JavaApplet による ADF 検証システムの実現*

2K－3

原田 雅之, 唐澤 博†
山梨大学 工学部 電子情報工学科‡

E-mail: {harada, karasawa}@opal.esi.yamanashi.ac.jp

1 はじめに

情報基礎の将来は, プログラミング中心から問題分析・アルゴリズム中心へ移行すべきであり, プログラミング言語は将来的には現在のアセンブラ教育のような位置になることが予測される. 職業系以外の高校は特にその指向が重要であり, 職業系高校は実学指向からいけば今後もプログラミング能力が必要とされるだろう. しかしこの場合でも, アルゴリズム記述を主目標とし, プログラミングはアルゴリズム記述を計算機に移し変える過程のもの程度の認識に移行させる. すなわち, アルゴリズムを記述し, ターゲット言語プログラムに変換して, そのあと適切な微調整を実施して最終プログラムを得るというものである. その微調整作業のためにターゲット言語教育は必要になるが, 現在のようにそれをカリキュラムの中心にすえることがなくなるであろう.

そこで, プログラム言語による記述の詳細を捨ててアルゴリズムのみの記述に専念できるような, 流れ図のレベルの記述形式 ADF (Algorithm Description Format) が提案された [1]. プログラム言語による記述は, すでにマシンの事情を人間に考慮させているレベルとなっている. 一方, 人間が真にエネルギーをそそぐべきレベルはアルゴリズム記述のレベルである.

種々の言語は, それぞれの問題領域と結び付いている. この観点に立てば, その主張は再吟味される必要がある. 結局, さまざまな問題領域に依存しない真に処理手順を示す部分をアルゴリズム記述の範囲と考えるべきであろう.

2 ADF 検証システムの目的

以上のことから, ADF の記述は

(1) 計算機を動かす枝葉的な要請から独立していること
(2) さまざまな問題領域に依存しない真の処理手順

と整理される. このことから, ADF の実行システムは, 「指定した操作ステップにおける, 指定した変数の内容のトレース」機能を持っているべきである. これをやりやすくするために, ADF のインタープリタを作成し, コマンド群も揃える必要がある.

また, 学習場面で学生が ADF を用いて記述したロジックが, 期待する結果を生じるか検証可能な手段が提供されるべきである. ADF を説明の道具として使うだけでなく教具として役立たせるには, 検証システムを伴う必要がある. 検証システムとは, ADF を解釈実行するシステムのことである.

3 機能概要

設計や説明において, ADF は段階的詳細化に主として応用されるが, この場合はスタブを必ずしも必要としない. しかし, 本システムが検証する対象は, 完全な記述のみとする. 完全な記述とは, アルゴリズムの詳細化が, ADF モジュールライブラリとして用意している関数群にまでリダクションされていることを意味する.

検証システムは, 以下の要素から構成される.

- ADF インタプリタ
- ADF 基本モジュールライブラリ
- ADF 拡張モジュールライブラリ
- ADF エディタ
- ADF 実行窓

ADF インタプリタは, ADF 記述を中間コードにコンパイルしてから中間コードを逐次実行する. ユーザからの要求にしたがって様々な動作応答を出力する.

ADF 基本モジュールライブラリ (BML) は, ADF のモジュール名の規則にしたがって記述された組み込みモジュール名の一覧表と, そのモジュール名に対応する中間コードデータベースから構成される.

ADF 拡張モジュールライブラリ (EML) は, プログラマが登録可能なライブラリであり, 登録以後は, ADF 基本モジュールライブラリの内容とまったく同様に利用できる. BML との違いは, モジュール定義記述も保存することである. この記述は要求によりいつでも参照可能としておくことで, ライブラリ化したモジュール内容のドキュメントの役割を持たせる.

ADF エディタは, ADF の文法を理解して段下げや動的エラーチェックを行なう.

---

*ADF inspection system for Java Applet

†Masayuki Harada, Hiroshi Karasawa

‡Yamanashi University, 4-3-11 Takeda, Kofu, Yamanashi 400, Japan

4 UNIX 上でのプロトタイプの作成

まず, ADF 検証システムの基本機能確認のため, UNIX 上にて字句解析系生成システム lex および構文解析系生成システム yacc を用いてプロトタイプの作成を行なった. この動作例を図 1 に示す.

