カットオペレータの日本語による読み下し方式

3E-10

小谷 善行
東京農工大学工学部数理情報工学科

Prologを学生に教えるとき、教えにくいところのひとつがカットオペレータである。これを日本語でわかりやすく読み下すことを試みている[1]。Prologのプログラムの説明には論理的説明と手順的説明の２通りがある。論理的説明によれば理解は著しく速い。しかしカットオペレータがあると、その部分は手順的説明をするのが普通であり、それが教えにくい点であった。

本稿ではカットオペレータを含むPrologプログラムを、なるべく適切に日本語で読み下すにはどうすればよいかついて論じる。この結果は次の点に寄与すると考える。（１）Prologプログラムの説明の多くの部分を論理的説明ですますことができる。（２）日本語Prologの言語仕様の設計の知見を得る。

はじめに

日本人は、外国語の文をそのまま自国語で読み下す、漢文という世界に類を見ない不思議な技法を作り出した。これは自国の文化を破壊せず、多くの情報を取り込むのに意義があったかもしれない。このように自国語で読み下すということは、対象の理解のための親和性に関して大変よい力となろう。Prolog言語についても日本語で自然に読み下せるとしたら、学習者のさまざまな障壁が取り除かれるだろう。

導出原理をもとにして構成された言語であるPrologは、その中心となる部分は「純Prolog」と呼ばれる。この部分は述語論理に対応しているので、論理的に読み下すのは容易である。そしてその読み下しはそのまま「論理的説明」になっている。

一方、Prologをプログラム言語として成立させるために、純Prologから逸脱する部分が付加された。その部分としては、（１）カットオペレータ、（２）節の動的追加・削除（assert,retract）、（３）その他の組み込み述語（入出力など）がある。

もし、これらを読み下し、論理的説明に組み込むことができれば、意義が大きい。ここではカットオペレータの読み下しを論じる。

カットオペレータの読み下し方

ここに一つの案を示す。すなわち

<定義節>::=<胴体><頭部>
<胴体>::={<ﾘﾃﾗﾙ>...なら}
{<ﾘﾃﾗﾙ>...{{そ}のとき}だけ}

とする。ここで、上記の胴体の第２行があるとき、カットオペレータがその先頭（つまり、「なら」の後）にある、とする。つまり、カットオペレータから後のリテラル列を「なら」と「だけ」で囲む。

以下が通常のPrologとを読み下した例である。

	通常のProlog	日本語による読み下し
1	p：－q，r．	q、rならp。
2	p：－!，q，r．	q、rのときだけp。
3	p：－q，!，r．	qならrのときだけp
4	p：－q，r，!．	q、rなら そのときだけp。

このおのおのの節の後に「p：－s．」があったとする。そうすると、この読み下しのなかの「だけ」の効果により、この節はまったく自然に括弧内を付加して読むことができる。

１． （また）sなら（であっても）p．
２． （これはもう無意味：）sならp．
３． （qでないなら）sならp．
４． （q，rでないなら）sならp．

引数を含む定義節の例

ほとんどの場合、述語は引数を持っている。引数があるとすると、上記のまま読み下すと、意味の不適合が起きる。次の例で比較する。

太郎が飲む（X）：－甘い（X），酒（X）．
次郎が飲む（X）：－!，甘い（X），酒（X）．
三郎が飲む（X）：－甘い（X），!，酒（X）．
四郎が飲む（X）：－甘い（X），酒（X），!．
（定義１）

すなわちこれをそのまま読むと、

すべてのXに対して、Xが甘くてXが酒であるなら、太郎はXを飲む。
すべてのXに対して、Xが甘くてXが酒であるときだけ、次郎はXを飲む。
すべてのXに対して、Xが甘いなら、Xが酒であるときだけ、三郎はXを飲む。
すべてのXに対して、Xが甘くてXが酒であるなら、そのときだけ四郎はXを飲む。 （１）

となる。この場合次のような問題点が出る。

（ア）「すべてのXに対して」が本体でなく条件の部分に掛かるように誤解する危険がある（これはカットオペレータがなくても同じである）。
（イ）「だけ」による限定が「Xを」にも及ぶために、意味が曖昧になる。「飲む対象の決定」つまりXへのユニフィケーションは最初に起きる。

このことを考慮し、

（ア）「あるものを・・・なら、それを・・・」という形で変数を読む。
（イ）限定が掛からないようする。意味は括弧で補足する。

How to Read Cut Operator in Japanese

Yoshiyuki KOTANI
Tokyo University of Agriculture and Technology

とすると、（２）のように読むことができる。

あるものが甘くてそれが酒であるなら、太郎は飲む（飲むのはそれ）。
あるものが甘くてそれが酒であるときだけ、次郎は飲む（飲むのはそれ）。
あるものが甘いなら、それが酒であるときだけ、三郎は飲む（飲むのはそれ）。
あるものが甘くてそれが酒であるなら、そのときだけ四郎は飲む（飲むのはそれ）。（２）

