CCG に基づく言語的計算効果の評価遅延（補足資料）

松岡 大樹 [1,2]　谷中 瞳 [1,2,3]

[1] 東京大学　[2] 理化学研究所　[3] 東北大学

{daiki.matsuoka, hyanaka}@is.s.u-tokyo.ac.jp

言語処理学会第 32 回年次大会 (NLP2026) [TS2-7]

本資料は，NLP2026 でのポスター発表[1)]の補足として，ポスターでは紙面の都合上記載できなかった意味合成に関する詳細を説明するものである．具体的には，

- 意味合成を定式化する上で有用となる**モナド (monad)** の概念を導入し，
- 先行研究におけるモナドに基づく**照応の合成的な分析**を紹介した上で，
- 本研究が提案する規則に基づく意味合成の過程を説明する．

1　モナド

動機付け　自然言語の一部の意味現象は，プログラミング言語意味論における**計算効果 (computational effect)** の概念に対応している (Shan, 2002; Barker & Shan, 2014; Bumford & Charlow, in press)

→ **形式意味論の分析に，計算効果を扱うための数学的な概念を応用できないか？**

言語学における意味現象	言語表現の例	対応する計算効果
不定性 (indefiniteness)	*a/an*, *some*, *A or B*	非決定性
前提 (presupposition)	*the*, *stop …ing*	失敗しうる計算のエラーハンドリング
スコープ (scope)	*every*, *no*	継続を介した制御（詳細は後ほど）

基本的な定義　モナド ··· 計算効果を持つプログラムが持つ性質を一般化した代数的構造のこと (Wadler, 1995; Atkey, 2009)

> **定義 1（指標付きモナド）**　以下から成る 3 つ組 $\langle \mathsf{T}, \eta, \gg\!= \rangle$ であって，かつモナド則 (monad law) という 3 つの等式を満たすものを，**指標付きモナド (indexed monad)** という（以下では単にモナドという）．
>
> - 構成要素
> - ${}^{r}\mathsf{T}^{o}\alpha := (\alpha \to o) \to r$（$\alpha, r, o$ は型）
> - $\eta : \alpha \to {}^{r}\mathsf{T}^{r}\alpha$
> - $\gg\!= : {}^{r}\mathsf{T}^{o}\alpha \to (\alpha \to {}^{o}\mathsf{T}^{o'}\beta) \to {}^{r}\mathsf{T}^{o'}\beta$

1)　予稿：https://www.anlp.jp/proceedings/annual_meeting/2026/pdf_dir/TS2-7.pdf

- モナド則
  - 左単位則 (left identity): $\eta\, v \gg\!= f = f\, v$
  - 右単位則 (right identity): $m \gg\!= \eta = m$
  - 結合則 (associativity): $(m \gg\!= f) \gg\!= g = m \gg\!= \lambda v.\, (f\, v \gg\!= g)$

- 各構成要素に対する直感的な説明
  - 型 ${}^{r}\mathsf{T}^{o}\alpha$：型 α の値に計算効果を付して得られる値の型．この効果による状態変化を r（変化前）と o（変化後）により指定している．
  - 1 項関数 η：値に「何もしない」効果を付与する
    - 実際，$\eta\, v$ の型 ${}^{r}\mathsf{T}^{r}\alpha$ において状態の変数は r のまま変化しない
  - 2 項関数 $\gg\!=$（bind と読まれる）：計算効果を持つ 2 つのプログラムを連結する
    - (i) まず ${}^{r}\mathsf{T}^{o}\alpha$ を評価して，型 α の値を得てから，
    - (ii) その値を後続する計算 $\alpha \to {}^{o}\mathsf{T}^{o'}\beta$ に渡す
- 以下の **do 記法 (do-notation)** を用いると分かりやすくなる

> **定義 2（do 記法）**
>
> - $\mathtt{pure}\, v = \eta\, v$
> - $\mathtt{do}\,\{x_1 \leftarrow m_1;\, \ldots;\, x_n \leftarrow m_n;\, \pi\} := m_1 \gg\!= (\lambda x_1. \cdots (m_n \gg\!= (\lambda x_n.\, \pi)))$

  - モナド則を do 記法で書き直すと…
    - 左単位則：$\mathtt{do}\,\{v \leftarrow \mathtt{pure}\, x;\, f\, v\} \;=\; f\, x$
    - 右単位則：$\mathtt{do}\,\{v \leftarrow m;\, \mathtt{pure}\, v\} \;=\; m$
    - 結合則：$\mathtt{do}\,\{v' \leftarrow \mathtt{do}\,\{v \leftarrow m;\, f\, v\};\, g\, v'\} = \mathtt{do}\,\{v \leftarrow m;\, v' \leftarrow f\, v;\, g\, v'\}$

