214-OS-08a-3

限定継続によるフォーカスと逆スコープの分析

An Analysis of Focus and Inverse Scope by Delimited Continuations

叢悠悠[*1] 浅井健一[*1] 戸次大介[*1*2*3]
Yuyu So Kenichi Asai Daisuke Bekki

[*1]お茶の水女子大学大学院人間文化創成科学研究科
Ochanomizu University, Graduate School of Humanities and Sciences

[*2]国立情報学研究所
National Institute of Informatics

[*3]独立行政法人科学技術振興機構, CREST
CREST, Japan Science and Technology Agency

There exist several linguistic phenomena which require the surroundings of a relevant lexical item for the semantic representation of the whole sentence. The surrounding can be regarded as the "continuation" of the term. Continuations represent the rest of the computation. In this paper, we propose an analysis of focus and inverse scope by means of control operators `shift/reset` (Danvy and Filinski 1990), and show the implementation on OchaCaml (Masuko and Asai 2011). We also observe the interaction of these two phenomena.

1. はじめに

フォーカスや逆スコープ (inverse scope) といった言語現象においては，その意味表示を与える際に特定のフレーズを取り囲むコンテキストが必要になる．このコンテキストは，ある言語表現に対する「残りの計算」とみなすことができる．プログラミングにおいて，ある項に対する残りの計算は「継続」とよばれる．実際に，継続を用いてさまざまな言語現象を説明する研究がされ始めており，中でもフォーカスと逆スコープの意味を表現する研究としては Barker [1], Barker [2], Bekki and Asai [3] などがあるが，これらは複数のフォーカスや，三つ以上の量化表現を含む文に対して間違った予測を提示することがある．本研究では限定継続命令 `shift/reset` (Danvy and Filinski [4]) を用いた分析および OchaCaml (Masuko and Asai [6]) による実装を行い，既存研究が説明できなかった例文についても適切な意味表示を与えられることを示す．また，フォーカスと逆スコープをともに含む文についても OchaCaml で意味表示を記述し，両者の相互作用を観察する．

2. 継続

継続とは，ある時点における残りの計算を指す．たとえば，$1 + (2 * 3) - 4$ という式の $(2 * 3)$ の部分を計算しているときの継続は，$(2 * 3)$ が "hole" となった計算，すなわち「現在実行している部分の結果が返ってきたら，それに 1 を足し，4 を引く」というものになる．これは $(2 * 3)$ のまわりの計算となっている．

継続の中でも範囲の限られたものを限定継続という．本研究では限定継続を扱うための命令として `shift/reset` を採用する．`shift` は継続を切り取る命令，`reset` は `shift` が切り取る継続の範囲を限定する命令である．これらの命令を使用できる言語として OchaCaml がある．OchaCaml は Caml Light [5] に `shift/reset` を直接実装したものである．以下は OchaCaml のプログラムの一例である．

```
# 1 + reset (fun () ->
    2 * shift (fun k -> k (k 3))) ;;
~> 13
```

`shift` 命令によって，その時点の継続が `k` に束縛される．ここで捕捉される継続は，`shift` 節のまわりの計算のうち，`reset` で囲まれた範囲のものである．上のプログラムにおいて，継続 `k` は `fun x -> 2 * x` という計算となる．この `k` が 3 に二回適用され，最後に `reset` の外側にある 1 が足されることによって，結果は 13 となる．

本研究では OchaCaml の上で `shift/reset` を用いて文の意味表示を記述する．ここでは "John" や "Mary" といった名前を文字列，"love" のような述語や "every" のような量化表現を文字列をつなげる関数として定義する．たとえば，"John loves Mary." の意味表示は `love m j` という形で与えられ，それを簡約すると "love (j, m)" という文字列になる．

3. フォーカス

フォーカスとは，文中で重要となる部分のことである．たとえば，"Who does John love?" という質問に対し，"John loves Mary." という応答があった場合，質問をした側が知りたかった情報である "Mary" がフォーカスとなる．フォーカスを含む文を発話する際には，フォーカスの部分にピッチアクセントが置かれる傾向があるが，これはフォーカスが対比のニュアンスをもつことを意味する．したがって，フォーカスの意味を正しく捉えるためには，フォーカスされた言語表現を他の言語表現に置き換えた命題を考慮する必要がある (Rooth [8])．このような命題を元の命題に対する alternative とよぶ (ただし，元の命題も alternative の一つである)．先の応答の場合，フォーカスされた Mary を Sue や Alice など他の人物に置き換え，それぞれの命題が成立するか否かを考える．

ここで，先の応答に副詞 "only" および "also" を加えた文を考える．以下，"only" のフォーカスを $[\ ]_{F_o}$，"also" のフォーカスを $[\ ]_{F_a}$ で表す．

(1) John only loves $[\text{Mary}]_{F_o}$.
(John は Mary だけを愛している.)

