : S) = M
module N = H G
```

1.2. モジュールの分解

モジュールのコンポーネントの内、数個しか利用しない場合、それらのコンポーネントをパターンマッチで取り出せると便利です：

```
module {type t val x} = F X
```

これは意味的には次のコードと等価です。

```
local
  module M = F X
in
  type t = M.t
  val x = M.x
end
```

すなわち、ファンクタ F を X に適用した結果から、型コンポーネント t と値コンポーネント x を取り出します。これは、次のように include と透過的シグネチャ指定 (transparent signature ascription) の組合せで表現できます。(x が型 int -> t を持つと仮定します。)

```
include F X : {type t val x : int -> t}
```

しかし、この方法だと t がこのファンクタ適用で生成された「新しい抽象型」である場合、x だけを取り出すことができません[*1]。

```
include F X : {val x : (* ここに何を書いたらいいのか分からない *)}
```

一方、パターンマッチを使った方法ではそれが可能です。

```
module {val x} = F X
```

これは x の型を書かずに済むからです。

また、パターンはネストできると良いでしょう：

---

1 ここで、F が generative functor であると仮定しています。Applicative functor であった場合は {val x : int -> (F X).t} と書けます。

```
module {module M = {type t}} = F X
```

省略記法もあると便利そうです：

```
module {type M.t} = F X
```

2. Focusing によるパターンマッチの形式化

さて、モジュールレベルのパターンマッチは理論的にどう扱えばよいのでしょうか。1 つの方法は、パターンマッチを単なる糖衣構文として扱うことです。`module _ = ...` はフレッシュな識別子 `M` を用いて `module M = ...` の糖衣構文とし、`module {type t val x} = ...` もフレッシュな識別子 `M` を用いて `module M = ... type t = M.t val x = M.x` の糖衣構文とする方法です。確かにこの方法は上手く機能します。しかしながら、本稿では別の、より型理論的なアプローチを取ります。特定の言語機能を型理論的に解析する利点は、往々にしてその一般化が自然に現れることです。たとえば Harper, Mitchell & Moggi は、ML モジュールをファンクタも含めて型理論的に扱うことで、Standard ML の 1 階ファンクタを高階ファンクタへと一般化することができました [4, 1]。

今回重要となるのは**シークエント計算**と、それに対する **focusing** です。私はシークエント計算も focusing も大して詳しくないので大雑把な説明をしますが、シークエント計算は「各 connective に対して left rule と right rule がある」演繹体系です。Right rule は自然演繹の導入規則に、left rule は自然演繹の除去規則に対応します。シークエント計算で特徴的なものは left rule です。たとえば、直感主義における論理積 $A \wedge B$ の left rule は次のように表されます。

$$\frac{\Gamma, A, B \vdash C}{\Gamma, A \wedge B \vdash C}$$

シークエント計算に対する focusing [2, 10] は証明探索 (proof search) のために発明されました。Focusing は「シークエント計算の一部の規則は invertible — つまり結論が前提のすべてを含意する — であり、invertible な規則は任意のタイミングで適用してもよい」という洞察に基づいています。つまり、まず invertible な規則を適用し続け、適用できなくなったら、特定の命題に **focus** を取り、non-invertible な規則を適用します。たとえば、あるシークエント $A \to B \vdash C \wedge D$ の直感主義における導出木をボトムアップに構築したいとき、まず最初に含意の left rule を適用するか、論理積の right rule を適用するか少なくと

も 2 通りあります。しかし、focusing によると最初に適用するのは、invertible である「論理積の right rule」しかありません。このように、導出の方法に制限を掛けることで探索空間を大幅に削減することができます。