- モナド則に対する直感的な説明
  - 左単位則：x を pure に渡してから関数 f に渡すのは，単に f に x を渡すのと同じ
  - 右単位則：m から値 v を取り出して pure に渡すのは，単に m を実行するのと同じ
  - 結合則：入れ子になった do を平坦にする操作を正当化
    - 左辺側で一番深い位置にある $v \leftarrow m$ が，右辺側では最上位へと移されている

継続　ここでは，具体的な計算効果として，「自身の**継続 (continuation)** に操作を施す」というものを考える．

- 継続 $\cdots$ ある値の次に評価される計算のこと
  - 例：$(y + x) \times z$ における x の継続 $\cdots (y + \_\_) \times z$

- 継続を扱うためのモナド（モナド則の証明は割愛）

> **定義 3（継続モナド）**　以下からなる 3 つ組 $\langle \mathsf{C}, \eta, \gg\!= \rangle$ を，継続モナドという．
>
> - ${}^{r}\mathsf{C}^{o}\alpha := (\alpha \to o) \to r$
> - $\eta\, v := \lambda k.\, k\, v$

- $m \gg\!= f := \lambda k.\, m\, (\lambda x.\, f\, x\, k)$

- $\alpha \to o$ が値 α の継続の型で，r は計算全体の返り値の型．
- $\alpha = \mathsf{e},\ o = r = \mathsf{t}$ だと，古典的な一般化量化子の型と一致することに注意

2　モナドに基づく照応の分析

分析の前準備　まず，意味表示に用いるための補助的な定数をいくつか導入する．

- 量化
  - $\mathtt{ev} : {}^{r}\mathsf{C}^{r}\mathsf{e} := \lambda k.\forall x.\, k\, x$（全称量化詞の意味表示）
- 照応
  - $\mathtt{bind} : \alpha \to {}^{r}\mathsf{C}^{\alpha \to r}\alpha := \lambda x.\lambda k.\, k\, x\, x$（先行詞に適用する変換）
    - 直感：後続する状態 $\alpha \to r$ に自身を引数として渡す
  - $\mathtt{pro} : {}^{\mathsf{e} \to r}\mathsf{C}^{r}\mathsf{e} := \lambda k.\lambda x.\, k\, x$（代名詞の意味表示）
    - 直感：評価前の状態 $\mathsf{e} \to r$ から e（＝参照先）を取り出す
  - 補足：bind と pro は互いに相殺しあう（補題 $(*)$：証明は割愛）

$$\mathtt{do}\,\{x \leftarrow \mathtt{bind}\, a;\ y \leftarrow \mathtt{pro};\ f\, x\, y\} = f\, a\, a\ \cdots\ (*)$$

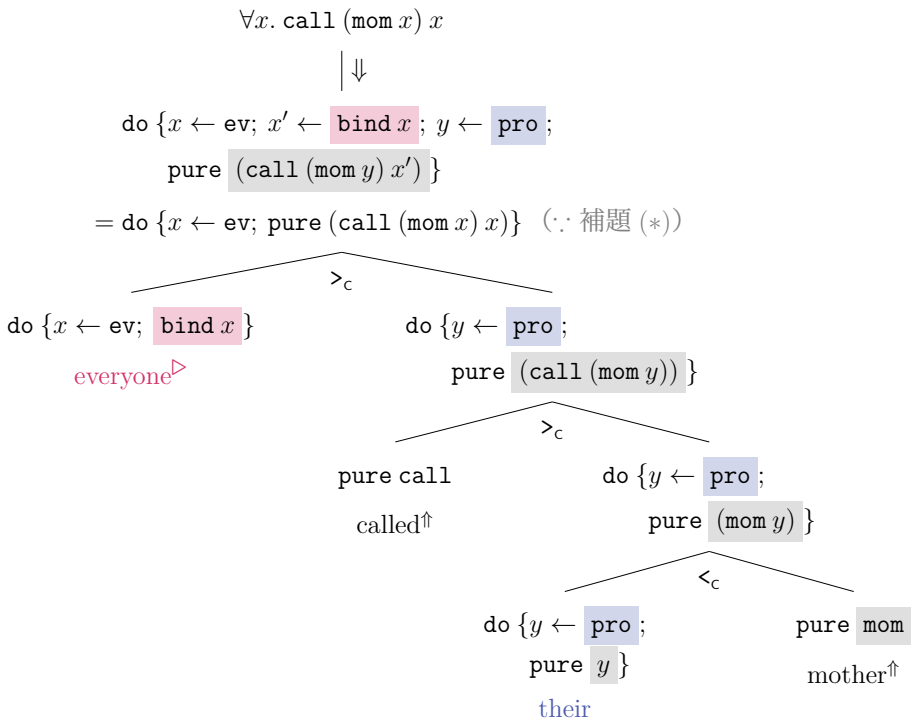
意味合成の規則　通常の関数適用に加えて，計算効果を扱うための規則を導入（ここでは統語範疇は割愛する）

