CCG に基づく言語的計算効果の評価遅延

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

概要

自然言語の意味解釈は，統語構造だけでなく，線形順序にも依存する場合がある．理論言語学では，代名詞などに見られるこの種の順序依存性に対して，プログラミング言語での逐次実行に伴う計算効果（副作用）と同様の形式的分析を与えられることが示唆されている．しかし，wh 疑問文のように語順の入れ替えを伴う構文では，計算効果の評価が表層的な位置よりも後ろに遅延される場合がある．本研究は，代名詞束縛の現象に注目し，Combinatory Categorial Grammar (CCG) の枠組みにおける wh 疑問文の分析を応用することで，代名詞の計算効果の評価遅延に対して理論的分析を与える．

1　はじめに

自然言語の形式意味論における基本的な作業仮説として，意味計算は統語構造に従ってボトムアップに行われるという**合成性の原理 (principle of compositionality)** がある．このようなアプローチのもとでは，表層的な順序に依存して意味解釈が決まる表現の扱いが課題とされてきた．代表的な例として，代名詞の**束縛 (binding)** を考えてみよう．(1) に示すとおり，代名詞は，量化詞に束縛された変数のように解釈されることがある．

(1)　Everyone called their mother.
　　(✓ For each person x, x called x's mother.)

このような束縛の可否は，線形順序に影響される [1]．実際，代名詞と量化詞の順序を入れ替えた (2) では，束縛変数としての解釈は容認されない．

(2)　Their mother called everyone.
　　(✗ For each person x, x's mother called x.)

この種の順序依存の意味現象に合成的な分析を与えるにあたっては，プログラミング言語理論における**計算効果 (computational effect)**（ないし副作用 (side effect)）の概念が有効であることが指摘されている [2, 3, 4]．計算効果とは，純粋な値の計算とは異なる，プログラムの逐次的な実行に伴う影響（例：状態の更新や入出力）を指す概念である．代名詞束縛に関していえば，図 1 に示すように，参照する・されるという振る舞いをある種の計算効果として定式化することができる．

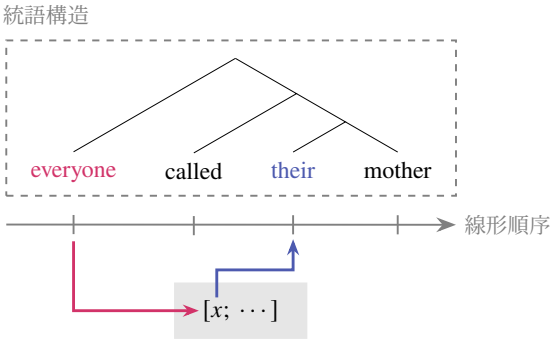


図 1　言語的計算効果の図示．代名詞の参照先を保存する状態を “everyone” が更新し，“their” が参照している．

しかし，この計算効果に基づくアプローチに対する課題として，wh 疑問文のような語順の入れ替えを伴う構文の扱いが挙げられる．例えば (3) では，代名詞が量化詞より後に現れるにもかかわらず，束縛が可能となる．

(3)　[Which of their relatives] did everyone call __ ?
　　(✓ For each person x, which of x's relatives did x call?)

これは，代名詞を含む wh 句の解釈が，“call” の目的語の位置（下線部）へと遅延されているとみなせる．このように語順の入れ替えに伴って解釈の位置が変わる振る舞いを**再構築 (reconstruction)** という [5]．

では，計算効果に基づく枠組みのもとで，どうすれば再構築に伴う代名詞の評価遅延を分析できるだろうか？本研究は，Combinatory Categorial Grammar (CCG) [6] における wh 疑問文の分析を応用することで，この問題に取り組む．具体的には，CCG における**関数合成 (function composition)** の規則を応用することが有効であることを示す．