各 connective は、negative あるいは positive といった **polarity** [3] を持ちます。Negative な connective は right rule が invertible で、left rule が non-invertible です。Positive な connective はその逆です。線形論理においては、negative connective と positive connective は綺麗な対象性を持って現れますが、直感主義論理においては、いくつかの connective は negative とも positive とも捉えられます。たとえば、乗法的論理積 $A \otimes B$ は positive、加法的論理積 $A \,\&\, B$ は negative ですが、$A \wedge B$ は positive かつ negative です。

Robert Harper の Computational trinitarianism は型理論・論理学・圏論において、1 つの分野で生じた概念は別の 2 分野においても意味を持つべきである、と主張しています。圏論は本稿では扱わないので置いておくとして、型理論と論理学の関係性を見てみましょう。ラムダ計算は論理学、特に自然演繹と深い関わりがあります。そして Standard ML のような関数型言語の多くはラムダ計算を元にしています。一方、関数型言語などにおいて特に便利で重要な機能であるパターンマッチは、自然演繹に対応する概念がありません。パターンマッチはシークエント計算に対する focusing から生じます [8, 9]。

ML のモジュールシステムに focusing の概念を最初に持ち込んだのは、Crary の今年の論文 [12] です。(この論文は非常に面白いのでぜひ読んでほしいです。ちなみに私はこの論文で focusing を知りました。)

2.1. 構文・型システム

本稿で扱う言語は [12] に基づいています。そして [12] 自体は [6, 7] の流れを汲んでいます。

まず、カインドと型コンストラクタの文法を与えます。カインド Ω は「項を分類する型コンストラクタ」に付くカインドです。カインド Ω を持つ型コンストラクタを単に型と呼びます。型コンストラクタ $\star$ はカインド 1 を持ちます。Singleton kind $\mathrm{S}(c)$ は「型 c と等しいことを証明できる型」を分類します。また、α, β, γ で型変数を表します。

$$
\begin{aligned}
k &::= \Omega \mid \Pi\alpha{:}k.k \mid \Sigma\alpha{:}k.k \mid 1 \mid \mathrm{S}(c) \\
c &::= \star \mid \alpha \mid \lambda\alpha{:}k.c \mid c\ c \mid \langle c, c\rangle \mid \pi_1 c \mid \pi_2 c \mid c \to c
\end{aligned}
$$

次にシグネチャの文法を与えます。

$$\sigma ::= 1 \mid (\!|k|\!) \mid \langle\!|c|\!\rangle \mid \Pi\alpha{:}\sigma.\sigma \mid \Sigma\alpha{:}\sigma.\sigma$$

依存和 $\Sigma\alpha{:}\sigma_1.\sigma_2$ において、型変数 α が σ_2 に自由に出現しない場合 $\sigma_1 \times \sigma_2$ と書くことがあります。

シグネチャから静的部分をカインドとして取り出す $\mathrm{Fst}(\sigma)$ を定義します。

$$\mathrm{Fst}(1) \equiv 1 \qquad \mathrm{Fst}\big((\!|k|\!)\big) \equiv k \qquad \mathrm{Fst}\big(\langle\!|c|\!\rangle\big) \equiv 1$$

$$\mathrm{Fst}\big(\Pi\alpha{:}\sigma_1.\sigma_2\big) \equiv \Pi\alpha{:}\mathrm{Fst}\big(\sigma_1\big).\mathrm{Fst}\big(\sigma_2\big) \qquad \mathrm{Fst}\big(\Sigma\alpha{:}\sigma_1.\sigma_2\big) \equiv \Sigma\alpha{:}\mathrm{Fst}\big(\sigma_1\big).\mathrm{Fst}\big(\sigma_2\big)$$

項、モジュール、モジュールのパターンの文法を与えます。

$$\begin{aligned}
e &::= x \mid \lambda x{:}c.e \mid e\ e \mid \mathrm{Ext}M \\
p &::= m \mid \top \mid p \wedge p \mid \bigstar \mid \langle p, p \rangle \\
M &::= \bigstar \mid (\!|c|\!) \mid \langle\!|e|\!\rangle \mid m \mid \lambda\alpha/p{:}\sigma.M \mid M\ M \mid \langle M, M \rangle \mid \pi_1 M \mid \pi_2 M \\
&\quad \mid \mathrm{let}\ \alpha/p = M\ \mathrm{in}\ M
\end{aligned}$$

