限定継続を用いた inverse scope の分析と実装

叢 悠悠 [1]　浅井 健一 [2]　戸次 大介 [2,3,4]

[1] お茶の水女子大学

[2] お茶の水女子大学大学院人間文化創成科学研究科

[3] 国立情報学研究所

[4] 独立行政法人科学技術振興機構, CREST

{g1020519, asai, bekki}@is.ocha.ac.jp

1　はじめに

Inverse scope とよばれる言語現象は，複数の量化表現を含む文において，それらのスコープ関係が表層的な順序と逆転する読みが生じることを指す．生成文法の枠組みにおいて，inverse scope は非顕在的移動を必要とする現象とされてきた (May [5, 6])[1]．

非顕在的移動を起こす目的は，ある語彙項目のまわりのコンテキストを明示化することにある．このコンテキストは，プログラミングにおける「継続」とみなすことができる (Barker [1], Bekki and Asai [2])．本研究では，限定継続命令 `shift/reset` (Danvy and Filinski [3]) を用いて inverse scope reading の意味表示を与える．また，OchaCaml (Masuko and Asai [4]) を用いてそれらの意味表示がどのように簡約されるかを観察する．

本稿では以下，第 2 節で生成文法における inverse scope の分析を概説する，第 3 節では継続を用いた inverse scope の先行研究を取り上げ，第 4 節では本研究における inverse scope の分析とその実装について述べる．

2　Inverse scope

every や *some*, *more than two* といった量化表現を複数含む文は，量化のスコープ関係が文中の順序と一致した surface scope reading と，順序が逆転した inverse scope reading をともにもつことがある．たとえば，“Some woman loves every man” という文には以下の 2 通りの読みが存在する．

(1) Some woman loves every man.
（ある女性が全ての男性を愛している.）

a. $\exists x\,(woman\,(x) \wedge \forall y\,(man\,(y) \rightarrow love\,(x,\,y)))$
（ある特定の女性が全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.）

(1a) の読みは *some* と *every* のスコープ関係が文中の順序と一致しており，$\exists x$ が $\forall y$ より上の意味表示となる．一方，(1b) では *every* のスコープが *some* より上になっている．これが (1) に対する inverse scope reading である．

生成文法では，論理形式 (Logical Form) を用いて文の意味的な構造を表現する．文の表層的な構造から論理形式に変換する際には，ある語彙項目が移動 (movement) を起こすことがある．このような移動は Wh-疑問文における Wh 節の移動などと異なり，文の音声形式に影響しないため，非顕在的移動 (covert movement) とよばれる．May [5, 6] は，surface/inverse scope reading に対応する論理形式を導出する際に，非顕在的移動の一種である量化子繰り上げ (Quantifier Raising) を行っている．量化子繰り上げを行うと，量化された名詞句全体がスコープ内にトレースを残し，自身が含まれる最小の S ノードの付加部に移動する．例として，文 (1) に対して量化子繰り上げを適用してみよう．

(2) [$_{S}$ [$_{NP}$ some woman] [$_{VP}$ loves [$_{NP}$ every man]]]

[1] 一方，範疇文法では各語彙項目の結合順序の選択によって inverse scope に対応する読みを導出できる (Steedman [8])．たとえば，主語と動詞を先に結合させると，目的語位置の量化表現が主語位置の量化表現より上のスコープを取ることが可能になるが，この手法は日本語のように主語，目的語，動詞という語順をもつ言語に応用することができない．

a. $[_{S}\,[_{NP}\,\text{some woman}]_2\;[_{S}\,e_2\;[_{VP}\;\text{loves}\;[_{NP}\,\text{every man}]]]]$

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

c. $[_{S}\,[_{NP}\,\text{every man}]_3\,[_{S}\,[_{NP}\,\text{some woman}]\;[_{VP}\,\text{loves}\;e_3]]]$

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

(2) は (1) の表層構造を表したものである．*some woman* に対して量化子繰り上げを行うと，(2a) のような表示になる．(2a) の 2 つ目の S ノードは *some woman* のトレース e_2 と (2) の VP ノードから構成されているが，これは *some woman* に対するコンテキストとなっている．このコンテキストに含まれるもう一つの量化表現 *every man* にも量化子繰り上げを適用すると，*every man* がトレース e_3 を残して自身を含む最小の S ノード，すなわち (2a) の 2 つ目の S ノードの付加部に移動する．これにより (2b) が得られ，3 つ目の S ノードに *every man* に対するコンテキストが明示される．文の先頭から *some woman*, *every man* の順に量化子繰り上げを行うことで得られた (2b) は，surface scope reading に対応する論理形式となっている．一方，目的語位置にある *every man* を先に移動させて (2c) の形にし，その後 *some woman* を移動させると，inverse scope reading に対応する (2d) が得られる．(2c) の 2 つ目の S ノードは *every man* に対するコンテキストであり，(2d) の 3 つ目の S ノードは *some woman* に対するコンテキストである．May [5] は，このように 1 つの文がもつ複数の読みを異なる論理形式で表現している[2]．論理形式の中で，他の量化表現を自身のコンテキストに含んでいるものは上のスコープをとる．

