1B1-06

言語表現によるプログラミングの実験的考察

An Experimental Study on Computer Programming with Linguistic Expressions

金子 望[*1] 鬼沢 武久[*1]
Nozomu KANEKO Takehisa ONISAWA

[*1] 筑波大学 システム情報工学研究科
Systems and Information Engineering, University of Tsukuba

This paper describes the end-users programming system that generates computer programs through interaction using linguistic expressions with users who have little knowledge of computer programming. The present system employs the idea of *computer programming by paraphrasing*, which is usually used as human's everyday communication. The system also employs the case-based reasoning method for paraphrasing so that users can add the meanings of linguistic expressions that the system does not have. Two subject experiments are carried out in order to confirm the usefulness of the present system. The one task is simple text editing and the other task is more complex one. Ten subjects in all perform the experiments and the experimental results are discussed in this paper.

1. はじめに

自然言語 (NL) は, 人間のコミュニケーション手段の中で最も柔軟かつ表現力豊かなものである. そのため機械と人間のインタフェースに自然言語を用いることは人工知能研究の重要なテーマの一つとなっており, 積木の世界において人間と自然な英語で対話を行う SHRDLU システム [1] など, 人工知能研究の初期から盛んに研究されている. 近年では従来の数値に基づく情報処理に代わってあらゆる情報処理を日常言語で行う「日常言語コンピューティング」という概念が提案されるなど, 言語の持つ重要性が改めて指摘されている [2].

従来のコンピュータプログラミングはプログラミング言語を用いて行われるため, プログラミング言語の知識を持たないエンドユーザがプログラミングを行うことは困難である. このような問題を解決するために, これまでエンドユーザプログラミングに関する様々な研究が行われてきた. その一つに, 直接操作による例示によってプログラムを自動生成する Programming by Demonstration (PBD) 技術がある [3]. しかし, PBD システムは直接操作のみからユーザの意図を推論するため, 常にユーザの意図がシステムに正しく伝わるとは限らず, そのような推論エラーをユーザが訂正することは困難である. また, 言語表現を用いたエンドユーザプログラミングシステムには例えば, 英語による入力から Java 言語の構文木を生成する NaturalJava [4] や, オントロジーを用いて概念レベルで問題解決を行う CLEPE [5] などがある. しかしこれらのシステムは, 限定された世界に関する知識のみを扱っていたり, 実世界の膨大な量の知識が必要であったりと, 未解決の問題を持っている.

本研究では, プログラミングの知識を持たないエンドユーザでも自然にプログラミングを行えるように, 人間同士の日常的な対話で有効な「言い換え」をプログラミングの手段として取り入れた "言い換えによるプログラミング" [6, 7] の考え方を採用する. システムの行う言い換えには事例ベース推論を用いる. そのためシステムにとって未知の言語表現の意味を, ユーザ自身が追加することが可能である. 本論文では, テキスト編集タスクを実行する 2 種類の被験者実験を行い, システムの有効性を実験的に議論する.

連絡先: 〒 305-8573 茨城県つくば市天王台 1-1-1
E-mail: nozom@fhuman.esys.tsukuba.ac.jp

2. 言い換えによるプログラミング

言語表現を用いたコンピュータプログラミングは, 言語表現から機械言語への変換とみなせる. ここで, 変換プロセスは図 1 のように「翻訳」と「言い換え」の 2 つの部分に分けられる. 言語表現の集合 L は無数に要素が存在するのに対して, 機械言語 M はそれより要素の数がずっと少ない有限の集合である. そこで, M とほぼ 1 対 1 に対応する部分集合 L_0 を定めて, L_0 と M の間でのみ写像 t を定義する. このとき, 翻訳は特定の言語表現と機械言語の間の写像 t である. また言い換えは L 内の 2 つの言語表現の間の写像 p である. そして, 任意の言語表現に対し, 複数の言い換えを行うことにより翻訳可能な言語表現 L_0 へと写像し, 最終的に機械言語へと翻訳する. 言い換えは言語表現間の写像であるため, プログラミングの知識を持たないエンドユーザでも理解できるという利点がある.