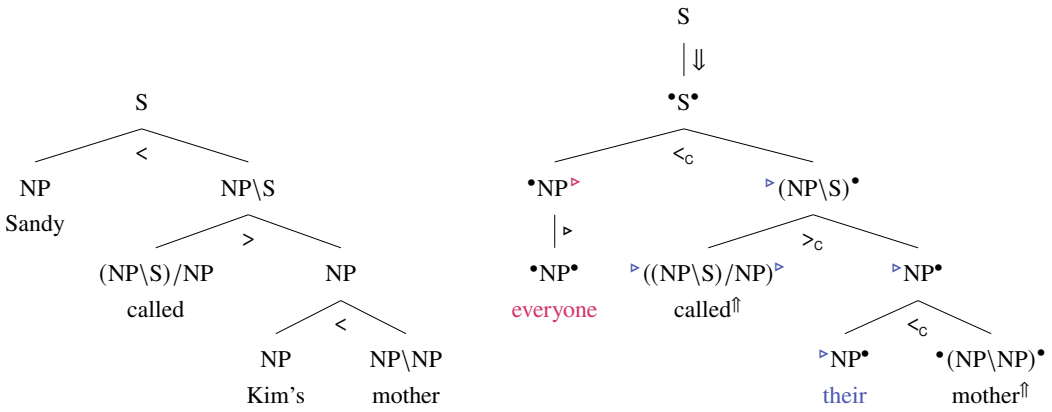


図 2　左：計算効果を伴わない導出の例．右：(1) の導出（なお，単項規則を適用する過程は適宜省略する）．

2　計算効果に基づく束縛の分析

本節では，まず以降の議論のベースとなる**範疇文法 (categorial grammar)** の枠組みを導入する．範疇文法では，各表現に対して文法的な型（範疇; category）が割り当てられ，これが複合的な表現を構成する方法を規定する．範疇には，S, NP 等の基本範疇に加えて，関数範疇 X/Y および $Y\backslash X$ があり，スラッシュの向きに対応して以下の関数適用規則が定義される．簡単な導出の例を図 2 左に示す．[1)]

関数適用規則

$$(>)\ B/A\ \ A \Longrightarrow B \qquad (<)\ \ A\ \ A\backslash B \Longrightarrow B$$

では，ここに計算効果を扱うための機構を導入しよう．まず，計算効果の情報を明示的に扱えるように範疇を拡張する．具体的には，${}^{e_1}X^{e_2}$ を，事前状態が e_1，事後状態が e_2 となる計算効果を持つ表現の範疇として導入する [8]．ここでは話を単純化して，状態の候補として代名詞の参照先の候補がある (▷)・ない (•) の 2 通りのみを考える．すると，例えば代名詞の範疇は ${}^{\triangleright}\mathrm{NP}^{\bullet}$ と書くことができる．

加えて，効果を伴う範疇を扱えるように組合せ規則を拡張する．まず，関数適用と同時に計算効果を左から右へと合成するような規則を定義する．

計算効果を考慮した関数適用規則

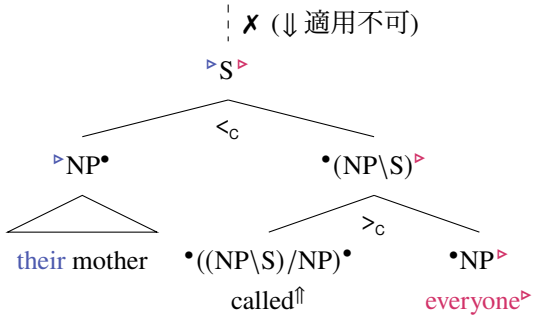
$$(>_c)\ \ {}^{e_1}(B/A)^{e_2}\ \ {}^{e_2}A^{e_3} \Longrightarrow {}^{e_1}B^{e_3}$$
$$(<_c)\ \ {}^{e_1}A^{e_2}\ \ {}^{e_2}(A\backslash B)^{e_3} \Longrightarrow {}^{e_1}B^{e_3}$$

次に，以下に示すいくつかの単項 (unary) 規則を導入する．⇑ は，通常の範疇を計算効果付きの範疇へと繰り上げる役割を持つ．逆に ⇓ は，そのような範疇を通常の範疇へと戻す役割を持つ．最後に ▷ は，代名詞を束縛できるよう量化詞を変換する．

