プログラム理解システム汎用化のためのプログラム表現の抽象化ついて

3K－3

荻原哲也　上野晴樹
東京電機大学

1．はじめに

我々は，約１０年をかけてプログラミング教育向きの統合型知的プログラミング環境INTELLITUTORの開発を行ってきた[1]．理系大学におけるアルゴリズム中心型のプログラミング教育の中で利用されることを意図している．主として，Pascalによるクイックソートプログラムを例題として，システム開発や評価を行ってきた．INTELLITUTORシステムは，プログラム入力支援システムGUIDE，論理ミスの検出，意図の推定，訂正の助言を行うプログラム理解システムALPUS[2]，および，プログラミング知識の教授を行う知的CAIシステムTUTORから構成される．

一方，近年，Ｃ言語がパーソナルコンピュータ，ワークステーションなどの小型コンピュータの世界で広く普及しつつあることを背景として，Pascalに代わってCの教育が行われるようになってきた．こうした背景に対応するために，INTELLITUTORを両言語対応に拡張しつつある．これにともなって，サブシステムであるALPUSも拡張を行っているが，その方法として，両言語で書かれたプログラムを一度言語独立な抽象表現に書き換えてから理解処理を行うやり方を採用した．本稿では，Ｃ言語プログラムを抽象化する方法とその能力の汎用化について報告する．なお，Ｃ言語はANSI Cとする．

2．プログラム理解システムALPUS

ALPUSは，アルゴリズム知識，プログラミング技法知識，プログラミング言語知識，バグ知識の４つを使い，プログラムに含まれる複数の論理ミスを検出し，学生の意図を推論し，訂正の助言を行う．

ALPUSの理解は，次の４ステップにより達成する[図１]．第１は，学生の作成したプログラムの言語独立な表現への変換である．これを抽象化と呼び，複数の言語への対応を可能にする．本稿で報告するところである．第２は正規化と呼び，学生の作成したプログラムの意味を壊すことなく，ALPUSの持っている知識とマッチングの取りやすい形への変換を行う．第３は，変数同定と呼び，与えられたアルゴリズムを実現するために必要な変数を，学生の作成したプログラムから探し出すことを行う．第４は，プロセスの同定である．これは，正規化されたプログラムの各セグメントとHPGの各プロセスとの対応付けを行う．完全に対応がとれたとき理解出来たとする．

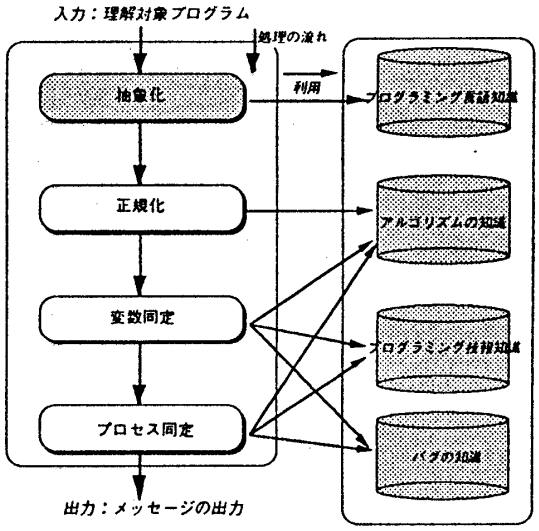


図１　プログラム理解システムALPUS

3．抽象化の考え方

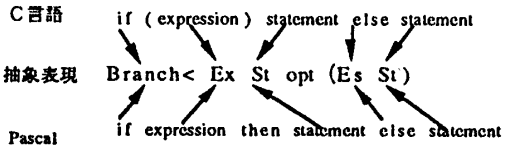
プログラミング言語に関する知識は，プログラミング言語に共通の言語要素である基本的プログラム文に関する概念的知識と個別のプログラミング言語に関する知識の２段の階層をしていると考えられる．１つのプログラミング言語を習得すれば，２番目以降の言語の修得が容易になること，複数の言語を知っているプログラマは時々表記上の混乱を起こすことがあることから，このような記憶構造が推察できる．ALPUSの理解は，プログラム言語を通してプログラマが表現している内容の理解を行うことを考慮すると，ソースコードにおける情報はそのまま残されることが望ましい．以上により，PascalやC言語と同様な命令型言語の形をとる表現を定義し，これに変換することを抽象化とする．基本的に，抽象表現と各プログラミング言語における表現は１対１対応する．また，主にPascalとＣ言語の共通，又は，類似なものとして扱えるものについて対象とする．

Abstraction of program representation to improve the flexibility of program understanding system
Tetsuya OGIHARA Haruki UENO
Department of systems Engineering,Tokyo Denki University
Isizaka,Hatoyama,Saitama,350-03,Japan