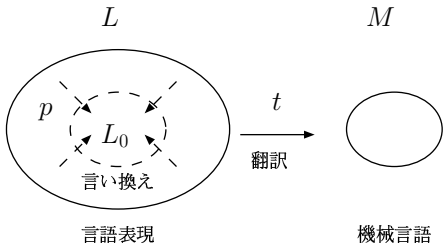

図 1: 翻訳と言い換え

3. システム構成

本システムの構成を図 2 に示す. 本システムは言語表現によってユーザと対話を行いながら, 言い換えと翻訳を通じてプログラムを生成する. 言い換えと翻訳には事例ベース推論を用いており, ユーザは言い換え事例を追加することが可能である. 翻訳事例に関しては, ユーザはその妥当性を判断できないためユーザによる追加は行わず, あらかじめ必要な事例を用意しておくものとする. システムは入力された言語表現と類似した事例を検索し, 選択された事例を基に言い換えまたは翻訳を出力する. なお本システムは, テキストエディタ GNU Emacs 上に構築されており, Emacs Lisp を用いて実装している.

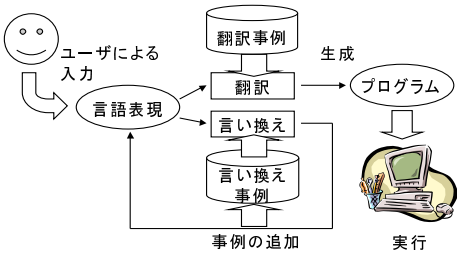

図 2: システム構成

3.1 翻訳実行部

翻訳実行部は, 事例ベース推論を用いて入力された言語表現からコンピュータプログラムを生成する. 翻訳実行部は言語表現とプログラムの対応関係を記述した翻訳事例のデータベースを持っており, 句構造に基づいて入力された言語表現を事例ベースに含まれる言語表現と比較する. 言語表現の句構造は, 形態素解析と係り受け解析を行った結果の構文木として得られる. 形態素解析には茶筌 [8] を用いている.

3.2 言い換え実行部

言い換えも翻訳と同様に事例ベース推論によって実行されるが, 出力がプログラムではなく言語表現であるという点が異なる. 言い換えは以下の手順で行われる. まず, 入力された言語表現に部分的に一致した事例が候補として選択される. 構文木同士の類似度を, Convolution Kernel [9, 10] に基づいて (1) 式により定義する.

$$K(T_1, T_2) = \max_{n_1 \in N_1} \max_{n_2 \in N_2} C(n_1, n_2) \qquad (1)$$

ここで T_1,T_2 は構文木であり, N_1,N_2 はそれぞれ T_1,T_2 に含まれる全てのノードの集合である. また, ノード間の類似度 $C(n_1, n_2)$ は次のように求められる.

Case 1. ノード n_1 と n_2 の品詞が異なるとき, $C(n_1, n_2) = 0$

Case 2. ノード n_1 と n_2 が同一の単語のとき, $C(n_1, n_2) = 1$

Case 3. ノード n_1 と n_2 が同一の単語でなく, かつ n_1 と n_2 の品詞が同じとき,

$$C(n_1, n_2) = \prod_{k \in ch(n_1)} \prod_{l \in ch(n_2)} (1 + C(k, l)) \qquad (2)$$

ただし (2) 式において $ch(n_1)$, $ch(n_2)$ はそれぞれ n_1, n_2 の子ノードの集合である. 次に, システムは入力された言語表現との類似度が高い候補から順に, 図 3 の手順で新しい言い換えを生成する. (1) 言い換え事例の言い換え前の言語表現と入力された言語表現を比較し, 句構造同士の差分を検出する. (2) 言い換え事例の言い換え後の言語表現に (1) で得られた差分を適用する. その結果, (3) 新たな言い換えを生成する. 最後に, 生成された言い換えが入力された言語表現に対して適切かどうかをユーザに尋ね, 適切と判断された場合は新たな言い換えをデータベースに登録する.

3.3 言い換え事例の追加

類似事例が見付からなかった場合, ユーザ自身に言い換えを求め, 言い換え事例を追加する. このようにして追加された言語表現はその後の対話で使用することができる.

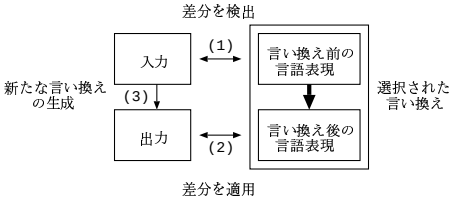

図 3: 言い換え生成の流れ図

4. 実験

本システムの有効性を検証するために実験を行う. 実験は GNU Emacs 21.3 を使用して行われ, 実験環境の OS は Linux である.

