

アセンブラプログラミング演習におけるチャンクとプログラムスライスに基づく答案プログラム分析システム※

立岩佑一郎^{†a)} 吉田 裕一^{†*} 山本 大介[†] 高橋 直久[†]

A System for Analyzing Students' Program by Chunks and Program Slices in Assembly Programming Exercise※

Yuichiro TATEIWA^{†a)}, Hirokazu YOSHIDA^{†*}, Daisuke YAMAMOTO[†],
and Naohisa TAKAHASHI[†]

あらまし 我々はこれまでにアセンブラプログラミング演習において答案プログラムの自動正誤判定機能を有するプログラミング演習システムを開発してきた。この機能は、答案プログラムが制御構造、命令語、及び計算機リソースに関する問題文の要求を満たすかを判定するものである。しかし、アセンブラシミュレータのステップ実行機能の提供だけでは、誤り原因を特定できない学習者が存在する。そこで、次の三つの機能をもつ答案プログラム分析システムを開発した；プログラムから制御構造や意味のある命令系列を抽出し表示する機能、プログラムのステップ実行結果から制御構造の特徴を表す箇所と、問題文で指定された命令語と計算機リソースの変化を表す箇所を抽出する機能、命令語と計算機リソースの要求に関する誤りの原因を含む可能性のある命令系列を求める機能。演習において開発したシステムの評価を行った結果、誤り原因の特定に一定の効果が認められた。

キーワード アセンブラプログラミング、プログラムスライス、チャンク、デバッグ支援

1. ま え が き

筆者らの実施している授業「システムプログラム」は、計算機リソース（レジスタ、主記憶）の制御及び制御構造（順次、反復、分岐、関数・手続き）において、アプリケーションソフトウェアの要求に対するハードウェアの動作の理解を目的としている。この授業は、高水準言語で記述されたプログラムを低水準言語で書き直す方法の講義と C 言語のプログラムをアセンブリ言語 CASL II [1] で書き直すアセンブラプログラミング演習から構成される。学習者は、CASL II プログラミングを未体験であり、演習問題の読解に必要な C 言語の知識をもっている。学習者は、C 言語によるアプリケーションソフトウェアの要求をアセンブリ言語によるハードウェアの動作と関連づける。このほか、例えば関数の処理やメモリの処理など、C 言語とアセン

ブリ言語との対応を理解することで、C 言語の理解を深めることも期待されている。

演習では、先述の制御構造を用いた構造化プログラミングを行う。問題文にはプログラムの制御構造、及び命令語と計算機リソースの使用法が文章と C 言語のプログラムで記述されている。

学習者の答案プログラム（以降、答案と略す）の評価では、以下の 2 種類の誤りがないかを調べる。

- 答案が問題文で指定されたとおりに制御構造を実現していない（制御構造誤り）
- 答案が問題文で指定されたとおりに命令語と計算機リソースを使用していない（命令語・計算機リソース誤り）

我々はこれまでに、これらの誤りの有無を自動的に判定する機能を有するアセンブラプログラミング演習システムを開発した [2], [3]。この機能により、学習者は答案の正誤評価を即座に得られるようになった。この機能は、不正解の学習者に対して、判定（不正解）と判定に用いたシミュレータ動作コマンド系列を表示する。シミュレータ動作コマンド系列とは、CASL II シミュレータを自動実行するためのもので、プログラ

[†] 名古屋工業大学, 名古屋市
Nagoya Institute of Technology, Gokiso-cho, Showa-ku,
Nagoya-shi, 466-8555 Japan