3 継続を用いた分析

継続とは，ある時点における残りの計算のことを指す．たとえば，$1 + (2 * 3)$ という計算の $2 * 3$ の部分を実行しているときの継続は，「現在実行している部分の結果に 1 を足す」という計算になる．この計算は $\lambda x.\,(1 + x)$ という関数で表すことができる．

[2](2c) は移動した主語とそのトレースを含む句の間に他の句が入っており，空範疇原理 (Empty Category Principle) に違反している．そのため，May [6] は 1 つの論理形式 （(2) の場合は (2d)）に対する複数の解釈によってスコープの曖昧性を表現できるとしている．

3.1 CPS 変換

計算の中で継続を使いたい場合は，プログラムを継続渡し形式 (Continuation-Passing Style, CPS) に変換して継続を明示化する必要がある．継続渡し形式のプログラムでは，それぞれの関数に継続のための引数が加わり，継続にそれまでの計算結果が渡される．以下に Plotkin [7] による CPS 変換規則を示す．

$$\begin{aligned}
[\![x]\!] &= \lambda k.\,k\,x \\
[\![\lambda x.\,M]\!] &= \lambda k.\,k\,(\lambda x.\,[\![M]\!]) \\
[\![M\,N]\!] &= \lambda k.\,[\![M]\!](\lambda m.\,[\![N]\!](\lambda n.\,(m\,n)\,k))
\end{aligned}$$

継続が明示されることによって，ユーザによる実行順序の制御が可能になる．たとえば，上に示した CPS 変換規則において，3 つ目の関数適用のケースは左側の M を先に計算し，その後右側の N を計算するとしているが，これを $\lambda k.\,[\![N]\!](\lambda n.\,[\![M]\!](\lambda m.\,(m\,n)\,k))$ に書き換えることで，右から計算するように変更できる．

3.2 Barker [1] の分析

Barker [1] は，継続を取り入れた文法 ”continuized grammar” の枠組みで inverse scope の分析を行っている．Continuized grammar では，各語彙項目が CPS 変換されることによって，自身に対する継続が明示される．これらを使って文を構成する際に，左右どちらから計算するかによって，surface scope reading と inverse scope reading が導出できる．たとえば，量化表現を含む主語と動詞句から文を構成することを考えたとき，そのプロセスは 2 通り存在する．

(3) a. $S \to NP\ VP$
$\lambda k.\,[\![NP]\!](\lambda x.\,[\![VP]\!](\lambda P.\,k\,(P\,x)))$

b. $S \to NP\ VP$
$\lambda k.\,[\![VP]\!](\lambda P.\,[\![NP]\!](\lambda x.\,k\,(P\,x)))$

(3a) は関数適用に対する CPS 変換規則のうち左から実行するものに対応しており，主語が先に計算される．主語の継続 $\lambda x.\,[\![VP]\!](\lambda P.\,k\,(P\,x))$ に動詞句の量化表現が含まれるため，先に出現する量化表現が上のスコープをとる意味表示，すなわち surface scope reading に対応する表示が得られる．一方，(3b) は右から実行する CPS 変換規則に対応しており，動詞句から計算される．よって，後ろにある量化表現が上の読み，つまり inverse scope reading が得られる．

3.3 Barker [1] の問題点

Bekki and Asai [2] は，Barker [1] の分析について，以下の例文を用いて批判している．

(4) Some teachers introduce most students to every company.

(3) は *some, most, every* の 3 つの量化表現を含む．Bekki らによると，3 つの量化表現 Q_1, Q_2, Q_3 を含む文において，量化のスコープが逆転した $Q_3 > Q_2 > Q_1$ という読み（本稿では reversed reading とよぶ）はあまりみられない．しかし，Barker の定義に従って，右から実行する CPS 変換規則を採用した場合，このような読みが導出できてしまう．

一方で，Bekki らは主語の量化表現が他の 2 つの量化表現の間に入る *most* > *some* > *every* や *every* > *some* > *most* という読み（本稿では intermediate-inverse scope reading とよぶ）は可能であるとしている．しかし，Barker の分析ではこれらの読みを導出することができない．Barker に従うと，主語と動詞句および目的語と前置詞句それぞれの実行順序に関して，左あるいは右から計算するという選択肢があるが，これは主語の量化表現が他の 2 つの両方に対して上あるいは下のスコープを取ることしかできないことを意味する．つまり，以下の 4 つの読みのみが導出される．

some > *most* > *every*（全て左から）
most > *every* > *some*（VP から，VP 内は左から）
some > *every* > *most*（主語から，VP 内は右から）
every > *most* > *some*（全て右から）

4 限定継続による分析と実装

本研究では，限定継続命令 `shift/reset` (Danvy and Filinski [3]) を用いて inverse scope の意味表示を与える．