補助的な規則群

$$(⇑)\ A \Longrightarrow {}^{e}A^{e} \qquad (⇓)\ {}^{\bullet}\mathrm{S}^{\bullet} \Longrightarrow \mathrm{S}$$
$$(\triangleright)\ {}^{e}\mathrm{NP}^{\bullet} \Longrightarrow {}^{e}\mathrm{NP}^{\triangleright}$$

以上を踏まえて，(1) の導出を図 2 右に示す．“their” に付随する ▷ に注目すると，これが複合的な表現へと引き継がれていき，最終的に主語の側の ▷ と相殺されていることがわかる．これにより ⇓ が適用でき，解消されない代名詞が残らない文 S の導出が完了する．一方で，(2) のように代名詞が先行する場合は，同様の導出はできない．実際，以下の導出に示すように，2 つの ▷ が相殺されないまま残るため，代名詞は束縛されることがない．



このように，計算効果の概念を用いることで，述語項構造レベルの計算と，順序に依存した束縛の計算とを同時に扱うことができる．しかし，1 節で述

1)　紙面の都合上，意味表示に関する詳細は付録に示す．なお，意味合成の過程に関する形式検証を，定理証明支援系 Agda [7] を用いて行った．コードは https://github.com/ynklab/NLP2026_split で公開している．

べた通り，このアプローチのもとで再構築を扱う方法は自明ではない．これを踏まえ，以降では，(i) 範疇文法の一種である CCG において wh 疑問文を扱える仕組みとして提案された関数合成の規則を説明し，その上で (ii) 計算効果に基づくアプローチに関数合成による分析を整合させる方法を提案する．

3 関数合成による wh 疑問文の分析

関数合成規則は，2 つの関数範疇を繋げるはたらきを持つ規則である．関数適用規則と同様，スラッシュの向きに応じて 2 種類が定義される．

関数合成規則

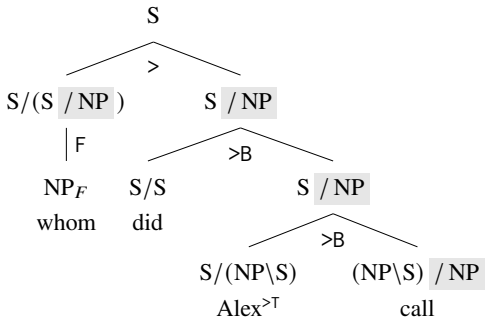
$$(>\mathsf{B})\ \ C/B\ \ B/A \implies C/A$$
$$(<\mathsf{B})\ \ A\backslash B\ \ B\backslash C \implies A\backslash C$$

また，ここでは，補助的な規則として，型繰り上げ (type-raising) と前置 (fronting) を用いる（なお，前置規則は素性 F を持つ範疇にのみ適用される）．

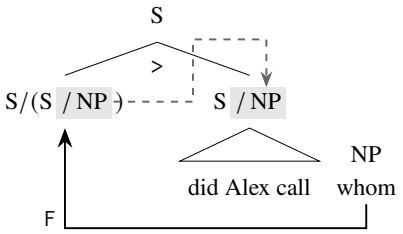
型繰り上げ規則・前置規則

$$(>\mathsf{T})\ \ A \implies B/(A\backslash B) \qquad (<\mathsf{T})\ \ A \implies (B/A)\backslash B$$
$$(\mathsf{F})\ \ A_F \implies \mathrm{S}/(\mathrm{S}/A)$$

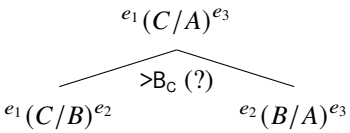
これらを用いて wh 疑問文を導出する方法を確認しよう．例として，“Whom did Alex call?” の導出を以下に示す．



