柔軟な計算機コマンド記述の枠組の提案

6R－5

古宇田フミ子　近山 隆
{fumiko, chik}@logos.t.u-tokyo.ac.jp
東京大学工学部*

1　はじめに

計算機コマンドは機種に固有な名称を知っていなければ使えない。固有な名前を知っていれば使えるか、というと、その名前で表される処理の内容を正しく理解していないと、希望の処理結果が得られないことも起こる。

この問題は、コマンドの表現形態には依らない。例えば、アイコンの場合、そのアイコンがどんな機能を持っているかを知らなければ正しく選択できない。

この問題の解決法の一つとして、目的のものを直接指示するのではなく、間接的に、機能や属性を説明(以下、記述名と呼ぶ)して問い合わせるような形で、必要なコマンドを得る(必ずしも実行する訳ではない)ことが考えられる。ところが、このような記述では利用者の意図、視点、知識量などにより同じことを説明する場合であっても記述の仕方が種々に異なることが予想される。

この論文では利用者が計算機にやらせたいことを自由に記述し、それを解決して対応するコマンド名(の組合せ)を求める方法の概念設計案を示すことを試みる。

利用者記述の表現法はいろいろあるが、筆者らの調査結果[KC98]から判断して自然言語の形とする。

2　問題分析

2.1　記述の多様性の要因

利用者の記述ではどのような書き方が可能であろうか。同じ内容を表現する方法として少なくとも以下の諸点が挙げられる。

文章の形態: 平叙文か疑問文の形か、また、利用者の意図を表したものか、単に事実の説明か、等の違いで異なる表現が可能となる。例「I get a file. How to get a file? I want to get a file.」

動作記述の視点: 意味論で言われている「状況の型(situatuin type)」の違いに当たる。例えば、位相動詞としての見方、即ち、動作の開始、継続中、または、終了に注目する。具体的には、「 put a file, transfer a file, get a file. 」別の例では、動作を全体として捉えるか、部分に注目して捉えるか、という見方もある。

言い回しの違い: 同じ動作を動詞句(つまり、節)として見るか、名詞句として捉えるかで表現が異なる。例「because this event occurs, because of occurrence of this event 」また、何を主語に置くかで表現が変わる。例「This file belongs to me. I have this file.」

対象記述の視点: 対象を具体的な固有名で言うか、または、その対象の性質等を説明してその対象を指すようにするかで表現が異なる。意味論で知られた参照(reference)の度合の違いによる表現の差もある。

利用者の意図や知識に基づいて記述されるので、記述の内容が詳しくなったり、省略が行なわれたりする。

2.2　コマンドの理解とは

現実の計算機ではコマンドが分かるということは、(1)個々のコマンド名の処理内容やコマンド間の制御構造の意味、(2)コマンド間の制御構造を含めたコマンドの正確な固有表記法、の両方が分かる、ということであろう。ところが、前者に関しては、慣れていない利用者はコマンド名から受ける印象や先入観などによってコマンド名と処理との対応が正しくないことが起こる。コマンドの処理が利用者の予想した方法(How to do)と異なるやり方の処理を行なう場合は尚更分かりにくい。

コマンドの処理方法とは独立に、利用者の立場に立って、利用者が理解しているような処理方法で説明でき、これを計算機側で理解できればコマンドを理解しにくいという問題は解消するであろう。換言すると、コマンドの理解の方法が利用者が思っている処理方法とシステムの処理方法が異なっていてもよいとするものである。

3　解決手法

3.1　動作処理の概念化

ある処理を行なう方法は一通りとは限らない。ところが、コマンドはある処理について特定のやり方だけを表

*A Framework for flexible description of computer commands
Funiko Kouda and Takashi Chikayama
Graduate School of Engineering, the University of Tokyo
3-1 Hongo 7-chome, Bunkyo-ku, Tokyo 113-8656, Japan