* 現在, パナソニックアドバンストテクノロジー株式会社

a) E-mail: tateiwa@nitech.ac.jp

※ 本論文は教育工学研究専門委員会推薦論文である。

ム実行にあたっての CASL II シミュレータの初期状態（レジスタやプログラムカウンタなど）やステップ実行回数などである。

不正解の学習者は答案、問題文、及びシミュレータ動作コマンド系列により、答案の誤り原因の特定を試みる。制御構造誤りの原因特定のために、問題文の C プログラムの制御構造と答案の制御構造を手作業で分析する。命令語・計算機リソース誤りの原因特定のために、答案の実行過程を手作業や CASL II シミュレータのステップ実行により分析したり、命令の依存関係を手作業で分析したりする。分析結果を参考に、答案から問題文の指定を満たすと考えられる箇所を取り除き、満たしていない箇所（すなわち、誤り原因）を特定する。学習者はこのような誤り原因の探索とその修正を繰り返すことで、講義内容を定着させていく。しかし、講義内容の多くが定着していない、答案の分析ミスに気づかない、または C プログラムの分析ミスに気づかない学習者は、上述の作業を正しく効果的に行えないため、誤り原因を特定できず、演習をあきらめてしまう。

そこで、問題文と答案における制御構造、命令語と計算機リソースの変化、及び命令の依存関係を誤り原因の探索に役立つような形で学習者に示す。このため、下記の三つの機能（あるいは特徴）を有する答案分析システムを提案する。これにより、学習者が答案において問題文の指定を満たしている箇所と満たしていない箇所の判別をしやすくなることで、つまり学習者を減らすことが期待される。また、自身による答案分析の結果検証へ用いることで、講義内容において未定着の箇所を効率的に特定できるようにもなる。

(1) 静的チャンク抽出機能：本機能は、答案と正解例プログラム（以降、正解例と呼ぶ）から制御構造など意味のある命令系列（静的チャンクと呼ぶ）を抽出して、それらを比較表示する（この表示を静的チャンク表示と呼ぶ）。

(2) 振舞い抽出機能：本機能は、答案と正解例の詳細なステップ実行結果から、制御構造の特徴を表す箇所と、問題文で指定された命令語と計算機リソースの変化を表す箇所を抽出し（これら二つの箇所を振舞いと呼ぶ）、それらを比較表示する（この表示を振舞い表示と呼ぶ）。

(3) 誤り原因絞込み機能：本機能は、答案と正解例へのスライシング技術の適用と、答案と正解例の特定の実行過程の照合により、命令語・計算機リソース誤りの原

因を含む可能性のある静的チャンクと命令系列を求め、それを表示する（この表示を誤り潜在域表示と呼ぶ）。

2. 演習問題例と誤り答案例

図 1 に示す問題は、二つの引数を受け取り、加算した結果を *GR1* に格納するという関数 *sum* の作成に関するものである。アセンブラプログラムの制御構造、及び命令語と計算機リソースの使用法が、文章及び C プログラムにより指定されている。

この問題に対する正解例、及び二つの不正解答案を図 2 に示す。制御構造誤りのプログラム (*control structure error*) は、正解例 (*correct answer*) の 2 行目と 3 行目に示された命令（関数の開始処理）が誤って記述され、4 行目と 5 行目の命令（引数の読出し処理）が記述されていない。命令語・計算機リソース誤りのプログラム (*operation code and computer resources error*) は、制御構造は正しいが、引数の読出し処理である 4 行目と 5 行目の LD 命令の第 2 オペランドが誤っている。

3. 答案プログラム分析システム

3.1 システムの概要

本論文では、次のように呼称を定義する。ある命令 *i* をステップ実行して得られたステップデータ *s* があるとき、*i* を *s* の素となる命令と呼ぶ。ある静的チャンクに含まれる命令を素とするステップデータ系列を動的チャンクと呼ぶ。静的チャンク *sc* に含まれる命令を素とするステップデータ系列の動的チャンク *dc* があるとき、*sc* を *dc* の素となる静的チャンクと呼ぶ。動的チャンクにおいて値が更新された計算機リソースを動的チャンクの出力と呼ぶ。