$\top$ はワイルドカードパターンを表します。And パターン $p_1 \wedge p_2$ は p_1 と p_2 の両方にマッチする場合のみマッチします。モジュール $\bigstar$ はシグネチャ 1 を持ち、何もコンポーネントを持たないモジュールを表します。ファンクタ $\lambda\alpha/p{:}\sigma.M$ において、型変数 α と、パターン p 内の各モジュール変数のスコープは M になります。同様に $\mathrm{let}\ \alpha/p = M_1\ \mathrm{in}\ M_2$ において、型変数 α と、パターン p 内の各モジュール変数のスコープは M_2 になります。

文脈は次の文法で表されます。

$$\Gamma ::= \varepsilon \mid \Gamma, \alpha{:}k \mid \Gamma, x{:}c \mid \Gamma, m@c{:}\sigma$$

ここで、$m@c{:}\sigma$ はモジュール m がシグネチャ σ を持ち、m の静的部分は c であることを意味します。また、文脈において、型変数・モジュール変数は重複して束縛することはないと仮定します。

本稿では α 同値なオブジェクトを構文的に同一視します。また、$A\big[c/\alpha\big]$ と書いて、A 内の自由な α を c で (変数を捕縛しないように) 置換したものを表します。$A\big[M/m\big]$ と $A\big[e/x\big]$ も同様です。

Left inversion judgement $p@c{:}\sigma \Rightarrow \Gamma$ を与えます。

$$\frac{}{m@c{:}\sigma \Rightarrow m@c{:}\sigma}\,\mathrm{HypL}$$

$$\frac{}{\top@c{:}\sigma \Rightarrow \varepsilon}\,\mathrm{Weak} \qquad \frac{p_1@c{:}\sigma \Rightarrow \Gamma_1 \qquad p_2@c{:}\sigma \Rightarrow \Gamma_2}{p_1 \wedge p_2@c{:}\sigma \Rightarrow \Gamma_1,\Gamma_2}\,\mathrm{Contract} \qquad \frac{}{\star@c{:}1 \Rightarrow \varepsilon}\,1\mathrm{L}$$

$$\frac{p_1@\pi_1 c{:}\sigma_1 \Rightarrow \Gamma_1 \qquad p_2@\pi_2 c{:}\sigma_2[\pi_1 c/\alpha] \Rightarrow \Gamma_2}{\langle p_1, p_2\rangle@c{:}\Sigma\alpha{:}\sigma_1.\sigma_2 \Rightarrow \Gamma_1,\Gamma_2}\,\Sigma\mathrm{L}$$

型付け規則を与える前に、モジュールから静的部分を取り出す $\Gamma \vdash \mathrm{Fst}(M) \gg c$ を定義します。

$$\frac{}{\Gamma \vdash \mathrm{Fst}(\star) \gg \star} \qquad \frac{m@c{:}\sigma \in \Gamma}{\Gamma \vdash \mathrm{Fst}(m) \gg c} \qquad \frac{}{\Gamma \vdash \mathrm{Fst}(\langle\!| c |\!\rangle) \gg c}$$

$$\frac{}{\Gamma \vdash \mathrm{Fst}(\langle\!| e |\!\rangle) \gg \star} \qquad \frac{p@\alpha{:}\sigma \Rightarrow \Gamma' \qquad \Gamma, \alpha{:}\mathrm{Fst}(\sigma), \Gamma' \vdash \mathrm{Fst}(M) \gg c}{\Gamma \vdash \mathrm{Fst}(\lambda\alpha/p{:}\sigma.M) \gg \lambda\alpha{:}\mathrm{Fst}(\sigma).c}$$

$$\frac{\Gamma \vdash \mathrm{Fst}(M_1) \gg c_1 \qquad \Gamma \vdash \mathrm{Fst}(M_2) \gg c_2}{\Gamma \vdash \mathrm{Fst}(M_1\ M_2) \gg c_1\ c_2}$$

$$\frac{\Gamma \vdash \mathrm{Fst}(M_1) \gg c_1 \qquad \Gamma \vdash \mathrm{Fst}(M_2) \gg c_2}{\Gamma \vdash \mathrm{Fst}(\langle M_1, M_2\rangle) \gg \langle c_1, c_2\rangle} \qquad \frac{\Gamma \vdash \mathrm{Fst}(M) \gg c}{\Gamma \vdash \mathrm{Fst}(\pi_i M) \gg \pi_i c}$$