連絡先: 叢悠悠，お茶の水女子大学大学院人間文化創成科学研究科 理学専攻 情報科学コース 戸次研究室，東京都文京区大塚 2-1-1，so.yuyu@is.ocha.ac.jp

(2) John also loves $[\text{Mary}]_{\text{F}_\text{a}}$.
（John は Mary も愛している.）

“only” はその命題を満たすものがフォーカスされた言語表現のみであることを意味する．(1) の場合，alternatives は “John loves x” という形の命題となるが，そのうち x に Mary を代入した命題のみが真であり，他の人物を代入したものはすべて偽となる．一方，“also” はその命題を満たすもので，フォーカス以外のものが存在することを前提とする．(2) で考えると，“John loves x“ という形の真の命題で，x に Mary 以外の人物を代入したものが alternatives のなかに存在することが前提とされる．これらを表現するためには，フォーカスのまわりの部分，すなわちフォーカスが hole になった命題が必要である．そして，“only” の場合は hole にフォーカスされた言語表現を代入したものは真，それ以外は偽とし，“also” の場合は hole に代入して成り立つもので，フォーカスされた言語表現ではないものが存在するとすればよい．前節で述べたように，`shift` 命令を使うことで `shift` 節のまわりの計算を切り取ることができる．したがって，フォーカスを `shift` 命令で表現し，“only” や “also” のスコープを `reset` 命令で囲むことで，フォーカスのまわりの部分を取り出すことができる．この考えに従うと，“only” のフォーカスに対する意味表示と “also” のフォーカスがもたらす前提を以下のように表現することができる (Bekki and Asai [3], 叢ら [9]).

$[M]_{\text{F}_\text{o}} \stackrel{\text{def}}{\equiv}$ `shift (fun k -> forall x (k x <-> x = M))`
$[M]_{\text{F}_\text{a}} \stackrel{\text{def}}{\equiv}$ `shift (fun k -> exists y (k y & not (y = M)))`

上の定義に従って (1) の意味表示を与え，OchaCaml で実行すると，以下のように簡約される（ただし，`j`, `m` はそれぞれ John, Mary の指示対象である）.

```
reset (fun () -> love
(shift (fun k -> forall x (k x <-> x = m))) j);;
⇝ "forall x (love (j, x) <-> x = m)"
```

簡約前の意味表示において，`k` は `fun x -> love x j` という関数となる．これはフォーカスされた “Mary” が hole となった命題を表している．簡約後の意味表示は，すべての x について,「John は x を愛している」という命題が真であるのは x が Mary であるとき，かつそのときのみであることを表す.

二つの副詞が一つの言語表現をフォーカス対象としている場合は，それぞれのフォーカスに対するコンテキストを区別する必要がある．そこで，二つの階層の限定継続命令を定義した．ここでは通常の `shift/reset` に対応する命令を `shift1/reset1` とする．`shift2/reset2` は `shift1/reset1` より階層が上の命令であり，`reset2` のスコープに `reset1` があった場合，`shift2` は `reset1` を越えて `reset2` までの継続を切り取ることができる．これらを使うと，(3) のような文に対して正しい意味表示を与えることができる.

(3) Sue has also thought that John only loves $[[\text{Mary}]_{\text{F}_\text{a}}]_{\text{F}_\text{o}}$.
（Sue は John が Mary だけを愛していると思っていたこともある.）

```
reset2 (fun () -> think
 reset1 (fun () -> love
  shift1 (fun k1 -> forall x (k1 x <-> x =
   shift2 (fun k2 -> exists y
    (k2 y & not (y = m))))) j) s) ;;
⇝ "exists y (think (s,
    forall x (love (j, x)
     <-> x = y)) & not (y = m))"
```

4. 逆スコープ

逆スコープとは，複数の量化表現を含む文において，後ろにある量化表現が上のスコープをとる読みが生じることを指す．たとえば，(4) には以下の二通りの読みが存在する.