- 関数適用
  - $(\mathtt{>})\ f\ x := f\ x$
  - $(\mathtt{<})\ x\ f := f\ x$
- **計算効果を考慮した関数適用**
  - $(\mathtt{>_c})\ F\ X := \mathtt{do}\,\{f \leftarrow F;\ x \leftarrow X;\ \mathtt{pure}\,(f\ x)\}$
  - $(\mathtt{<_c})\ X\ F := \mathtt{do}\,\{x \leftarrow X;\ f \leftarrow F;\ \mathtt{pure}\,(f\ x)\}$
- 補助的な規則群
  - $(\Uparrow)\ x = \mathtt{pure}\ x$
  - $(\Downarrow)\ X = X\ (\lambda x.\, x)$
  - $(\triangleright)\ X = \mathtt{do}\,\{x \leftarrow X;\ \mathtt{bind}\ x\}$

具体例への適用

(1)　Everyone^{i} called their$_{i}$ mother.　$(\rightsquigarrow \forall x.\, \mathtt{call}\,(\mathtt{mom}\ x)\ x)$

- 導出の過程



- 導出過程に関する補足
  - 全体として，計算効果を線形に合成し，pure 内部で述語項構造を計算
    - この挙動はモナド則の帰結（以下の計算過程を参照）

$$
\begin{aligned}
(<_c)\,(\mathtt{do}\,\{y \leftarrow \mathtt{pro};\ \mathtt{pure}\ y\})\,((\Uparrow)\,\mathtt{mom}) &= \mathtt{do}\,\{z \leftarrow \mathtt{do}\,\{y \leftarrow \mathtt{pro};\ \mathtt{pure}\ y\};\ f \leftarrow \mathtt{pure}\,\mathtt{mom};\\
&\qquad \mathtt{pure}\,(f\ z)\}\\
&= \mathtt{do}\,\{y \leftarrow \mathtt{pro};\ z \leftarrow \mathtt{pure}\ y;\ f \leftarrow \mathtt{pure}\,\mathtt{mom};\\
&\qquad \mathtt{pure}\,(f\ z)\}\ (\because \text{結合則})\\
&= \mathtt{do}\,\{y \leftarrow \mathtt{pro};\ \mathtt{pure}\,(\mathtt{mom}\ y)\}\ (\because \text{左単位則})
\end{aligned}
$$

  - 最後のステップ (⇓) は，$x \leftarrow \mathtt{ev}$ のスコープを閉じる操作に相当

$$
\begin{aligned}
(\Downarrow)\,\mathtt{do}\,\{x \leftarrow \mathtt{ev}; \cdots\} &= (\mathtt{ev} \gg\!= \lambda x.\ \eta\,(\mathtt{pure}\,(\mathtt{call}\,(\mathtt{mom}\ x)\ x)))\,(\lambda x.\ x)\\
&= (\lambda k.\ (\lambda k'.\ \forall x.\ k'\ x)\ (\lambda x.\ (\lambda k''.\ k''\ (\mathtt{call}\,(\mathtt{mom}\ x)\ x))\ k))\ (\lambda x.\ x)\\
&= \forall x.\ \mathtt{call}\,(\mathtt{mom}\ x)\ x
\end{aligned}
$$

理論的含意　ここでの分析は，基本的に線形順序に従って照応を導出する

- いわゆる**交差効果** (crossover effect; Postal, 1971) は，線形順序の問題として説明される

  (2)　*Their$_i$ mother called everyone^i.

- 交差効果を**統語構造上の関係**の問題とみなす伝統的な見解とは異なる
  - Reinhart (1983): A 位置からの **c 統御 (c-command)** の有無に帰着
  - しかし，c 統御だけでは説明がつかない交差効果の事例が様々に存在し (Barker, 2012; Elliott & Sudo, 2021)，線形順序を考慮する理論にも一定の注目がなされている (Barker & Shan, 2014; Chierchia, 2020; Matsuoka et al., 2025)

(3) 所有表現からの束縛

a. [Everyone^i's teacher] called their$_i$ mother.

b. *Their$_i$ mother called [everyone^i's teacher].