$$\frac{\Gamma \vdash \mathrm{Fst}(M_1) \gg c_1 \qquad p@\alpha{:}\sigma \Rightarrow \Gamma' \qquad \Gamma, \alpha{:}\mathrm{Fst}(\sigma), \Gamma' \vdash \mathrm{Fst}(M_2) \gg c_2}{\Gamma \vdash \mathrm{Fst}(\mathrm{let}\ \alpha/p = M_1\ \mathrm{in}\ M_2) \gg c_2[c_1/\alpha]}$$

型付け規則を与えます。まずは focus 規則。

$$\frac{}{\Gamma \vdash \star : 1}\,1\mathrm{R} \qquad \frac{\Gamma \vdash M_1 : \sigma_1 \qquad \Gamma \vdash M_2 : \sigma_2}{\Gamma \vdash \langle M_1, M_2\rangle : \sigma_1 \times \sigma_2}\,\Sigma\mathrm{R} \qquad \frac{}{\Gamma \vdash m : \Gamma(m)}\,\mathrm{Hyp}$$

$$\frac{\Gamma \vdash M_1 : \Pi\alpha{:}\sigma'.\sigma \qquad \Gamma \vdash M_2 : \sigma' \qquad \Gamma \vdash \mathrm{Fst}(M_2) \gg c}{\Gamma \vdash M_1\ M_2 : \sigma[c/\alpha]}\,\Pi\mathrm{E}$$

$$\frac{\Gamma \vdash M : \Sigma\alpha{:}\sigma_1.\sigma_2}{\Gamma \vdash \pi_1 M : \sigma_1}\,\Sigma\mathrm{E1} \qquad \frac{\Gamma \vdash M : \Sigma\alpha{:}\sigma_1.\sigma_2 \qquad \Gamma \vdash \mathrm{Fst}(M) \gg c}{\Gamma \vdash \pi_2 M : \sigma_2[\pi_1 c/\alpha]}\,\Sigma\mathrm{E2}$$

$$\frac{\Gamma \vdash c : k}{\Gamma \vdash \langle\!| c |\!\rangle : \langle\!| k |\!\rangle}\,\mathrm{Static} \qquad \frac{\Gamma \vdash e : c}{\Gamma \vdash \langle\!| e |\!\rangle : \langle\!| c |\!\rangle}\,\mathrm{Dynamic}$$

そして、inversion 規則。

$$\frac{p@\alpha{:}\sigma \Rightarrow \Gamma' \qquad \Gamma, \alpha{:}\mathrm{Fst}(\sigma), \Gamma' \vdash M : \sigma'}{\Gamma \vdash \lambda\alpha/p{:}\sigma.M : \Pi\alpha{:}\sigma.\sigma'}\,\Pi\mathrm{R}$$

$$\frac{\Gamma \vdash M_1 : \sigma_1 \qquad p@\alpha{:}\sigma_1 \Rightarrow \Gamma' \qquad \Gamma, \alpha{:}\mathrm{Fst}(\sigma_1), \Gamma' \vdash M_2 : \sigma_2 \qquad \alpha \notin \mathrm{FTV}(\sigma_2)}{\Gamma \vdash \mathrm{let}\ \alpha/p = M_1\ \mathrm{in}\ M_2 : \sigma_2}\,\mathrm{Let}$$

Retyping 規則。

$$\frac{\Gamma \vdash M : \langle\!| k' |\!\rangle \qquad \Gamma \vdash \mathrm{Fst}(M) \gg c \qquad \Gamma \vdash c : k}{\Gamma \vdash M : \langle\!| k |\!\rangle}\,\eta\mathrm{Static}$$

$$\frac{\Gamma \vdash \pi_1 M : \sigma_1 \qquad \Gamma \vdash \pi_2 M : \sigma_2}{\Gamma \vdash M : \sigma_1 \times \sigma_2}\,\eta\Sigma$$

$$\frac{\Gamma \vdash M : \Pi\alpha{:}\sigma_1.\sigma_2' \qquad \Gamma, \alpha{:}\mathrm{Fst}(\sigma_1), m@\alpha{:}\sigma_1 \vdash M\ m : \sigma_2}{\Gamma \vdash M : \Pi\alpha{:}\sigma_1.\sigma_2}\,\eta\Pi$$