4.1 実験 1

実験 1 の被験者は 5 名で, 図 4 に示す編集前の文章を, 図 5 に示す編集後の文章に変換するための言語表現を入力することで行われる.

```
タイトル：言語理解の構造
著者：テリー・ウィノグラード
出版社：産業図書
出版年：1976
ISBN：4-7828-5236-3
```

図 4: 編集前のテキスト (実験 1)

```
テリー・ウィノグラード：『言語理解の構造』, 産業図書 (1976).
```

図 5: 編集後のテキスト (実験 1)

実験の結果, 5 人の被験者全員が目標とするプログラムを作成している. 表 1 は得られた言い換え事例の内訳を示したものである. 「具体化」はある言語表現の意味を具体的に言い直した場合である. ユーザの意図とシステムが実行可能な操作の間に概念的なギャップが存在する場合, 「具体化」の言い換えによってそのギャップが埋められる. 例えば, 「文献データのフォーマットを変更する」を具体的に言い換えると、「タイトルを編集して、著者と出版社を移動して、出版年を書く」のようになる. 「言い回しの変更」はユーザ自身の言葉をシステムの語彙に追加する言い換えの一種で, 例えば「選択する」と「選択を開始する」の間の言い換えである. それ以外に, ユーザの入力ミスやシステムのバグによって誤って追加された事例がいくつか存在する.

表 1: 言い換え事例の内訳 (実験 1)

被験者	A	B	C	D	E	全体
具体化	10	6	16	18	16	66
言い回し	1	5	10	1	2	19
ミス	1	0	1	2	5	9
バグ	0	0	1	2	0	3
総数	12	11	28	23	23	97

実験終了後に「システムの使いやすさ」「得られた言語表現の主観的な分かりやすさ」「言い換えによるプログラミングは直観的かどうか」について 5 段階評価のアンケートを行っている. その結果は表 2 のようになり, 概ね言語表現を用いたプログラミングは理解しやすいという評価が得られている.

表 2: アンケート結果 (実験 1)

被験者	A	B	C	D	E
システムの使いやすさ	3	4	4	2	4
言語表現の分かりやすさ	4	4	4	3	5
直観的かどうか	4	5	5	4	4

表 3 は被験者毎の言い換えの傾向を示したものである. この表から, 被験者 B, C は言い換え後の言語表現の約 30%が再び他の言語表現に言い換えられていることがわかる. 一方, 被験者 D はそのような複数回の言い換えをほとんど行っておらず, 入力した言い換えが再利用されることが少ないため, 言い換え後の言語表現が非常に多い. このように, 言い換えの傾向は被験者によって大きく異なっている.

表 3: 言い換えの傾向 (実験 1)

被験者	A	B	C	D	E	全体
言い換え後の言語表現の総数 ($=X$) (平均)	52 (4.33)	49 (4.45)	88 (3.14)	113 (4.91)	96 (4.17)	398 (4.10)
複数回の言い換えが行われた言語表現の数 ($=Y$)	10	15	34	8	11	78
Y/X	19%	31%	39%	7%	12%	20%

この実験で得られた言い換え事例の一部を表 4 および表 5 に示す. この結果から, 被験者毎に異なった言い換え事例が獲得されていることがわかる. それぞれの言い換えは単なる表記の違いだけでなく, その構造についても被験者毎に異なっており, これは各被験者の知識構造を反映したものと考えられる.

表 4: 実験 1 で得られた言い換え事例 (被験者 A)

言い換え前	言い換え後
著者を行頭に移動。	文頭に移動。「著者」を検索。行頭に移動。選択を開始する。行末に移動。切り取る。1 文字削除。文頭に移動。改行。文頭に移動。貼り付ける。
著者を文頭に移動する。	著者を行頭に移動。
著者にコロンをつける。	文頭に移動。「著者」を検索。行末に移動。「：」を挿入。
タイトルに括弧とコンマをつける。	文頭に移動。「タイトル」を検索。2 文字右に移動。「『」を挿入。行末に移動。「』,」を挿入。
出版年にカッコとピリオドをつける。	文頭に移動。「出版年」を検索。2 文字右に移動。「(」を挿入。行末に移動。「) .」を挿入。
改行を消す。	行末に移動。1 文字削除する。
「著者：」を削除。	「著者：」を「」に置換。
タグ消しその 1。	文頭に移動。「著者: 」を「」に置換。行末に移動。改行を消す。「タイトル: 」を「」に置換。行末に移動。改行を消す。
タグ消しその 2。	「出版社: 」を「」に置換。改行を消す。「出版年: 」を「」に置換。
タグ消しその 3。	1 行下に移動。行頭に移動。選択を開始する。行末に移動。選択範囲を削除する。
文献データの書式を変更する。	著者を文頭に移動。著者にコロンをつける。タイトルに括弧とコンマをつける。出版年にカッコとピリオドをつける。タグ消しその 1。タグ消しその 2。タグ消しその 3。

4.2 実験 2

実験 2 では, 被験者に以下の 4 つの課題を与え, 図 6 に示すテキストを編集して, 図 7 に示すような HTML 形式とするための言語表現を入力する実験を行う. 本実験の被験者は実験 1 とは別の 5 名である.