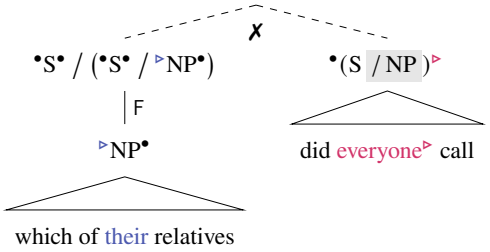
ここでは，まず，主語の “Alex” に型繰り上げ >T を適用し，NP を自動詞 NP\S から文 S への関数に変換している．すると，“did Alex call” を関数合成によって組み合わせることができ，全体としては欠けた目的語 / NP を引き継ぐような導出が構成される．この結果できた S/NP を，F 適用後の “whom” に引数として渡すことで，文 S が導出される．この最後のステップは，以下の図に示すように，本来 “did Alex call” の右側に現れるはずの目的語を左側から与えていると直感的に捉えることができる．



では，この関数合成による wh 疑問文の分析は，どのようにすれば 2 節の束縛の分析と組み合わせられるだろうか？素朴な方法として，以下のような計算効果を考慮したバージョンの関数合成を定義することが考えられる．



しかし，この規則だけでは，(3) で見たような再構築の現象を説明することができない．実際，上の素朴な規則を用いて導出を行うと “did everyone call” の範疇は •(S/NP)▷ となり，下に示すとおり，これは wh 句と組み合わせることができない．この帰結を回避するためには，•S• / e NP e′ のような，計算効果を伴う NP からの関数を構成する必要がある．



4 提案

以上を踏まえて，次の規則を提案する．

提案：計算効果を伴う関数の分割規則

$$(\circledast_{>})\ \ e_1(B/A)^{e_2} \implies {}^{e_1}B^{e_3} \,/\, {}^{e_2}A^{e_3}$$
$$(\circledast_{<})\ \ e_2(A\backslash B)^{e_3} \implies {}^{e_1}A^{e_2} \,\backslash\, {}^{e_1}B^{e_3}$$

この規則は，計算効果を伴う関数から，計算効果を伴う表現間の高階の関数を作り出すはたらきを持つ．直感的には，関数のスラッシュ /, \ で計算効果

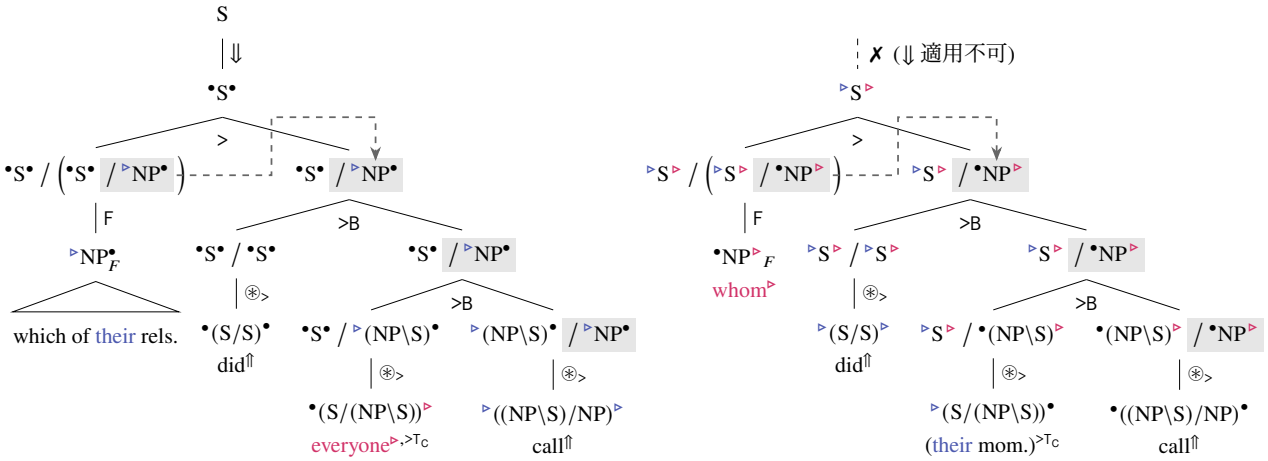
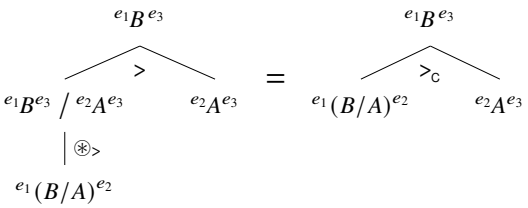
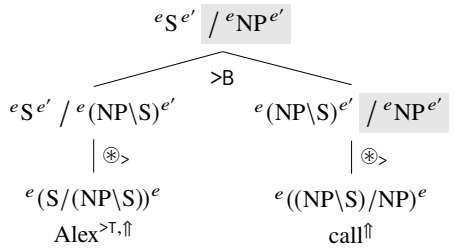


