1F4-1

日常言語による状況に依存したプログラミング

Context-Sensitive Programming in Everyday Language

杉本 徹[*1] 伊藤 紀子[*2] 岩下 志乃[*3]
Toru Sugimoto Noriko Ito Shino Iwashita

[*1]理化学研究所 脳科学総合研究センター
Brain Science Institute, RIKEN

[*2]同志社大学 文化情報学部
Faculty of Culture and Information Science, Doshisha University

[*3]九州大学大学院 芸術工学研究院
Faculty of Design, Kyushu University

This paper presents a processing model of everyday language programs with which non-programmars can specify their complex requests to manipulate application software. In particular, we discuss methods for implementing flexible operations according to the user's and the system's situations of both program construction and execution, that reflect context-dependency of everyday language.

1. はじめに

近年の情報技術の発展に伴い多くの人が日常生活の中でパーソナルコンピュータやデジタル化された機器に接するようになったが，それらが持つ豊富で複雑な機能を必ずしも使いこなせていないのが現状である．ユーザが希望する操作を簡単に指定できるように，応用ソフトウェアにおけるテンプレートやウィザード，デジタル機器におけるプリセットメニューなどが用意される場合もあるが，多様なユーザの多様なニーズに十分に応えることは難しい．一般に，情報システムの操作を自分が望む手順，条件に従って実行するためには，何らかのプログラミングが必要となり，プログラミング言語などの専門知識が必要となる．また，大抵のシステムではそのシステム独自の機能をサポートする専用のライブラリ（関数，クラス，メソッドなど）を利用することが不可欠であり，たとえ熟練したプログラマであってもその仕様を時間をかけて学ぶ必要がある．

このような問題に対処するため，我々は**日常言語**によるプログラミング方式を提案している [Sugimoto 04a, 杉本 04b]．この枠組で，ユーザは行いたい処理の手順や条件を日常言語テキストとして表現し，システムに入力する（入力テキストの例を図 1 に示す）．システムは，入力テキストの意味内容を理解し，実行可能なプログラムに変換する．文献 [Sugimoto 04a, 杉本 04b] では，電子メール管理タスクを例として処理手法を与えることにより，言葉を使って容易にプログラムを作成し複雑な操作を実行できることを示した．

一方，日常言語には**状況依存性**，すなわちそれが使用される状況に応じて異なる形式や意味を持つという性質がある．本稿では，日常言語を使ったプログラミングによって状況に応じた柔軟なプログラムの作成と実行を行うことができることを示し，その可能性について議論する．

(1) 理研の人から来たメールは「理研」フォルダに入れる．
(2) 理研の人のメールアドレスは，末尾に「.riken.jp」が付く．
(3) 研究室の人からのメールは,「研究室」フォルダに入れて自宅に転送する．
(4) ただし，伊藤さん，高橋さん，小林さんからのメールでセミオティックベースに関するものは,「ＳＢ」という名前のフォルダに入れる．
(5) この時，ファイルがあったらマイドキュメントの「ＳＢ」という名前のフォルダに保存する．
(6) また，このファイルが文書ファイルの場合は，それを出す．

図 1: 日常言語プログラムの例

連絡先: 杉本徹，2006 年 4 月より芝浦工業大学工学部情報工学科，〒 135-8548 東京都江東区豊洲 3-7-5, sugimoto@shibaura-it.ac.jp．岩下志乃は，2006 年 4 月より東京工科大学コンピュータサイエンス学部．

2. プログラミングにおける状況依存性

プログラミングの過程では，プログラムを作成する状況とプログラムを実行する状況の違いを考慮する必要がある（図 2）．作成後直ちに一度だけ実行されるようなプログラムの場合は，作成の状況と実行の状況が一致するか，前者において後者が予見できることがほとんどであるが，多くのプログラムは記憶装置に保存され，後から何度でも呼び出して利用することができる．このような場合，プログラムがどのような状況で実行されるか作成時には分からないこともありうる．

ここで状況の構成要素としては，プログラムが動作するシステムの環境や状態，およびプログラム作成者のプロファイル，すなわち知識や好み，くせなどが考えられる．また，プログラムが実行時にユーザと対話的なやり取りを行う場合は，さらにユーザのプロファイルもプログラム実行の状況の一部となる．

通常のプログラミング言語を用いたプログラミングでは，プログラマはプログラム作成の状況に依存しないような形で仕様を記述する必要がある．さらに，プログラム実行の状況が不