提案システムは図 3 のように、静的チャンク抽出機能 (Function for extracting static chunks)、振舞い抽出機能 (Function for extracting behavior)、及び誤り原因絞込み機能 (Function for calculating potential fault region) から構成される。本論文では、プログラムから抽出したい静的チャンクの抽出条件を静的チャンク条件 (Static chunk condition) と呼ぶ。Answers は答案、Correct answers は正解例である。Step data (answer) は、答案のステップデータ系列、Step data (correct answer) は正解例のステップデータ系列である。Collating resources は、動的チャンクの出力のうち、その値が振舞いに影響されるもの（以降、照合リソースと呼ぶ）である。Replaced step data (answer)

次に示す C プログラムで表される関数 `sum` を CASL プログラムで実現しなさい
プログラム実行前に引数は下図のようにスタックに積まれているものとする

<code>int sum(int x, int y)</code>	Offset	Stack	
{		...	
<code>int t = x+y;</code>	2	y	
<code>return t;</code>	1	x	
}	0	Rtn Adr ← SP	

ただし、次に示す項目を満たした CASL プログラムを作成すること。

- ・戻り値は **GR1** に入れて渡す
- ・**GR7** をスタックフレームのポインタとして用いる
- ・**GR1** を変数 `x` として用いる
- ・**GR2** を変数 `y` として用いる
- ・引数の読み出し及び、関数の復帰処理はスタックフレームのポインタを用いて実現する
- ・引数読み出し時のスタックフレームのポインタ退避処理は **PUSH 命令** によって実現する
- ・関数復帰処理におけるスタックフレームのポインタ回復処理は **POP 命令** によって実現する
- ・関数 `sum` のラベルは **P11** を用いる

図 1 関数実装の演習問題

Fig. 1 A question on function implementation.

	correct answer	control structure error	operation code and computer resources error
1:	P11 START	P11 START	P11 START
2:	PUSH 0, GR7	SSP GR7	PUSH 0, GR7
3:	SSP GR7	PUSH 0, GR7	SSP GR7
4:	LD GR1, 2, GR7	ADDA GR1, GR2	LD GR1, 1, GR7
5:	LD GR2, 3, GR7	LSP GR7	LD GR2, 2, GR7
6:	ADDA GR1, GR2	POP GR7	ADDA GR1, GR2
7:	LSP GR7	RET	LSP GR7
8:	POP GR7	END	POP GR7
9:	RET		RET
10:	END		END

図 2 図 1 の問題に対する正解例と不正解答案

Fig. 2 A correct answer and incorrect answers of the question in Fig. 1.

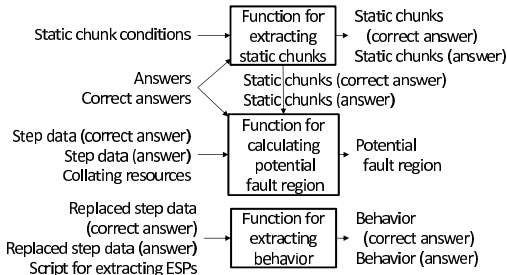


図 3 システム構造

Fig. 3 System overview.

と `Replaced step data (correct answer)` は、答案のステップデータと正解例のステップデータのアセンブラシンボルを、問題文の C プログラムの高級言語シンボルに置き換えたもの（以降、シンボル置換え後ステップデータ系列）である。Script for extracting ESPs は、シンボル置換え後ステップデータ系列から、振舞いとして抽出するシンボル置換え後ステップデータ系列の条件（以降、拡張構造パターン抽出スクリプトと呼ぶ）である。

静的チャンク抽出機能は、静的チャンク条件に基づくパターンマッチングにより答案と正解例から静的チャンク (Static chunks (answer), Static chunks (correct answer)) を抽出する。答案が制御構造誤りであるかを判定し、制御構造誤りであれば答案と正解例の静的チャンクを表示する（静的チャンク表示）。