図 3　左：(3) の導出．右：(4) の導出．

が分割されているとみなせる．そしてこのとき，分割後の関数は，もとの関数が持っていた計算効果の順序を保持したものになっている．実際，$\circledast_>$ の直後に $>$ を適用した結果は，全体として $>_C$ を適用した結果と同一になる（$\circledast_<$ についても同様）．



ここで鍵となるのは，このような高階の関数に対しても関数合成を適用することができる，という点である．例として “Alex” と “call” の合成を考えると，それぞれに $\Uparrow$ を適用した後，$\circledast_>$ によって分割すれば，2 つを $>\mathbf{B}$ で組み合わせることができる．



これを踏まえて，再構築の例 (3) に対する導出を図 3 左に示す．欠けた目的語として計算効果付きの範疇 $/\,{}^{\triangleright}\mathrm{NP}^{\bullet}$ が引き継がれ，最終的に wh 句がその部分を埋めている．これにより，最終的に $\Downarrow$ を適用して S を導出でき，束縛が達成されたことがわかる．

さらに，この分析は，wh 句自身から代名詞への束縛にもそのまま適用できる．(4) が示すように，目的語の wh 句は，主語に含まれる代名詞を束縛することができない．

(4)　Whom did their mother call __ ?
　　(✗ Which person x is s.t. x’s mother called x?)

この文に対して束縛の解釈を導出しようと試みる様子を図 3 右に示す．仮に “whom” に $\triangleright$ を適用しても，最終的に文全体の範疇は ${}^{\triangleright}\mathrm{S}^{\triangleright}$ となり，代名詞が束縛されないままとなっていることがわかる．

5　関連研究

計算効果に基づくアプローチの先行研究 [8] は，wh 疑問文を扱うために，wh 句の評価位置に発音されない記号（空所; gap）が存在すると仮定していた．しかし，この手法では，(5) のように wh 句が節境界を超えて移動する長距離依存 (long-distance dependency) を扱えないことが指摘されている [9].

(5)　Whom does Sandy say [that Kim called __]?

本研究で採用した関数合成の規則は節境界をまたいで適用することができるため，このような長距離依存のケースにも対応可能であるという利点がある．

6　おわりに

本研究は，代名詞束縛を計算効果として扱う枠組みと，CCG の関数合成に基づく wh 疑問文の分析をベースとして，計算効果を伴う範疇間の高階の関数を作り出す規則を新たに導入することで，代名詞の評価が遅延される再構築現象に対する理論的分析を与えた．今後の課題として，表層的な順序の入れ替えを伴うその他の構文（例：日本語やヒンディー語のかき混ぜ (scrambling) [10]）へと分析を拡張することが挙げられる．

謝辞

本研究に関して有益なコメントをしていただいた戸次大介教授，窪田悠介教授，磯野真之介助教に感謝いたします．本研究は，JST BOOST JPMJBS2418, JSPS 科研費 JP24H00809, JST CREST JPMJCR2565 の支援を受けたものです．

参考文献

[1] Chris Barker. Quantificational binding does not require c-command. **Linguistic Inquiry**, Vol. 43, No. 4, pp. 614–633, 2012.