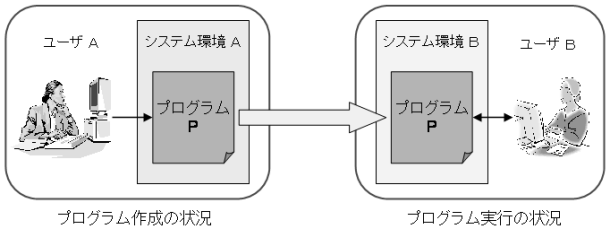

図 2: プログラム作成と実行の状況

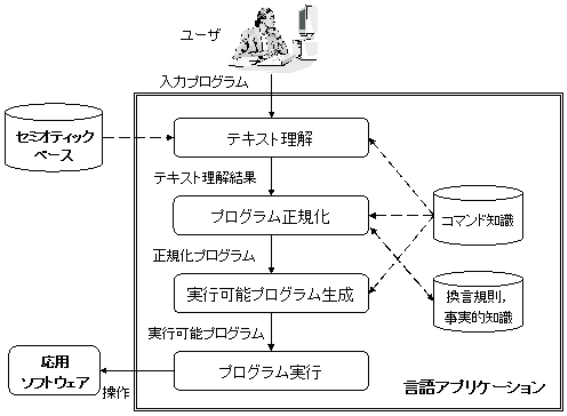

図 3: 日常言語プログラムの処理モデル

確定な場合は，想定されるさまざまな状況に対応できるように場合分けの形式でプログラムを作成する必要がある．次節以降では，状況依存性を備えた日常言語を用いるプログラミングでは，そのような手間がいずれも不要となることを示す．

3. 処理モデル

操作の手順や実行条件を表現した複数の文からなる一貫したテキストを**日常言語プログラム**と呼ぶ．日常言語プログラムの処理モデルの概要を図 3 に示す．この処理モデルは，日常言語を使って情報処理を行う**日常言語コンピューティング** [岩爪 03] の枠組を利用しており，特に，言葉を使って応用ソフトウェアを操作する言語アプリケーションの一機能と位置付けられる．

ユーザが入力した日常言語プログラムに対して，まずセミオティックベースと呼ばれる言語資源を使ってテキスト理解処理を行う [伊藤 04]．その結果，各文の語彙文法・意味・概念構造およびテキスト全体の修辞構造 [Mann 88] が同定される．Pane らの実験結果 [Pane 01] からも分かるように，人は複雑な論理的条件や場合分けの組み合わせ，繰り返しの構造を明示的に表現しない傾向がある．テキストの修辞構造は，操作・条件間の暗黙に表された関係を抽出しプログラムの制御構造を作り出すための大きな手がかりとなる．

入力されたプログラムは一般に，実行可能な形，すなわち言語アプリケーションのコマンドと明示的な制御構造の組み合わせの形にはなっていない．そこでコマンドに関する知識を参照して各文の実行可能性を検査し，必要に応じて換言規則を用いた換言の処理を行う．このプログラム正規化の過程で，文中の暗黙的なループ構造の検出も行われる．

その後，入力プログラムの修辞構造を考慮しながら各文の意味内容を統合し，実行可能な形をしたプログラムテキスト（**実行可能プログラム**）を生成する．図 1 のプログラムの処理結果として生成される実行可能プログラムを図 4 に示す．

最後に，言語アプリケーションのコマンド実行機能を使って，応用ソフトウェアの操作を実行する．生成された実行可能プログラムは，逐次翻訳（インタープリタ）方式または一括変換（コンパイラ）方式により実行環境におけるソフトウェア実行機能と動的に結び付けられ，実行される．

4. 議論

4.1 プログラム作成の状況に依存した処理

プログラム作成の状況に依存した処理は，主にテキスト理解の過程で実現される．例えば，図 1 のプログラムに含まれる「ファイルがあったら」,「それを出す」という表現は，このプログラムを入力したユーザがそれぞれ「添付ファイル」,「印刷する」の意味で使ったとすれば，プログラム生成の過程でそのような一般的な語句に置き換える必要がある．また例えば「伊藤さん」という表現は,「伊藤」という名前を持つ人一般を指すのではなく，プログラム作成者が「伊藤さん」と呼んでいる特別の対象を指している．このような表現の処理は，前節で述べた処理モデルにおけるテキスト理解の機能によって実現される．テキスト理解では，表層の言語表現に依存しない概念的意味の同定が行われる．特に，セミオティックベースはユーザプロファイル情報の一種として発話者の知識レベルや言語使用の傾向に関する情報を参照することができ，それによって個別のユーザに特化したテキスト理解を行うことができる．

4.2 プログラム実行の状況に依存した操作実行

生成された実行可能プログラムに含まれる操作や対象は，そのプログラムが実行される環境（言語アプリケーション）のコマンド実行機能により，対象ソフトウェアの適切なメソッドやオブジェクトと動的に結び付けられて実行される．例えば，図 1 のプログラム中の「自宅」という語は，このプログラムを入力したユーザの自宅のメールアドレスを表す具体的な文字列に置き換えられることなく図 4 の実行可能プログラム中でもやはり「自宅」という語で表現され，最終的にプログラム実行の過程でそのプログラムを呼び出したユーザの自宅のメールアドレスに結び付けられる．これは「出す（印刷する）」に関しても同様であり，実行時の環境やファイルの種類に応じて適切な手続きが呼び出されることになる．さらに，このプログラムの実行時にユーザとのやり取り（例えば，操作の確認）が必要である場合は，ユーザのプロファイルに応じて例えば「添付ファイル」という語をより分かりやすい「メールに付いているファイル」に置き換えて出力するなどの処理が必要となる可能性もある．これらの処理はすべて，我々の枠組において実行可能プログラムが，状況に応じてさまざまな意味を持ちうる日常言語を使って表現されていることによって可能となったと言える．