(4) Some woman loves every man.
（ある女性がすべての男性を愛している.）
a. $\exists x\,(woman\,(x) \wedge \forall y\,(man\,(y) \rightarrow love\,(x,\,y)))$
（ある特定の女性がすべての男性を愛している.）
b. $\forall y\,(man\,(y) \rightarrow \exists x\,(woman\,(x) \wedge love\,(x,\,y)))$
（全ての男性について，彼を愛する女性が存在する.）

(4a) の意味表示では，“some” に対する存在量化子が “every” に対する全称量化子より上のスコープをとっている．このスコープ関係は文中における “some” と “every” の順序と一致しているため，順スコープ解釈という．一方，(4b) では量化子のスコープが文中の順序と逆転している．こちらの読みを逆スコープ解釈という.

生成文法の枠組みにおいては，論理形式 (Logical Form) を用いて文の意味的な構造を表す．May [7] は逆スコープ解釈に対応する論理形式を導出する際に，量化子繰り上げ (Quantifier Raising) とよばれる操作を行っている．量化子繰り上げでは，量化された名詞句がトレースを残して自身が含まれる最小の S ノードの先頭に移動する．文中に含まれるすべての量化名詞句に対して量化子繰り上げを適用すると，各量化表現のスコープ関係が明示化された論理形式が得られる．(4a) に対応する論理形式は，(4) に対して “every man”, “some woman” の順で量化子繰り上げを適用することで得られる.

$[_{\text{S}}\,[_{\text{NP}}\,\text{every man}]_3\,[_{\text{S}}\,[_{\text{NP}}\,\text{some woman}]_2\,[_{\text{S}}\,e_2\ \text{loves}\ e_3]]]$

e_2 は “some woman”, e_3 は “every man” に対するトレースである．量化名詞句が移動したことによって，二つ目の S ノードと三つ目の S ノードに “every man” と “some woman” に対するコンテキストが明示される．論理形式においては，自身のコンテキストに他の量化表現を含んでいるものが上のスコープをとるため，この場合は “every man” が上の読みとなる．順スコープ解釈に関しては，文の表層構造から単純に意味表示を組み立てることができるが，逆スコープ解釈の場合は，何らかの手段で上のスコープをとる量化表現に対するコンテキストを明示する必要がある．このコンテキストは `shift` 命令で取り出すことができる (叢ら [11]).

本研究では，各量化表現について，主語，目的語，および前置詞句に含まれるものをそれぞれ定義した．以下は “every” の例である．`n` は “man” などの一項述語，`p` は “run” や “loves Mary” のような動詞句である．`s` と `o` はそれぞれ主語と目的語の個体を表す.

```
every n p = "forall x ((n x) -> (p x))"
every_acc n p s = "forall x ((n x) -> (p x s))"
every_pp n p o s = "forall x ((n x) -> (p o x s))"
```

そのうえで，逆スコープ解釈を導出するための INV オペレータを次のように定義した.

$[f]_{\mathrm{INV}} \stackrel{\mathrm{def}}{\equiv}$ `shift (fun k -> f k)`

INV オペレータは主語の型 $((e \to t) \to t)$ をもつ量化名詞句 `f` を引数として受け取る．`shift` 命令によって，`f` のまわりのコンテキストが `k` として取り出される．このコンテキストを `f` に渡すことで，`f` に含まれる量化表現が上のスコープをとる表示に簡約される．なお，`shift` 命令が含まれるため，実行する際には意味表示全体を `reset` で囲む必要がある．

(4b) の意味表示は “every man” に対して INV オペレータを適用することで導出できる．

```
reset (fun () ->
 some woman (love (inv (every man)))) ;;
⇝ "forall x (man (x) ->
   exists y (woman (y) & love (y, x)))"
```

一つの文が三つの量化表現 Q_1, Q_2, Q_3 を含む場合，可能な逆スコープ解釈は $Q_2 > Q_1 > Q_3$ と $Q_3 > Q_1 > Q_2$ の二通りに限られると筆者らは考える．これらの読みはそれぞれ Q_2 と Q_3 に INV オペレータを適用することで導出することができる．以下がその例である．

(5) Some teachers introduced [most students]$_{\mathrm{INV}}$ to every company. ($most > some > every$)

```
reset (fun () ->
 some teacher
 (every_pp company
  introduce (inv (most student)))) ;;
⇝ "most z (student (z),
   exists y (teacher (y) &
    forall x (company (x) ->
     introduce (y, z, x))))"
```