[2] Chung-chieh Shan. **Linguistic Side Effects**. PhD thesis, Harvard University, 2005.

[3] Simon Charlow. **On the semantics of exceptional scope**. PhD thesis, New York University, 2014.

[4] Dylan Bumford and Simon Charlow. **Effect-driven interpretation: Functors for natural language composition**. In press.

[5] Dominique Sportiche. Reconstruction, binding, and scope. In **The Wiley Blackwell Companion to Syntax, Second Edition**, pp. 1–58. John Wiley & Sons, Ltd, 2017.

[6] Mark Steedman. **The Syntactic Process**. MIT Press, 2000.

[7] The Agda Team. **Agda User Manual**, 2026. Release 2.9.0.

[8] Chris Barker and Chung-chieh Shan. **Continuations and Natural Language**. Oxford University Press, 2014.

[9] Cara Su-Yi Leong and Michael Yoshitaka Erlewine. Long-distance dependencies in continuation grammar. In **Proceedings of the 33rd Pacific Asia Conference on Language, Information, and Computation (PACLIC 33)**, pp. 114–122, 2019.

[10] Mamoru Saito. Long distance scrambling in Japanese. **Journal of East Asian Linguistics**, Vol. 1, No. 1, pp. 69–118, 1992.

[11] Philip Wadler. Monads for functional programming. In **International School on Advanced Functional Programming**, pp. 24–52. Springer, 1995.

[12] Robert Atkey. Parameterised notions of computation. **Journal of Functional Programming**, Vol. 19, No. 3–4, p. 335–376, 2009.

[13] Dylan Bumford and Simon Charlow. Dynamic semantics with static types. Manuscript, 2022.

付録：形式的定義

範疇　範疇の集合は，基本範疇を B として，以下のように帰納的に定義される．ここでは，基本範疇として少なくとも S, NP が含まれていると仮定する．

$$X ::= B \mid X/X \mid X\backslash X \mid X /\!\!/ X \mid X \backslash\!\!\backslash X \mid X \triangleright X$$

本文で用いた範疇 ${}^{e_1}X^{e_2}$ は，以下の略記として定義する．

$${}^{\bullet}X^{\bullet} := \mathrm{S} /\!\!/ (X \backslash\!\!\backslash \mathrm{S}) \quad {}^{\bullet}X^{\triangleright} := \mathrm{S} /\!\!/ (X \backslash\!\!\backslash (\mathrm{NP} \triangleright \mathrm{S}))$$

$${}^{\triangleright}X^{\bullet} := (\mathrm{NP} \triangleright \mathrm{S}) /\!\!/ (X \backslash\!\!\backslash \mathrm{S}) \quad {}^{\triangleright}X^{\triangleright} := (\mathrm{NP} \triangleright \mathrm{S}) /\!\!/ (X \backslash\!\!\backslash (\mathrm{NP} \triangleright \mathrm{S}))$$

組合せ規則　本文および本付録の議論に関係する範囲の規則を図 4 に示す．意味表示の詳細は以降で説明する．

モナド　計算効果を意味表示上で扱うにあたって，**モナド (monad)** の概念を用いる [11, 12]．紙面の都合により説明を簡略化するため，詳細は [3] や [13] を参照されたい．

まず，3 つ組 $\langle \mathsf{C}, \mathsf{pure}, \gg\!= \rangle$ を導入する．これを**指標付き継続モナド (indexed continuation monad)** という．

- ${}^{r}\mathsf{C}^{o}\alpha := (\alpha \to o) \to r$ （α, r, o は型）
- $\mathsf{pure} : \alpha \to {}^{r}\mathsf{C}^{r}\alpha$
  $\mathsf{pure}\ v := \lambda k.\ k\ v$
- $\gg\!= : {}^{r}\mathsf{C}^{o}\alpha \to (\alpha \to {}^{o}\mathsf{C}^{o'}\beta) \to {}^{r}\mathsf{C}^{o'}\beta$
  $m \gg\!= f := \lambda k.\ m\ (\lambda x.\ f\ x\ k)$