振舞い抽出機能は、拡張構造パターン抽出スクリプトに基づくパターンマッチングにより、答案及び正解例のシンボル置換え後ステップデータ系列から、振舞いを表すもの（以降、拡張構造パターンと呼ぶ）を抽出する。そして、拡張構造パターンを答案と正解例とで照合することで、値に差異のある計算機リソースを求める。値に差のある計算機リソースが見つかった場合、答案と正解例の振舞い (Behavior (answer), Behavior (correct answer)) を表示する（振舞い表示）。

あるプログラムにおいて命令語・計算機リソース誤りが発生しているとする。この誤りを解消するために追加する命令の挿入先の候補、及び修正または削除する命令の候補を「誤り潜在域 (Potential fault region)」と呼ぶ。誤り原因絞込み機能は、答案、正解例、答案と正解例のステップデータ系列、静的チャンク系列 (static chunks), 及び照合リソースを用いて、プログラムスライス及び動的チャンクを抽出・分析することで、誤りの原因を含む可能性のある静的チャンクと命令系列を誤り潜在域として求め、これを表示する（誤り潜在域表示）。

3.2 静的チャンク抽出機能

学習者は、制御構造誤りのある答案のアセンブラプログラム及び問題文の C プログラムから制御構造を手

作業で抽出し、それらを比較することで制御構造誤りの原因を絞り込む。学習者が原因を特定し、それを修正し、その答案をシステムにより正解と判定されることで、アセンブラプログラミングの制御構造に関する知識を定着させる。しかし、これらの作業を正しく効果的に行えずに、つまづいてしまう学習者が存在する。

本機能は、答案と正解例から静的チャンクを静的チャンク条件に基づいて抽出し、答案が制御構造誤りであるかを判定し、制御構造誤りであれば答案の命令系列と静的チャンク、及び正解例の静的チャンク名を表示する（静的チャンク表示）。静的チャンクは、制御構造以外にも、ある制御構造を実現する上で特定の役割をもつ命令系列や、典型的な実装誤りをもつ命令系列などである。例えば、関数の開始時に引数をレジスタに読み出す命令系列を一つのチャンクとして抽出して提示することが可能である。制御構造を細分化し、実装誤りを指摘することで、学習者による探索範囲を狭められる。

静的チャンク条件は、静的チャンク名と抽出する命令系列の条件（以降、命令系列条件と呼ぶ）から構成される。システム開発者は、一般的なアセンブラプログラミングに共通した静的チャンク（例えば、制御構造）の条件をシステム開発時に定義しておく。本システムの開発時において、「順次処理」、「関数の開始処理」、「関数の終了処理」、「引数の読み出し処理」、「繰り返し処理」及び「分岐処理」が定義された。また、教師は学習者にヒントとして提示したい静的チャンク（例えば、典型的な実装誤りの静的チャンク）の条件をシステム運用時に定義できる。演習実施に必要な静的チャンク条件（制御構造）はシステム開発者により定義されるため、教師は新たに静的チャンク条件を定義することなく演習を実施できる。本システムの運用時において「関数の開始処理（誤りあり）」、「関数の終了処理（誤りあり）」及び「引数の読み出し処理（誤りあり）」が定義された。

正解例が問題文で指定された制御構造に従っているため、正解例から抽出した静的チャンクは、問題文で指定された制御構造を含んでいる。学習者は、答案から制御構造を手作業で抽出し、それをシステムにより提示された答案の制御構造及び正解例の制御構造と比較することで、アセンブラプログラミングの制御構造に関する未定着箇所を発見する。また、学習者は、システムにより提示された答案の静的チャンクと正解例の静的チャンクとを比較し両者の差異を見つけること

答案プログラム

SSP GR7	関数の開始処理(誤りあり)
PUSH 0,GR7	
ADDA GR1,GR2	順次処理
LSP GR7	
POP GR7	関数の終了処理
RET	

正解例プログラム

(1)	関数の開始処理
	引数の読み出し
(2)	順次処理
	関数の終了処理

図 4 静的チャンク表示の例

Fig.4 An example of static chunks.