表 5: 実験 1 で得られた言い換え事例 (被験者 B)

言い換え前	言い換え後
選択する。	選択を開始する。
削除する。	選択範囲を削除する。
右へ移動する。	右に移動する。
行末へ移動する。	行末に移動する。
文頭へ移動する。	文頭に移動する。
タイトルを編集する。	選択する。「：」を検索する。左に移動する。削除する。右に移動する。『』を挿入する。行末に移動する。「』」を挿入する。
著者を移動する。	「著者」を検索する。右に移動する。選択する。行末に移動する。コピーする。文頭に移動する。貼り付ける。
出版社を移動する。	「出版社」を検索する。右へ移動する。選択する。行末へ移動する。コピーする。文頭へ移動する。行末へ移動する。「,」を挿入する。貼り付ける。
出版年を書く。	「出版年」を検索する。右に移動する。選択する。行末に移動する。コピーする。文頭に移動する。行末に移動する。「(」を挿入する。貼り付ける。行末に移動する。「) .」を挿入する。
しあげする。	行末に移動する。選択する。文末に移動する。削除する。
文献データの書式を変更する。	タイトルを編集する。著者を移動する。出版社を移動する。出版年を書く。しあげする。

課題 (1) 1 行目をタイトルにする

課題 (2) “*” で始まる 3 行目を大見出しにする

課題 (3) 行頭が “-” で始まる部分を箇条書きにする

課題 (4) 図 6 のテキスト全体を編集し, Web ブラウザで図 7 のように表示されるようにする.

この課題は, 課題 (1)〜(3) のすべての内容に加えて, HTML 形式にするための追加の課題を含んでいる.

```
Wiki とは

*概要

ウェブブラウザから発行・編集可能なツール、またはそのツールによって作られたウェブサイトのこと。誰でも簡単に記事を発行・編集できる点が Wiki の特徴。

*整形ルール

**段落、見出し、テキストの整形

通常は入力した文字がそのまま出力されますが、以下のルールに従ってテキスト整形を行うことができます。

HTML のタグは書けません。

-1 行目はタイトルになります。
-空行は段落の区切りとなります。
-アスタリスク (*) を行頭に書くと、大見出しになります。
-アスタリスク 2 個 (**) を行頭に書くと、小見出しになります。
-マイナス (-) を行頭に書くと、箇条書きになります。
```

図 6: 編集前のテキスト (実験 2)

実験の結果, 被験者 5 名中 3 名が 4 つの課題を全て達成できているが, 2 名の被験者 G,I は課題 (4) が達成できていない. これらの被験者はシステムとの対話においてしばしば言語表現の意味を正確に理解せず, そのため課題を達成できなかったものと考えられる.

実験後に行ったアンケートの結果は表 6 の通りである. 実験 2 は実験 1 と比べて難しい課題であるにも関わらず, 言語表現の分かりやすさ, および直観的かどうかの項目について, 分かりやすい, あるいは直観的と答えた被験者が多い. しかしシステムの使いやすさに関しては実験 1 より実験 2 で有意に低下が見られ, 課題の複雑さに対して不十分であるということがわかる. 一方, 表 7 は, 被験者がそれぞれの課題に対して感じた難易度を「とても易しい」から「とても難しい」までの 5 段階で回答した結果である. どの被験者も課題 (3) と課題 (4) で難

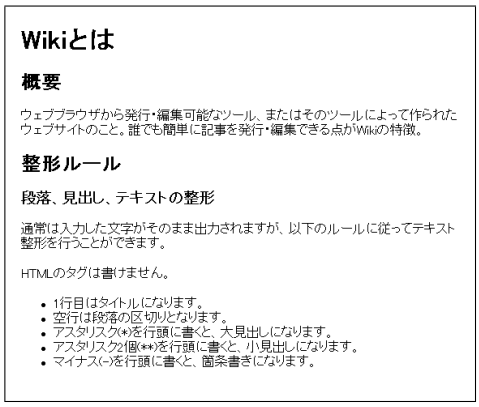

図 7: 編集後の状態 (実験 2)

易度が高くなったと感じており, 特に最後まで課題を達成できなかった被験者 G と I については課題 (3) で既に「とても難しい」と感じている.