ただし，Q_2 と Q_3 の両方に適用すると $Q_3 > Q_2 > Q_1$ という存在しない読みが導出されるため，一つの文の中で INV オペレータを使用できるのは一回のみとする．

5. フォーカスと逆スコープの相互作用

本節では，“only” のフォーカスと複数の量化表現をともに含む文について考察する．さまざまな例文に対して，逆スコープ解釈に対応する意味表示を OchaCaml で実行し，簡約結果が表現する読みが存在するか否かを検証する．

なお，簡単のため，“only” に対するフォーカス $[M]_{\mathrm{F_o}}$ を `f_o M` と表すことにする．これを展開すると `shift (fun k -> forall x (k x <-> x = M))` となる．また，ここではネストしていないフォーカスのみを考える．

具体的な例文をみてみよう．まず，フォーカスの対象となる言語表現と逆スコープをとる言語表現が別々の場合を考える．

(6) Some woman only introduced [John]$_{\mathrm{F_o}}$ to [every man]$_{\mathrm{INV}}$.

```
reset (fun () ->
 some woman
 (introduce (f_o j) (inv (every man)))) ;;
⇝ "forall x (man (x) ->
   forall y (exists z (woman (z) &
    introduce (z, y, x)) <-> y = j))"
```

OchaCaml では引数が右から実行される．そのため，INV オペレータが適用されている “every man” が最も上のスコープをとる表示に簡約される．この表示は「すべての男性について，彼に John だけを紹介した女性が存在する」という読みを表現しているが，これは (6) がもつ可能な読みの一つであると考える．

フォーカスが逆スコープをとる言語表現の後ろにある場合は，フォーカスの意味表示に含まれる量化表現が最も上のスコープをとる表示となる．

(7) Some woman only introduced [every man]$_{\mathrm{INV}}$ to [John]$_{\mathrm{F_o}}$.

```
reset (fun () ->
 some woman
  (introduce (inv (every man)) (f_o j))) ;;
⇝ "forall x (forall y (man (y) ->
    exists z (woman (z) &
     introduce (z, y, x))) <-> x = j)"
```

この場合はフォーカスされた “John” の意味表示に含まれる全称量化が最も上のスコープをとる．簡約された意味表示は「すべての男性について，彼に John だけを紹介した女性が存在する」という読みを表現している．これは，それぞれの男性に対して John を紹介した女性が異なるという状況を許すが，(7) はこのような状況を表しているだろうか．筆者らの考えとしては，この文の可能な読みは “some woman” が上のスコープをとる順スコープ解釈のみであり，“every man” が上のスコープをとる読みに解釈することは不可能である．INV オペレータによってある量化表現が主語位置の量化表現より上のスコープをとることを防ぐためには，“only” のスコープを主語位置の量化名詞句に渡す動詞句に限定すれば良い．(7) の場合は，動詞 “introduce” の前に `reset` を設けることで，“every man” が上のスコープをとらない読みが導出される．

```
some woman
 (fun x -> reset (fun () ->
  introduce (inv (every man)) (f_o j) x)) ;;
⇝ "exists x (woman (x) &
   forall y (forall z (man (z) ->
    introduce (x, z, y)) <-> y = j))"
```

一般に，“only” とそのフォーカスの間に量化表現が含まれる場合，その量化表現は上のスコープをとることができないと考えられる．これは，“only” とフォーカスを結びつけながら，二つの量化表現のスコープを逆転させることが人間の言語処理能力の限界を越えていることによるのかもしれない．

次に，逆スコープをとる量化名詞句がフォーカスされている場合を考える．

(8) Some teachers only introduced [[most students]$_{\mathrm{INV}}$]$_{\mathrm{F_o}}$ to every company.

```
reset (fun () ->
 some teacher
 (every_pp company
  introduce (f_o (inv (most student))))) ;;
⇝ "most x (student (x),
   forall y (exists z (teacher (z) &
    forall w (company (w) ->
     introduce (z, y, w))) <-> y = x))"
```