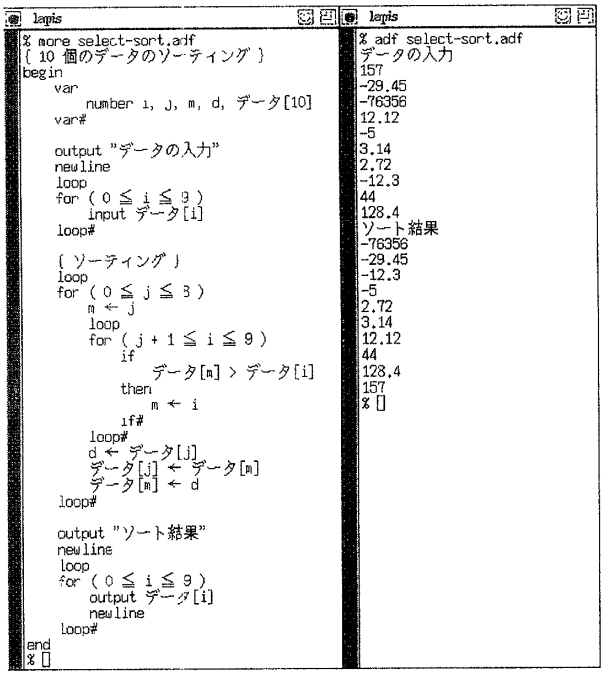

図 1: プロトタイプの動作例
(左: ADF, 右: 検証結果)

動作例はノーマルモードの場合であり, 与えられた ADF を忠実に解釈・実行している. また, トレースモードも存在し, 現在の行番号や呼び出された実行文, 変数の変動状況なども同時に出力し, アルゴリズムの動作状況の把握を容易としている.

5 Java Applet による実装

次に, Java Applet による ADF の編集, 実行, トレース処理機能を実装した検証システムを構築した. この動作例を図 2 に示す.

この Applet では, 大きく編集域と実行域に分かれており, それぞれの位置の変更やその領域のみにできる. また, それぞれの領域ごとに必要なダイアログやボタンが附属している. この動作例は, プロトタイプと同等の ADF を用いて解釈・実行した結果である. アルゴリズムは選択ソーティングであるが, 正確な動作をしているといえる.

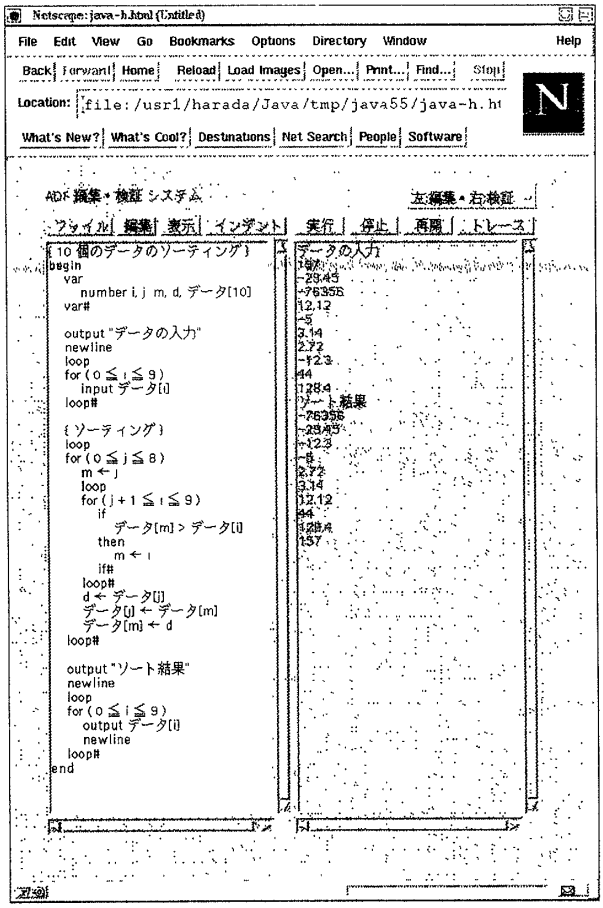

図 2: Applet による動作例

6 おわりに

ここでは, アルゴリズム記述形式 ADF のための検証システムを Java Applet として実現した. マシンや OS に依存しない技術として注目を集めている Java 言語の Applet として開発したことが, ADF にとって特定言語インディペンデント & ドキュメントベースな記述系として評価されると好ましい. なお, 本稿で例示したアルゴリズムは, 文献 [2] を参考にした.

参考文献

[1] 唐澤 博: 情報基礎教育のためのアルゴリズム記述形式の提案, 情報処理学会「コンピュータと教育」研究会資料, 95-CE-37-2, pp.9-16 (1995.7).

[2] 平田 富夫: アルゴリズムとデータ構造, 森北出版 (1990.1).