型 ${}^{r}\mathsf{C}^{o}\alpha$ の定義に現れる $\alpha \to o$ は，値 α の次に評価される計算（継続; continuation）の型であり，r は計算全体の返り値の型である．形式意味論では，量化詞がスコープを取る範囲を継続として定式化できることが知られている [8]．例えば，“everyone” の意味表示は次のようになる（ただし e は対象 (entity) の型である）．

(6) $\mathsf{ev} : {}^{r}\mathsf{C}^{r}\mathsf{e} := \lambda k.\forall x.\ k\ x$ （k が継続に対応する変数）

pure は，スコープを取らない純粋な値 v を C の値へと繰り上げる役割を持つ．一方 $\gg\!=$ は，m から値 $x : \alpha$ を取り出して関数 f に渡す役割を持つ．この逐次実行のような挙動を直感的にするために，以下の **do 記法**を用いる．

$$\begin{aligned}&\mathsf{do}\,\{x_1 \leftarrow m_1;\ \ldots;\ x_n \leftarrow m_n;\ \pi\}\\ &:= m_1 \gg\!= (\lambda x_1.\cdots(m_n \gg\!= (\lambda x_n.\ \pi)))\end{aligned}$$

なお，pure と $\gg\!=$ は以下のモナド則 (monad laws) という等式群を満たす（なお，等号には $\beta\eta$ 同値関係を用いる）．

- 左単位則：$\mathsf{do}\,\{v \leftarrow \mathsf{pure}\ x;\ f\ v\} = f\ x$
- 右単位則：$\mathsf{do}\,\{v \leftarrow m;\ \mathsf{pure}\ v\} = m$
- 結合則：$\mathsf{do}\,\{v' \leftarrow \mathsf{do}\,\{v \leftarrow m;\ f\ v\};\ g\ v'\}$
  $= \mathsf{do}\,\{v \leftarrow m;\ v' \leftarrow f\ v;\ g\ v'\}$

最後に，代名詞の束縛を扱うために，以下 2 つの定数を定義する．bind は規則 $\triangleright$ の定義に，pro は代名詞の意味表示に用いる．

(7) a. $\mathsf{bind} : \alpha \to {}^{r}\mathsf{C}^{\alpha \to r}\alpha := \lambda x.\lambda k.\ k\ x\ x$
b. $\mathsf{pro} : {}^{\mathsf{e} \to r}\mathsf{C}^{r}\mathsf{e} := \lambda k.\lambda x.\ k\ x$