(4) ロバ文

a. Every farmer [who owns a^i donkey] adores its$_i$ strength.

b. *Its$_i$ strength impressed every farmer [who owns a^i donkey].

(5) 前提投射との相互作用

a. Alex did not **know** [that Kim wrote a^i paper], and reviewed it$_i$.

b. Its$_i$ reviewer did not **know** [that Kim wrote a^i paper].

再構築　語順の入れ替えを伴う構文（例：wh 疑問文）では，**線形順序に従わない照応**が生じることがある (Cresti, 1995) → 今回の枠組みでどう分析するか？

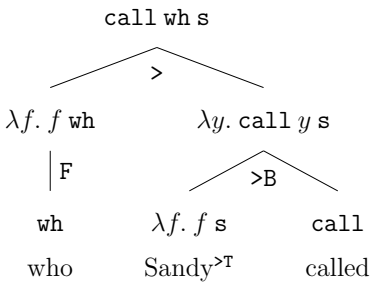
(6) [Which of their$_i$ relatives] did everyone^i call __?

- アイデア：**評価遅延としての再構築** (Barker, 2009)
  - wh 句の元位置に計算効果を伴う変数 X を置いて λ で束縛することで，遅延された wh 句の評価が行われる場所を保持できるようにしたい

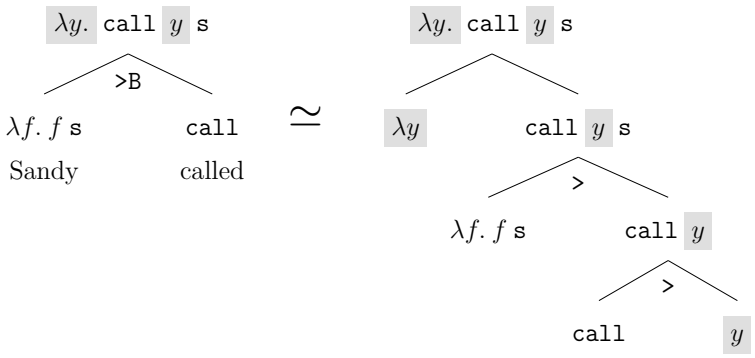
(7) λX. did everyone^i call X

3 関数合成を介した再構築の分析

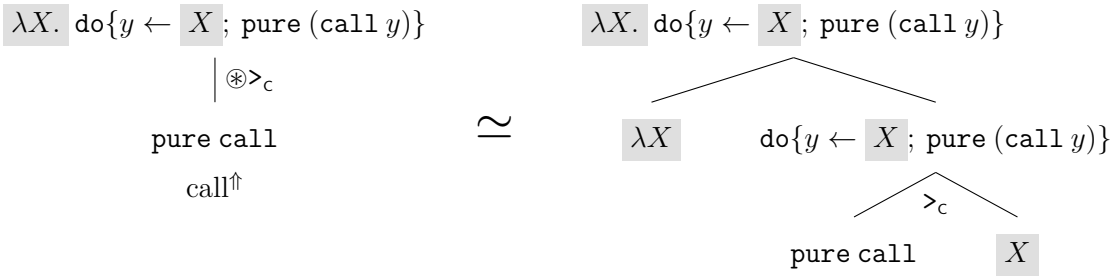
- wh 疑問文の分析：**関数合成 (function composition)** を用いる手法 (Steedman, 2000)
  - 関数合成
    - (>B) $g\ f := \lambda x.\, g\,(f\ x)$
    - (<B) $f\ g := \lambda x.\, g\,(f\ x)$
  - 補助的な規則群
    - 型繰り上げ (>T) $x := \lambda f.\, f\ x$（<T も同様）
    - 前置 (F) $x := \lambda f.\, f\ x$
  - 導出の例（ここでは *who* を定数として単純化）