で、制御構造誤りの原因を絞り込む。

図 2 の制御構造誤りのプログラムに対して、本システムは図 4 に示す静的チャンク表示を生成する。図 4 は、答案のプログラム、答案の静的チャンク、及び正解例の静的チャンク名を表示している。図中の「正解例プログラム」に示されている静的チャンク名の並びは、問題文で指定された制御構造を満たすものである。この並びと異なる答案は、制御構造誤りとシステムに判定される。図中 (2) は、引数の読み出しチャンクが答案に不足していることを示している。図中 (1) は、答案の 2 行目と 3 行目が「関数の開始処理（誤りあり）」に適合しており、関数の開始処理の実現の誤りが当該の 2 行に潜在することを示している。もし、「関数の開始処理（誤りあり）」が定義されていない場合、この 2 行は次の 3 行目と合わせて「順次処理」となる。しかし、このように学習者のミスとして指摘することで、学習者は誤り原因を特定しやすくなる。

3.3 振舞い抽出機能

学習者は、答案のアセンブラプログラム及び問題文の C プログラムのステップデータ系列を手作業や CASL II シミュレータのステップ実行により求め、それら系列から振舞いを表す箇所を求めて比較することで誤りの原因を絞り込む。学習者が原因を特定し、それを修正し、その答案がシステムにより正解と判定されることで、アセンブラプログラミングのステップデータに関する知識を定着させる。しかし、これらの作業を正しく効果的に行えずに、つまづいてしまう学習者が存在する。

本機能は、答案と正解例の振舞いを表すステップデータの素となる命令、その命令の実行により更新された計算機リソースの名前とその値、及びその命令の意味（制御構造の特徴を表すのか、命令語と計算機リソースの変化を表すのか）を振舞いとして学習者に表示する。正解例が問題文での指定を満たすものであるため、正解例のステップデータ系列は問題文で指定された振舞いを含んでいる。

学習者は、答案のアセンブラプログラム及び問題文の C プログラムからステップデータを求め、それらから振舞いを表す箇所を求める。そして、それら振舞いを表す箇所とシステムにより提供された答案の振舞い及び正解例の振舞いとを比較することで、アセンブラプログラミングのステップデータに関する未定着箇所を発見する。また、学習者は、システムにより提供された答案の振舞いと正解例の振舞いとを比較し両者の差異を見つけることで、命令語・計算機リソース誤りの原因を絞り込む。

アセンブリ構造誤り検出システム ASADS [2] は、問題文で指定したアセンブラシンボルと C プログラムで指定した高級言語シンボルの対応表（以降、シンボル対応表と呼ぶ）に基づき、ステップデータ中のアセンブラシンボルを高級言語シンボルに置き換えたステップデータ（シンボル置換え後ステップデータと呼んでいる）を作成する。そして、問題文で指定された振舞いを表すシンボル置換え後ステップデータ系列を構造パターンと呼び、これを教師の指定（構造パターン抽出スクリプト）に基づいてステップデータ系列から抽出する。

本研究では、構造パターン抽出スクリプトを拡張することで（拡張構造パターン抽出スクリプトと呼んでいる）、問題文で指定された振舞いを表すシンボル置換え後ステップデータにコメント文字列を対応づけて抽出できるようにした。ここで、この抽出したシンボル置換え後ステップデータとコメント文字列の系列を拡張構造パターンと呼ぶこととする。これにより、教師は、このコメント文字列にステップデータの素となる命令の意味を指定できる。

拡張構造パターンは、シンボル置換え後ステップデータ系列と拡張構造パターン抽出スクリプトのパターンマッチングにより抽出される。このマッチングでは、拡張構造パターン抽出スクリプトの構造パターン抽出スクリプト部分の要素—コメント文字列以外の要素—が用いられる。このため、マッチング方法は文献 [2] の第 4 章を参照されたい。