5. まとめ

本論文では，日常言語で書かれた操作の仕様を理解して，応用ソフトウェアの操作を実行する手法を提案した．専門知識が無いと理解できないプログラミング言語と異なり，日常言語には誰でも簡単に読み書きできるという利点がある．我々の枠組は，システムへの入力が日常言語であるため誰でも気軽に使うことができ，また実行直前のプログラムも日常言語で表現されるためシステムが行う処理の途中経過を容易に確認することができる．この二つの利点に加え，今回議論したように言語の状況依存性によって，プログラム作成者の置かれた状況となるべく独立に，かつ実行時の環境に合わせて動作する再利用性の高いプログラムを比較的容易に作成できると期待される．この点で我々の枠組は，自然言語テキストをさまざまな通信の媒体とすることで送信者のメッセージを受信者が自分の都合に合わせて解釈する「解釈を相手に委ねる」形の通信方式を目指す言語プロトコル [小林 02, 小林 05] の考え方とと共通点を持つと言うことができる．

```
受信メールボックスの中のすべてのメールに対して，次の処理を繰り返す．
  もし，そのメールの差出人が伊藤さんと等しい，または…，かつそのメールがセミオティックベースに関するならば，次の処理を行う．
    そのメールを「ＳＢ」フォルダに入れる．
    もし，添付ファイルがあるならば，その添付ファイルをマイドキュメントの「ＳＢ」フォルダに保存する．
    もし，添付ファイルが文書ファイルであるならば，その添付ファイルを印刷する．
  以上．
  そうでなければ，もし，そのメールの差出人が菅野先生と等しい，または…ならば，次の処理を行う．
    そのメールを「研究室」フォルダに入れる．
    そのメールを自宅に転送する．
  以上．
  そうでなければ，もし，そのメールの差出人アドレスの末尾に「.riken.jp」が付くならば，そのメールを「理研」フォルダに入れる．
以上．
```

図 4: 生成された実行可能プログラム

今後の課題としては，プログラミングの過程に影響を与える状況の構成要素を整理し，状況に関する情報を抽出して状況に依存した言語表現を理解・生成する手法を開発することが挙げられる．

参考文献

[伊藤 04] 伊藤 紀子, 杉本 徹, 菅野 道夫：選択体系機能言語学に基づく日本語テクスト理解システムの実装, *JASFL Occational Papers*, Vol. 3, No. 1, pp. 189–206 (2004)

[岩爪 03] 岩爪 道昭, 小林 一郎, 杉本 徹, 岩下 志乃, 高橋 祐介, 伊藤 紀子, 菅野 道夫：日常言語コンピューティング（第２報）—日常言語に基づく計算機資源の管理・実行環境を目指して—, 人工知能学会論文誌, Vol. 18, No. 1, pp. 45–56 (2003)

[小林 02] 小林 一郎, 岩爪 道昭, 杉本 徹, 岩下 志乃, 小澤 順, 菅野 道夫：自然言語をコンピュータの通信プロトコルにする研究, 日本ファジィ学会誌, Vol. 14, No. 5, pp. 491–502 (2002)

[小林 05] 小林 一郎, 菅野 道夫：言語プロトコル通信における基本的枠組みの再検討, 第 19 回人工知能学会全国大会, 北九州 (2005), 1B1-02

[Mann 88] Mann, W. C. and Thompson, S. A.: Rhetorical Structure Theory: Toward a Functional Theory of Text Organization, *Text*, Vol. 8, No. 3, pp. 243–281 (1988)

[Pane 01] Pane, J. F., Ratanamahatana, C. A., and Myers, B. A.: Studying the Language and Structure in Non-Programmers' Solutions to Programming Problems, *International Journal of Human-Computer Studies*, Vol. 54, No. 2, pp. 237–264 (2001)

[Sugimoto 04a] Sugimoto, T., Ito, N., Iwashita, S., and Sugeno, M.: Script Generation using Rhetorical Information in a Task Specification Text, in Thissen, W., Wieringa, P., Pantic, M., and Ludema, M. eds., *Proceedings of the 2004 IEEE International Conference on Systems, Man and Cybernetics*, pp. 2687–2692 (2004)

[杉本 04b] 杉本 徹, 伊藤 紀子, 岩下 志乃, 菅野 道夫：日常言語によるプログラミング, 第 18 回人工知能学会全国大会, 金沢 (2004), 3E2-03