$$\frac{\Gamma \vdash M : \sigma \qquad \Gamma \vdash \sigma \leq \sigma'}{\Gamma \vdash M : \sigma'}\,\mathrm{Sub}$$

Weak 規則は弱化、Contract 規則は縮約、Sub 規則は包摂を表しています。

ファンクタ適用 (ΠE) とストラクチャの射影 (ΣE1, ΣE2) は、シークエント計算ではなく自然演繹スタイルの規則となっています。これは自然演繹スタイルの方がより馴染みのある構文・規則になるからというのもありますが、ストラクチャの第 2 要素の射影についてはもう 1 つ理由があります。それは、依存和 (Σ) の negative connective としての left rule を上手く扱うためには更なる技巧が必要になるからです [11]。(ちなみに、ΣL 規則は依存和の positive connective としての left rule ですので、そのような問題は起きません。) 面白いことに、その技巧とは ML のモジュールシステムで伝統的に扱われている **selfification** という操作を利用することであったりします。そして、selfification は本稿での η 規則 (ηStatic, $\eta\Sigma$, $\eta\Pi$) と深い関わりがあります。これら η 規則は「モジュール M の η 展開 $\eta(M)$ がシグネチャ σ を持つとき、M 自身もシグネチャ σ を持つべき」という観測を反映したものです。

本稿では sealing $M :> \sigma$ を扱いません。しかし、[12] のようにモナドを導入すれば扱えるはずです。また、部分シグネチャ関係などの規則は [12] のサブセットであるので本稿では省略しています。

2.2. 操作的意味論

通常の left-to-right の値呼びの意味論を与えます。まず、値と評価文脈を定義します。

$$
\begin{array}{rcl}
v & ::= & \lambda x{:}c.e \\
V & ::= & \star \mid \langle\!| c |\!\rangle \mid \langle\!| v |\!\rangle \mid \lambda \alpha/p{:}\sigma.M \mid \langle V, V \rangle \\
E_{\mathrm{e}} & ::= & [] \mid E_{\mathrm{e}}\ e \mid v\ E_{\mathrm{e}} \mid \mathrm{Ext} E_{\mathrm{M}} \\
E_{\mathrm{M}} & ::= & [] \mid \langle\!| E_{\mathrm{e}} |\!\rangle \mid E_{\mathrm{M}}\ M \mid V\ E_{\mathrm{M}} \mid \langle E_{\mathrm{M}}, M \rangle \mid \langle V, E_{\mathrm{M}} \rangle \mid \pi_1 E_{\mathrm{M}} \mid \pi_2 E_{\mathrm{M}} \mid \mathrm{let}\ \alpha/p = E_{\mathrm{M}}\ \mathrm{in}\ M \\
E & ::= & E_{\mathrm{e}} \mid E_{\mathrm{M}}
\end{array}
$$

パターンマッチの意味論を定義するために、パターン代入 $M\{V/p\}$ を与えます。

$$
\begin{aligned}
M\{V/\top\} &\equiv M \\
M\{V/m\} &\equiv M[V/m] \\
M\{\star/\star\} &\equiv M \\
M\{\langle V_1, V_2 \rangle/\langle p_1, p_2 \rangle\} &\equiv M\{V_1/p_1\}\{V_2/p_2\} \\
M\{V/p_1 \wedge p_2\} &\equiv M\{V/p_1\}\{V/p_2\}
\end{aligned}
$$

簡約規則を与えます。ここで、$\mathrm{Fst}(V)$ は $\varepsilon \vdash \mathrm{Fst}(V) \gg c$ なる一意の c を表します。

$$
\begin{aligned}
\mathrm{let}\ \alpha/p = V\ \mathrm{in}\ M &\hookrightarrow M[\mathrm{Fst}(V)/\alpha]\{V/p\} \\
(\lambda \alpha/p{:}\sigma.M)\ V &\hookrightarrow M[\mathrm{Fst}(V)/\alpha]\{V/p\} \\
\pi_i \langle V_1, V_2 \rangle &\hookrightarrow V_i \\
(\lambda x{:}c.e)\ v &\hookrightarrow e[v/x] \\
\mathrm{Ext} \langle\!| v |\!\rangle &\hookrightarrow v \\
E[e] &\hookrightarrow E[e'] \qquad \text{if} \quad e \hookrightarrow e' \\