さらに「あるものを」のかわりに「・・・がある」ということにするとより、こなれた表現になる。

甘い酒があるなら、太郎は飲む　（飲むのはそれ）。
甘い酒があるときだけ、次郎は飲む（飲むのはそれ）。
甘いものがあるなら、それが酒であるときだけ、三郎は飲む（飲むのはそれ）。
甘い酒があるなら、そのときだけ四郎は飲む（飲むのはそれ）。　（３）

また、関数型（述語を満たすものを項として入れ子にする、文献[3]）で表現すると、（４）のようになる。このとき「だけ」は対応する項に付ければよい。なお、ここでこのままでは、定数と条件部が入れ子になったものとの読み方が区別できないので注意を要する。

甘い酒を、太郎は飲む。
甘い酒だけを、次郎は飲む。
甘いものがあるなら、そのうち酒だけを、三郎は飲む。
甘い酒があるなら、それだけを四郎は飲む。　（４）

この（４）の考えは「頭部の引数に定数があるとき」にも用いることができる。たとえば、

飲む（ビール）：－暑い，！．
飲む（ワイン）．

は、「暑ければ、ビールだけを飲む。（暑くなければ）ワインも飲む」と読める。カットオペレータがなければ「暑ければビールを飲む。ワインは（いつも）飲む」と読める。

読みの正当性

定義１の続きに、次の定義があるとする。

太郎が飲む（Ｘ）：－暖かい（Ｘ）．
次郎が飲む（Ｘ）：－暖かい（Ｘ）．　・・・
甘い（梅酒）．　　酒（梅酒）．
暖かい（お茶）．
甘い（ココア）．　　暖かい（ココア）．

このときの質問の成功失敗のパターンを示す。

質問　＼　引数	お茶	ｺｺｱ	変数X(ﾏｯﾁするもの)
?-太郎が飲む(引数)	○	○	○(梅酒､お茶､ｺｺｱ)
?-次郎が飲む(引数)	×	×	×
?-三郎が飲む(引数)	○	×	×
?-四郎が飲む(引数)	○	○	○(梅酒だけ)

これをみると（３），（４）の読みと処理の結果が一致しているのがわかる。「お茶」や「ココア」の定数による質問は、それぞれのものが「ひとつだけある」場合の意味に対応している。たとえば、四郎は甘い酒がない場合はココアを飲む。問い合わせが変数を使っているとき、「すべてのものがある」場合についての答えを出すことに対応している。つまり四郎は甘い酒である梅酒が存在するので、それ以外は飲まない（マッチしない）。逆に質問「？－四郎が飲む（Ｘ）．」は、「なんでもあるとすると四郎はなにを飲むか」と読むとよいと思われる。

ｆａｉｌを含む定義節

カットオペレータがなくて最後にｆａｉｌがある場合、その条件節のリテラルが純Prologから逸脱していなければ意味はない。例えば「ｐ：－ｒ，ｆａｉｌ．」という節があったとき、ｒは純Prologからはずれているはずである。これは「ｒを（すべて）行っておく」と読む位にしておく（ここでは深く論じない）。

カットオペレータの後にｆａｉｌがある場合、「・・・なら・・・ではない」と読むことにする。例えば、

ｐ：－ｑ，！，ｒ，ｆａｉｌ．

は、「ｑなら、ｐではない（ｒを行っておく）」となる。

飲む（Ｘ）：－酒（Ｘ），！，ｆａｉｌ．

は上記と同様、「あるものが酒なら、それを飲まない」とか「酒を飲まない」とか読める。

なお、以上のカットオペレータやｆａｉｌのパターンで普通のプログラムで出現する形はすべて尽くされるであろう。例えば、カットオペレータが条件節のなかに２つあるような使い方は普通はしない。

おわりに

Prolog言語の文を自然な日本語で読み下すことを、カットオペレータを中心にして論じた。この読み下しはそのまま教育の場で使っている。一方この考え方を逆向きに使い、日本語Prologを設計している（文献[2]）。

本稿をもとにして今後解決すべき課題を次に挙げる。

（１）Prologのｎｏｔを意味どおりに読むこと（変数の引数で「！，ｆａｉｌ」のある節を起動する）。

（２）日本語の「は」と「が」の違いは明確なものであるが、ここでの読みはこれを利用していない。このことをうまく取り入れたい。

（３）組み込み述語を始めとする、純Prologからはずれる部分の読み方。

文献

(1)小谷､知識指向言語Prolog､技術評論社､1986。
(2)小谷､LOGOの｢ﾋｭｰﾏﾝ･ﾌﾚﾝﾄﾞﾘｰ｣性とその日本語prologへの応用､ﾌﾟﾛｸﾞﾗﾐﾝｸﾞｼﾝﾎﾟｼﾞｳﾑ･夏のｼﾝﾎﾟｼﾞｳﾑ｢ﾋｭｰﾏﾝ･ﾌﾚﾝﾄﾞﾘｰなｼｽﾃﾑ｣､情報処理学会､1986。
(3)小谷、変数記述を減らしたPROLOGの仕様､情報処理学会第29回大会､1984。