introduce の第一引数のなかで，まず f_o が呼び出され，“only” のフォーカス対象として inv (most student) が渡される．shift が切り取る継続には，それを取り囲む reset が新たに設けられるため，この INV オペレータが切り取る継続は f_o までの計算となる．これが inv の計算によって most student に渡されるため，結果として “most” が最も上のスコープをとる表示に簡約される．簡約後の意味表示は「ほとんどの学生について，彼（彼女）だけを全ての会社に紹介した先生が存在する」という読みを表している．直感的に，(8) はこのような読みをもたないように思える．筆者らは，フォーカスされた量化名詞句が逆スコープをとることは難しいと考える．(8) において，“most” が “exists” より上のスコープをとらない読みは，some の第二引数に reset を設けることで導出できる．

```
reset (fun () ->
 some teacher
 (fun t -> reset (fun () ->
  every_pp company
  introduce (f_o (inv (most student))) t))) ;;
⇝ "exists x (teacher (x) &
    most y (student (y),
     forall z (forall w (company (w) ->
      introduce (x, z, w)) <-> z = y)))"
```

上の簡約結果が示す読みは「ほとんどの学生だけを全ての会社に紹介した先生が存在する」というものである．これは (8) がもつ読みであると考えられる．このように，“only” のスコープを制限すると，INV オペレータが適用されている量化表現が “only” の reset にブロックされて上のスコープをとることができなくなる．

6. おわりに

本研究では，フォーカスを含む文と逆スコープ解釈の意味表示を shift/reset で記述した．特に，フォーカスと逆スコープをともに含む文について意味表示を与え，OchaCaml による簡約結果の表現する読みが存在するかどうかを判断した．その結果，副詞とフォーカスの間に量化表現が含まれる場合や，量化名詞句が同時にフォーカスされている場合は逆スコープ読みに解釈することが難しく，OchaCaml による簡約結果の示す読みが存在しない場合があることが分かった．このようなケースにおいては，“only” のスコープを文全体ではなく，主語の量化名詞句が受け取る動詞句に制限することで自然な読みが得られた．

今回はフォーカスが一つの場合のみを考えたが，今後は複数のフォーカスやネストしたフォーカスが含まれる文についても考察を行う予定である．また，現在はフォーカスの意味表示に副詞のもつ意味を含ませているため，副詞によってフォーカスの意味表示が異なっている．今後はフォーカスを統一的に扱い，副詞の意味表示をそれぞれ違う形で定義する方法を考えたいと思う．

参考文献

[1] Barker, C.: Continuations and the Nature of Quantification, *Natural Language Semantics* 10(3), pp. 211–241 (2002).

[2] Barker, C.: Continuations in Natural Language, *the Fourth ACM SIGPLAN Continuations Workshop (CW'04). Technical Report CSR-04-1, School of Computer Science, University of Birmingham, Birmingham B152TT*, pp. 1–11 (2004).

[3] Bekki, D. and K. Asai: Representing Covert Movements by Delimited Continuations, In: K. Nakakoji, Y. Murakami, and E. McCready (eds.): *New Frontiers in Artificial Intelligence (JSAI-isAI 2009 Workshops, Tokyo, Japan, November 2009, Selected Papers from LENLS 6)*, Vol. LNAI 6284, pp. 161–180 (2010).

[4] Danvy, O. and Filinski, A.: Abstracting Control, In: *LFP90, the 1990 ACM Conference on Lisp and Functional Programming*, pp. 151–160 (1990).

[5] Leroy, X.: *The Caml Light system release 0.74* (1997).

[6] Masuko, M. and K. Asai: Caml Light + shift/reset = Caml Shift, Theory and Practice of Delimited Continuations (TPDC 2011), pp. 33–46 (2011).

[7] May, R.: *The Grammar of Quantification*, Doctoral dissertation, MIT, Cambridge (1977).

[8] Rooth, M.: Focus, *The Handbook of Contemporary Semantic Theory*, pp.271–298 (1997).

[9] 叢悠悠, 浅井健一, 戸次大介: 限定継続を用いたフォーカスの分析と実装に向けて, 情報処理学会第 214 回自然言語処理研究会 (2013).

[10] 叢悠悠, 浅井健一, 戸次大介: 限定継続を用いたフォーカスの分析と実装, 第 16 回プログラミングおよびプログラミング言語ワークショップ論文集 (2014).

[11] 叢悠悠, 浅井健一, 戸次大介: 限定継続を用いた inverse scope の分析と実装, 言語処理学会第 20 回年次大会発表論文集掲載予定 (2014).