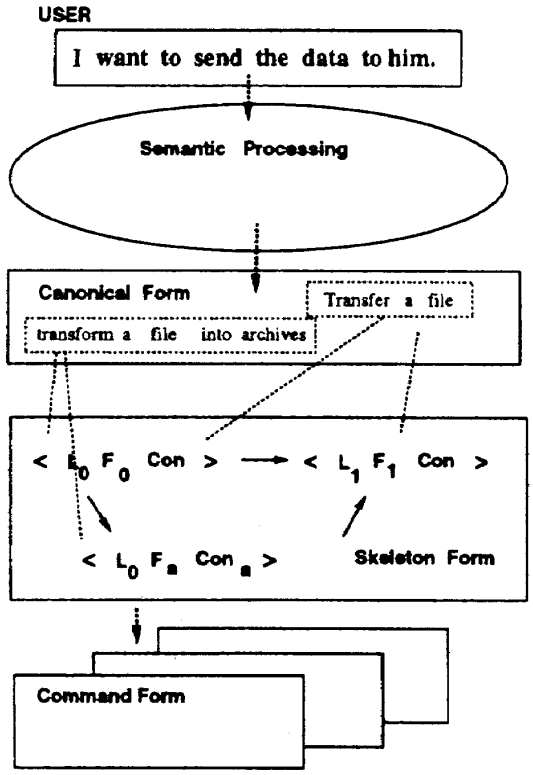

図 1: Overview of the Framework

す。計算機で行なう処理を理解するために、コマンドの処理という面を離れて、計算機で行なう処理を一般的、抽象的に捉え直す。

ある処理をするとは、その動作の前後で何かが変化することなので、(通常の状態遷移表現と同様に) 処理に関係する入力と変化する項目に注目して、これらの組を作り、変化の順序を列にした時間方向に進む非循環グラフ (direct acyclic graph) としてその処理を表す。

この記法では、処理の開始と終了を表す組を直接繋いだグラフは特定の方法に依存しない処理の記述 (what to do) となり、処理途中の変化の組を含めたグラフはその方法 (How to do) を用いた処理の記述となる。最初と最後が同じで途中が異なる列は別の方法を表す。この記法では、列同士は、同じ内容の組の所で繋ぐことができる。このことで処理の組合せが表現できる。

このようにこの記法は、処理の関係が「透けて」見えるので骨格形式 (skeleton form) と呼びたい (図 1)。

実際のコマンドは、この概念処理のグラフから、環境などを考慮した写像により導き出す。

3.2 記述から抽出するもの

ここでの利用者記述の理解は、厳密な字義の解釈ではなく、3.1節の変化するもの、即ち、何に対してどんな動作をどのように行なうかが分かればよい。

利用者の記述から抽出すべき内容は、(1) 記述から動作単位を取り出す、(2) 動作間の関係を明確にする、(3) 動作に関係するものを明確にする、(4) 対象物の記述では、物の種類や性質を見い出す、等である。

(2) は、個々の動作が、独立か、ある動作の対象が別の動作にも関係するか、動作間で処理の依存関係、前後関係、再帰関係等の関係があるか、である。

(3) では、動作の関与者、動作の場所や時間、動作の方法、理由、原因、等動作が行なわれる状況を取り出す。

(4) では、例えば、ファイル属性として、識別子、ファイルの役割が管理テーブル、プログラムソース、データリスト等である、という記述、ファイルの所有者、存在場所、アクセス資格、等の項目がある。

3.3 記述の標準形

3.2節の (1) については 2.1節の項目は、例えば、利用者の意図を除き、動作処理だけを取り出す等の変換で動作単位に分解する。

意味を変えないように動詞や関係する名詞などを入れ変え、ある処理に対する記述を、多様な形から特定の動作単位の表現に変換する。この特定の表現を標準形式 (canonical form) と呼ぶ。

4 枠組概要

利用者記述からコマンドへの変換概念図の例を図 1に示す。標準形式では動作表現が表されているが、しかし、動作自体が何であるか説明されていない。一方、骨格形式では動作そのものの記述がされている。そこで、標準形式の節で表される動作は対応するグラフの場所に写像される。

一例として、図 1では、L_i を場所、F を器、Con をその内容として、標準形式の「transfer a file」は

$< L_0, F_0, Con >$ から $< L_1, F_1, Con >$ へ直接向かうグラフであり、転送時にアーカイブを作る場合は

$< L_0, F_0, Con > \rightarrow < L_0, F_a, Con_a >$

$\rightarrow < L_1, F_1, Con >$ となる。

5 おわりに

自由で多様な利用者記述から特定のコマンドを求める方法の基本構想を示した。動作処理を概念化し、これを軸にして、多様な利用者記述の内容同定と個別のコマンドへの対応つけを行なう。この方式では、利用者の処理に対するやり方の考え方とシステムのそれが異なっても対応付けが可能となる利点がある。詳細は今後の課題である。

参考文献 [KC98] 情報処理学会 第 56 回 全国大会 6B-06