E[M] &\hookrightarrow E[M'] \qquad \text{if} \quad M \hookrightarrow M'
\end{aligned}
$$

2.3. メタ理論

型健全性は進行・保存によって特に問題なく証明できるはずです。

2.4. 例

例を見てみましょう。次の例では、ファンクタ m_F をモジュール m_X に適用して、その結果をワイルドカードパターン $\top$ により捨てています。

$$\text{let } \alpha/\top \;=\; m_F\; m_X \text{ in } \star$$

パターン $\top$ は任意のシグネチャに対応しているので、ファンクタであってもマッチします。

$$\text{let } \alpha/\top \;=\; \lambda\alpha/m{:}\langle\!|\,\Omega\,|\!\rangle.m \text{ in } \star$$

次の例では、高階ファンクタ m_H に対して、引数を使わないファンクタ $\lambda\beta/\top{:}\langle\!|\,1\,|\!\rangle.\star$ を与えています。

$$\text{let } \alpha/m_H \;=\; \lambda\beta/m{:}\left(\Pi\gamma{:}\langle\!|\,1\,|\!\rangle.\sigma\right).m\,\langle\!|\,\star\,|\!\rangle \text{ in } m_H\,\left(\lambda\beta/\top{:}\langle\!|\,1\,|\!\rangle.\star\right)$$

ストラクチャからコンポーネントを取り出す $\langle p_1, p_2\rangle$ パターンは次のようになります。

$$\text{let } \alpha/\langle m_t, m_x\rangle \;=\; m_F\; m_X \text{ in } \langle\!|\,\mathrm{Ext} m_X\,|\!\rangle$$

ここで、m_t はシグネチャ $\langle\!|\,\mathrm{S}(\pi_1\alpha)\,|\!\rangle$ を持ちます。モジュール変数 m_t を利用しない場合は次のようにワイルドカードパターンも使えます。

$$\text{let } \alpha/\langle\top, m_x\rangle \;=\; m_F\; m_X \text{ in } m_X$$

And パターンを使うことで、as パターンも表現できます。

$$\text{let } \alpha/m \;\wedge\; \langle m_x, \top\rangle \;=\; M \text{ in } \langle m_x, m\rangle$$

3. さいごに

本稿で扱わなかった重要な機能は **enrichment** です。Enrichment とは、ストラクチャの幅部分型付け、それから多相的な項の instantiation を表現する関係です。Enrichment は型理論では直接扱おうとすると複雑になりすぎるので、[5] のように elaboration で扱うのが通例です。

4. 参考文献

[1] Robert Harper, John C. Mitchell and Eugenio Moggi. Higher-order modules and the phase distinction. 1990.

[2] Jean-Marc Andreoli. Logic programming with focusing proofs in linear logic. 1992.

[3] Jean-Yves Girard. On the unity of logic. 1993.

[4] Robert Harper and John C. Mitchell. On the type structure of Standard ML. 1993.

[5] Robert Harper and Christopher A. Stone. A type-theoretic interpretation of Standard ML. 2000.

[6] Derek Dreyer. Understanding and evolving the ML module system. 2005.

[7] Daniel K. Lee, Karl Crary and Robert Harper. Towards a mechanized metatheory of Standard ML. 2007.

[8] Neelakantan R. Krishnaswami. Focusing on pattern matching. 2009.

[9] Noam Zeilberger. The logical basis of evaluation order and pattern-matching. 2009.

[10] Robert J. Simmons. Structural focalization. 2014.

[11] Karl Crary. Strong sums in focused logic. 2018.

[12] Karl Crary. A focused solution to the avoidance problem. 2020.