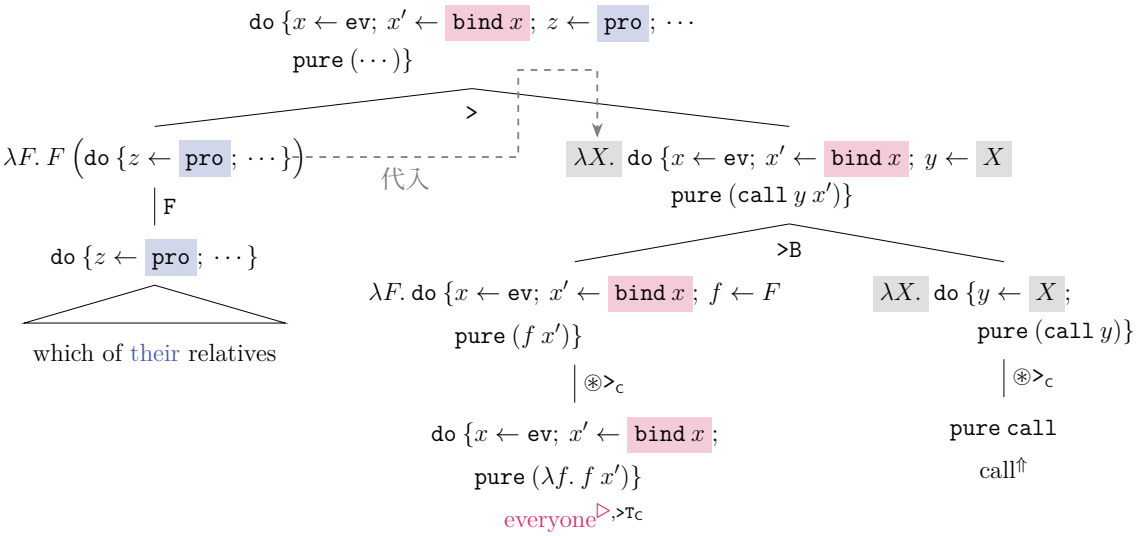
- 直感：目的語を変数 y として仮置きし，そこに後から wh 句を代入



- **関数の分割（提案）**：効果を伴う関数を，効果を伴う値の間の高階の関数へと変換する
  - 規則の定義は，**関数適用をカリー化**したもの
    - $(\circledast{>_c})\ F := \lambda X.\ (>_c)\ F\ X \quad (= \lambda X.\ \mathtt{do}\{f \leftarrow F;\ x \leftarrow X;\ \mathtt{pure}\ (f\ x)\})$
    - $(\circledast{<_c})\ F := \lambda X.\ (<_c)\ X\ F \quad (= \lambda X.\ \mathtt{do}\{x \leftarrow X;\ f \leftarrow F;\ \mathtt{pure}\ (f\ x)\})$
  - 直感：変数 X を仮で置いて関数適用してから，λ で束縛



- 導出の過程（*did* は省略）：X に wh 句の意味表示が代入された結果，pro の評価が bind の後へと遅延される



参考文献

Atkey, R. (2009). Parameterised notions of computation. *Journal of Functional Programming*, *19*(3–4), 335–376. https://doi.org/10.1017/S095679680900728X

Barker, C. (2009). Reconstruction as delayed evaluation.

Barker, C. (2012). Quantificational binding does not require c-command. *Linguistic Inquiry*, *43*(4), 614–633. https://doi.org/10.1162/ling_a_00108

Barker, C., & Shan, C.-c. (2014). *Continuations and Natural Language*. Oxford University Press.

Bumford, D., & Charlow, S. (in press). *Effect-driven interpretation: Functors for natural language composition*. https://ling.auf.net/lingbuzz/006884

Chierchia, G. (2020). Origins of weak crossover: When dynamic semantics meets event semantics. *Natural Language Semantics*, *28*(1), 23–76. https://doi.org/10.1007/s11050-019-09158-3

Cresti, D. (1995). Extraction and reconstruction. *Natural Language Semantics*, *3*(1), 79–122. https://doi.org/10.1007/BF01252885

Elliott, P. D., & Sudo, Y. (2021). Generalised crossover. *Semantics and Linguistic Theory (SALT) 30*, 396–408. https://doi.org/10.3765/salt.v30i0.4841

Matsuoka, D., Bekki, D., & Yanaka, H. (2025). A propositions-as-types approach to the generalized crossover effect. *Sinn und Bedeutung (SuB) 29*, 986–1003. https://doi.org/10.18148/sub/2024.v29.1258

Postal, P. (1971). *Cross-over phenomena*. Holt, Rinehart; Winston.

Reinhart, T. (1983). *Anaphora and semantic interpretation*. Routledge.

Shan, C.-c. (2002). Monads for natural language semantics. In K. Striegnitz (Ed.), *Proceedings of the ESSLLI 2001 Student Session* (pp. 285–298).

Steedman, M. (2000). *The Syntactic Process*. MIT Press.

Wadler, P. (1995). Monads for functional programming. In *International school on advanced functional programming* (pp. 24–52). Springer.