表 6: アンケート結果 (実験 2)

被験者	F	G	H	I	J
システムの使いやすさ	2	2	2	2	2
言語表現の分かりやすさ	4	2	5	4	4
直観的かどうか	4	4	2	5	4

表 7: 実験 2 の各課題の難易度

被験者	F	G	H	I	J
課題 (1)	1	3	1	3	1
課題 (2)	1	3	2	3	1
課題 (3)	3	5	4	5	3
課題 (4)	4	-	4	-	5

図 8 は課題 (3) で生成されたプログラムの例である. 図の上部に示す言語表現に対応して, 下部のプログラムが生成される.

```
行頭が"-"とマッチしたら、箇条書きのタグを挿入する
⇓
(if (save-excursion (beginning-of-line) (looking-at "-"))
    (progn
      (beginning-of-line)
      (delete-char 1)
      (insert "<li>")))
```

図 8: 生成されたコンピュータプログラムの例

実験後に多くの被験者から指摘されたことは, 追加した事例の修正を行いたいというものである. 現在のシステムでは被験者の試行錯誤が考慮されていないが, 被験者は課題を達成するための作業手順を明確に説明できない場合には試行錯誤で課題を行うことが多い. よって, 本実験の課題 (4) のような複雑な問題に対しては何らかの支援機能が必要であると考えられる.

5. おわりに

本稿では, 言語表現を用いたエンドユーザプログラミングシステムについて記述し, 被験者実験によってその有効性を確認した. 実験の結果, 本研究で提案する言い換えを用いたプログラミングは, ユーザにとって直観的な方法であり, 得られる言語表現も分かりやすいものであるという結果が得られた. しかし複雑な問題に対しては現在のシステムのインタフェースでは不十分であり, 何らかの支援機能が必要であることがわかった. また, 現在のシステムは文法的でない入力を扱うことができないため, ユーザに入力形式をある程度制御してもらっているが, 初心者ユーザを対象にする場合これは問題となる. そのため, 入力が非文法的であっても何らかの知識に基づいた言い換えを行って文法的な誤りを訂正するといった処理が必要になると考えられる.

参考文献

[1] テリー・ウィノグラード: 『言語理解の構造』, 産業図書 (1976).

[2] 伊藤紀子, 杉本徹, 岩下志乃, 岩爪道昭, 高橋祐介, 小林一郎, 菅野道夫: “セミオティックベースを使った日常言語アプリケーションシステム (第 2 報)”, 第 18 回人工知能学会全国大会, 石川, 2C1-05, June 2004.

[3] Henry Lieberman: “Your Wish is My Command: Programming by Example”, Morgan Kaufmann Publishers (2001).

[4] David Price et al.: “NaturalJava: A Natural Language Interface for Programming in Java”, Proc. of 5th International Conference on Intelligent User Interfaces, New Orleans, Louisiana, pp. 207-211 (2000).

[5] Kazuhisa Seta, Mitsuru Ikeda, Osamu Kakusho, and Riichiro Mizoguchi: Capturing a Conceptual Model for End-User Programming: Task Ontology As a Static User Model, Proc. of 6th International Conference on User Modeling, Chia Laguna, Sardinia, pp. 203-214 (1997).

[6] 金子望, 鬼沢武久: “対話と言い換えを用いた言語表現によるプログラミングシステム”, 第 18 回人工知能学会全国大会, Vol.18, pp.3E2-05 (2004).

[7] Nozomu Kaneko, Takehisa Onisawa: “End-User Programming by Linguistic Expression employing Interaction and Paraphrasing”, SCIS & ISIS 2004, Keio Univ, Japan (2004).

[8] 松本裕治 他: 形態素解析システム『茶筌』version 2.3.0 使用説明書, 奈良先端科学技術大学院大学 (2003).

[9] M. Collins, N. Duffy: “Convolution Kernels for Natural Language”, Neural Information Processing Systems, Vol. 14, pp. 625-632 (2002).

[10] Tetsuro Takahashi, Kozo Nawata, Kentaro Inui, Yuji Matsumoto: “Effects of Structural Matching and Paraphrasing in Question Answering”, IEICE Transactions on Information and Systems (2003).