4.1 `shift/reset`

継続のうち，範囲の限られたもののことを限定継続とよぶ．`shift/reset` は限定継続を扱うためのオペレータであり，`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定める命令である．例として $1 + \mathit{reset}\,(2 * \mathit{shift}\,k.\,(3 + k\ 4))$ という計算を考えると，`shift` によって切り取られる継続 k は，`shift` のまわりの計算のうち，`reset` で囲まれた範囲のもの，つまり $\lambda x.\,(2 * x)$ という関数になる．この式を簡約すると 12 になる．

4.2 INV オペレータ

本稿では，inverse scope reading を導出するための INV オペレータを以下のように定義する．

$$\frac{f : (e \to t) \to t}{\mathit{shift}\,k.\,(f\ k) : e}\,(INV)$$

INV オペレータは量化名詞句 f を引数として受け取る．`shift` によって k に束縛される継続は，f の周りのコンテキストである．このコンテキストが f に渡されることで，f に含まれる量化表現が上のスコープをとる表示となる．`shift` オペレータが含まれるため，実行する際は式全体を `reset` で囲む必要がある．

4.3 OchaCaml による実装

OchaCaml は `shift/reset` を使うことのできる ML 系言語である．本研究では，OchaCaml 上で `shift/reset` を使って複数の量化表現を含む文の意味表示を与え，それらがどのように簡約されるかを観察した．

意味表示に必要な述語や量化表現は仮に文字列をつなげる関数として定義した．たとえば，2 項述語 *love* は目的語 y と主語 x を受け取ったら，文字列 "$\mathit{love}\,(x,\ y)$" を返す．量化表現は主語と目的語，および前置詞句に含まれるものをそれぞれ以下のように定義している．

$\mathit{every}\ n\ p = \forall x\,(n\ x \to p\ x)$
$\mathit{every}_{acc}\ n\ p\ s = \forall x\,(n\ x \to p\ x\ s)$
$\mathit{every}_{pp}\ n\ p\ o\ s = \forall x\,(n\ x \to p\ o\ x\ s)$

n は *man*, *student* といった $e \to t$ 型の述語，p は *runs*, *loves Mary* のような $e \to t$ 型の動詞句である．s と o はそれぞれ主語と目的語の個体である．

これらを用いて (1) の 2 つの読みに対応する意味表示を与えると，以下のように簡約される[3]．

(1a) Some woman loves every man.

```
some woman (every_acc man love) ;;
"exists y (woman (y) &
  forall x (man (x) -> love (y, x)))"
```

[3] OchaCaml では reset 命令を `reset (fun () -> )` のように書く．

(1b) Some woman loves [every man]$_{\mathrm{INV}}$.

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

INV オペレータを使うと，一つ一つの量化表現に対して実行順序を指定することができる．このことによって，(3) の 2 つの intermediate-inverse scope reading も以下のように導出される．

(4a) 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))))"
```

(4b) Some teachers introduced most students to [every company]$_{\mathrm{INV}}$. (*every* > *some* > *most*)

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

(4a) では *most students* に対してのみ INV オペレータを適用している．このことによって，*most* のみが *some* より上のスコープをとる意味表示が得られる．同様に，(4b) では *every company* にのみ INV オペレータを適用しており，簡約すると *every* が先頭に移動した意味表示となる．

4.4 問題点

INV オペレータを定義したことで intermediate-inverse scope reading が導出されるようになったが，2 つ目および 3 つ目の量化表現の両方に対して INV オペレータを適用すると，reversed reading に対応する意味表示が得られてしまう．たとえば，(4) において *most students* と *every company* の両者に INV オペレータを適用した場合，*most* > *every* > *some* ではなく *every* > *most* > *some* という reversed reading となる（これは OchaCaml が引数が右から実行することによる）．Reversed reading が存在しないという立場をとるならば，このような読みを導出不可能にするような制約を設ける必要がある．現時点では，「主語位置以外の量化表現すべてに INV オペレータを適用してはならない」という制約を加えることにする．

5 おわりに

本研究では，限定継続命令 `shift/reset` を用いて inverse scope reading の意味表示を与えた．INV オペレータを定義したことによってそれぞれの量化表現の実行順序を指定できるようになり，Barker の分析が導出できなかった intermediate-inverse scope reading が導出可能になった．

なお，筆者らはフォーカスとよばれる言語現象についても `shift/reset` を用いた分析を行っている．今後，inverse scope とフォーカスをともに含む文における両者の相互作用を観察したいと考えている．

参考文献

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

[2] 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).

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

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

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

[6] May, R.: *Logical Form: Its Structure and Derivation*, MIT Press, Cambridge (1985).

[7] Plotkin, G. D.: Call-by-Name, Call-by-Value and Lambda Calculus, *Theoretical Computer Science* 1, pp. 125–159 (1975).

[8] Steedman, M.: *The Syntactic Process (Language, Speech, and Communication)*, MIT Press, Cambridge (2001).