4．抽象表現の概要

＜制御文の抽象表現＞

制御文は，if文，for文，前判定繰り返し，後判定繰り返し，を対象にする．次に，if文の抽象表現の例を示す．



＜手続き，関数の抽象表現＞

手続き（procedure），関数は，サブプログラムとして表現する．C言語のvoid型の関数を手続きと同様に見れるようにするためである．

＜入出力関数の抽象表現＞

入出力関数については，Pascalではread，write関数，C言語では，標準入出力関数であるscanf，printfを対象とし，抽象表現を置く．

＜型定義，定数定義の抽象表現＞

型定義は，Pascalのtype，C言語のtypedef，定数定義は，Pascal，C言語ともにconstに相当する表現を置く．

＜プログラム宣言の抽象表現＞

Pascalでは，プログラム宣言部がある．C言語では，Pascalのプログラム宣言文はない．しかし，必要な関数としてmain関数があるので，これを対応させる．

＜データ型の抽象表現＞

基本データ型といえるものには，整数型，実数型，無指定型を用意している．また，構造体型といえるものには，配列，Pascalにおけるrecord型，C言語におけるstructに相当するものを置く．

＜特殊記号の抽象表現＞

複文表記，代入，算術演算子，関係演算子，論理演算子に対応するものを置く．

5．システムの実装

抽象化モジュールでは，プログラミング言語共通の概念と各プログラミング言語の知識を利用し抽象化を行う[図2]．各知識の記述は，図3，図4に示す．

ALPUSは，プログラムコードを評価できる最小単位を1リストに区切り入力データとする．変換の手順は，入力データから1リスト（1行）取り出しプログラミング言語知識のキーワードの値をのぞき，取り出したリストがプログラミング言語のどの概念であるかを特定する．特定できないものは，テンプレートとの比較によって特定する．次に，共通概念知識へのポインタにより，共通概念知識を参照する．共通概念知識のテンプレートと変換ルールに基づき抽象化表現へ変換する．以上の操作をリストがなくなるまで繰り返す．また，式については随時，演算子が前に来る前置型へ変換し，C言語の簡易表現は，その都度is_equivalent_toの値を参照し，具体表現へ変換を行う．

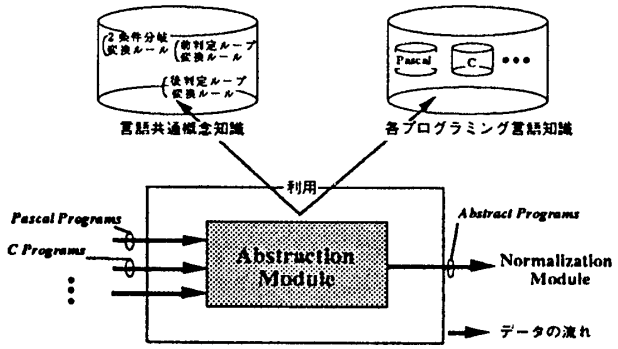


図2　抽象化モジュール

Slot name	Type	Value
Template	LIST	プログラミング言語共通概念知識のテンプレート
C	FLIST	C言語知識へのポインタ
Pascal	FLIST	Pascal知識へのポインタ
C-TranslateRules	LIST	C言語からプログラミング言語共通概念知識への変換規則
Pascal-TranslateRules	LIST	Pascalからプログラミング言語共通概念知識への変換規則

図3　プログラミング言語共通概念知識の記述

Slot name	Type	Value
Template	LIST	各プログラミング言語における知識のテンプレート
Keywords	LIST	テンプレートを特徴づけるキーワード
LanguageCommonConcept	FRAME	プログラミング言語共通概念知識へのポインタ
is_equivalent_to	LIST	簡易表現の具体表現の記述

図4　各プログラミング言語における知識の記述

6．おわりに

本稿では，プログラム理解システムの汎用化を目的としたプログラムの抽象化について報告した．

ここで行った抽象化は，表現の抽象化，見た目の抽象化といえるものである．振る舞いには各プログラミング言語特有なものがある．今後，C言語プログラムの理解を実現するために，プログラムデータを収集し分析，検討が必要となる．特に，C言語の特徴ともいえるポインタによるアセンブラライクな表現への対応の検討が必要であり，抽象表現の拡張が必要である．

＜参考文献＞

[1] Ueno.H.：Integrated Intelligent Programming Environment for Learning Programming, IEICE, Trans. on Information and Systems, vol.E77-D, no.1, 1994

[2] 上野晴樹：プログラム理解システムALPUSの考え方と方法，人工知能研究会資料，SIG-KBS-8903-3