このとき，以下の補題 $(*)$ が成り立つ（証明は割愛）．これは直感的には，bind の直後に pro を実行すると 2 つを相殺できることを示している．

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

導出の例　図 2 右の “everyone called their mother” の導出について，各ステップの意味表示を以下に示す．

- $[{}_{<\mathsf{c}}\text{their mother}^{\Uparrow}]$ の意味表示（これを s_1 とする）
  $(<_{\mathsf{c}})\ \mathsf{pro}\ ((\Uparrow)\ \mathsf{mom})$
  $= \mathsf{do}\,\{z \leftarrow \mathsf{pro};\ f \leftarrow \mathsf{pure}\ \mathsf{mom};\ \mathsf{pure}\ (f\ z)\}$
  $= \mathsf{do}\,\{z \leftarrow \mathsf{pro};\ \mathsf{pure}\ (\mathsf{mom}\ z)\}$ （$\because$ 左単位則）
- $[{}_{>\mathsf{c}}\text{called}^{\Uparrow}\ [{}_{<\mathsf{c}}\text{their mother}^{\Uparrow}]]$ の意味表示 (s_2)
  $(>_{\mathsf{c}})\ ((\Uparrow)\ \mathsf{call})\ s_1$
  $= \mathsf{do}\,\{f \leftarrow \mathsf{pure}\ \mathsf{call};\ w \leftarrow s_1;$
  $\quad \mathsf{pure}\ (f\ w)\}$
  $= \mathsf{do}\,\{z \leftarrow \mathsf{pro};$
  $\quad \mathsf{pure}\ (\mathsf{call}\ (\mathsf{mom}\ z))\}$ （$\because$ 結合則, 左単位則）
- $[{}_{<\mathsf{c}}\text{everyone}^{\triangleright}\ [{}_{>\mathsf{c}}\text{called}^{\Uparrow}\ [{}_{<\mathsf{c}}\text{their mother}^{\Uparrow}]]]$ の意味表示 (s_3)
  $(<_{\mathsf{c}})\ ((\triangleright)\ \mathsf{ev})\ s_2$
  $= \mathsf{do}\,\{y \leftarrow \mathsf{do}\,\{x \leftarrow \mathsf{ev};\ \mathsf{bind}\ x\};\ f \leftarrow s_2;$
  $\quad \mathsf{pure}\ (f\ y)\}$
  $= \mathsf{do}\,\{x \leftarrow \mathsf{ev};\ y \leftarrow \mathsf{bind}\ x;\ z \leftarrow \mathsf{pro};$
  $\quad \mathsf{pure}\ (\mathsf{call}\ (\mathsf{mom}\ z)\ y)\}$ （$\because$ 結合則, 左単位則）
  $= \mathsf{do}\,\{x \leftarrow \mathsf{ev};\ \mathsf{pure}\ (\mathsf{call}\ (\mathsf{mom}\ x)\ x)\}$ （$\because$ 補題 $(*)$）
  $= \lambda k.\ (\lambda k.\ \forall x.\ k\ x)\ (\lambda x.\ (\lambda k.\ k\ (\mathsf{call}\ (\mathsf{mom}\ x)\ x))\ k)$
  $= \lambda k.\forall x.\ k\ (\mathsf{call}\ (\mathsf{mom}\ x)\ x)$
- $[{}_{\Downarrow}\ [{}_{<\mathsf{c}}\text{everyone}^{\triangleright}\ [{}_{>\mathsf{c}}\text{called}^{\Uparrow}\ [{}_{<\mathsf{c}}\text{their mother}^{\Uparrow}]]]]$ の意味表示
  $(\Downarrow)\ s_3 = (\lambda k.\forall x.\ k\ (\mathsf{call}\ (\mathsf{mom}\ x)\ x))\ (\lambda x.\ x)$
  $= \forall x.\ \mathsf{call}\ (\mathsf{mom}\ x)\ x$

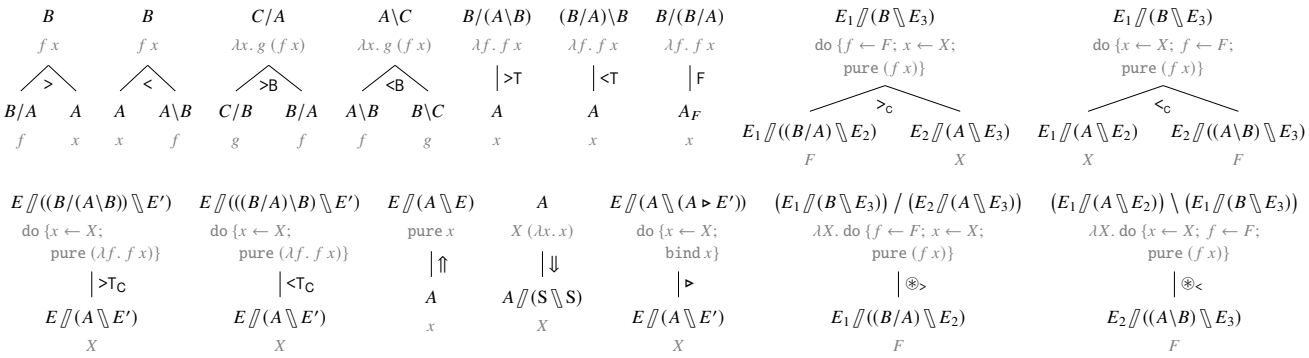


図 4　組合せ規則の一覧