図 5 は振舞い表示で、図 2 の命令語・計算機リソース誤りの答案の振舞い（図 5 左列）と問題文の指定する振舞い（図 5 右列）を表示している。ステップデータの素となる命令の一部（例えば、13 行目）、計算機リソースの値（例えば、19 行目）、及び命令の意味（例えば、17 行目）が示されている。

学習者は、答案と問題文の指定する振舞いの違いを

[[あなたの答案]]	[[正解例]]
1行目:スタックフレームのポインタを用いた関数の読み出し処理	1行目:スタックフレームのポインタを用いた関数の読み出し処理
2行目:PUSH	2行目:PUSH
3行目:SP[00FF]	3行目:SP[00FF]
4行目:	4行目:
5行目:SSP GR7	5行目:SSP GR7
6行目:GR7[00FF]	6行目:GR7[00FF]
7行目:	7行目:
8行目:変数xの読み出し	8行目:変数xの読み出し
9行目:LD GR1	9行目:LD GR1
10行目:x=0120	10行目:x=0004
11行目:	11行目:
12行目:変数yの読み出し	12行目:変数yの読み出し
13行目:LD GR2	13行目:LD GR2
14行目:y=0004	14行目:y=0003
15行目:	15行目:
16行目:	16行目:
17行目:スタックフレームのポインタを用いた関数の復帰処理	17行目:スタックフレームのポインタを用いた関数の復帰処理
18行目:LSP GR7	18行目:LSP GR7
19行目:SP[00FF]	19行目:SP[00FF]
20行目:	20行目:
21行目:POP GR7	21行目:POP GR7
22行目:SP[0100]	22行目:SP[0100]
23行目:	23行目:
24行目:RET	24行目:RET
25行目:SP[0101]	25行目:SP[0101]
26行目:	26行目:
27行目:実行結果表示	27行目:実行結果表示
28行目:SVC	28行目:SVC
29行目:x=0124	29行目:x=0007
30行目:y=0004	30行目:y=0003
31行目:	31行目:
32行目:	32行目:
目:*****	目:*****
[出力結果ここまで]	[出力結果ここまで]

図 5 振舞い表示の例

Fig. 5 An example of behavior.

確認することで誤り原因を特定する。図 5 の 1 行目にて「関数の開始処理」が開始されたことが示されている。図 5 の 8 行目と 12 行目で「引数の読出し処理」が示唆されており、17 行目にて「関数の終了処理」が開始されたことが示されている。そして、図 5 の 10 行目に答案と正解例の振舞いに違いが発生していることを読み取れる。これにより、誤り原因はプログラム（図 2）の 4 行目以前に存在することが分かる。

答案と正解例とで値の異なったレジスタ GR1 に影響する命令系列は、プログラムの 2~4 行目となる。また、GR7 は図 5 の 6 行目にて答案と正解例で値が等しい。以上により、学習者は誤り原因を「GR7 に影響のある命令がプログラム 3~4 行目の間に不足する」、「GR1 に影響のある命令がプログラム 4 行目以前に不足する」、及び「プログラム 4 行目の命令が誤っている」に絞ることができる。

3.4 誤り原因絞り込み機能

ダイナミックスライスはプログラムスライスの一種で、プログラムにある入力を与えて実行したときに、ある命令の実行により更新された変数に影響を及ぼす命令の系列である。本論文では、ステップデータ s において値の更新された変数 w があるとき、 (s, w) をス

ライシング基準という。

ある変数のプログラムスライスは、その一部に別の変数のプログラムスライスを含むことがある。前者のスライスから後者のスライスを取り除いたスライス—二つのスライスの差分—は、ダイスと呼ばれている [4]。本論文では、ダイナミックスライスから求めたダイスをダイナミックダイスと呼ぶこととする。

制御構造誤りがなく命令語・計算機リソース誤りのある答案のアセンブラプログラム及び問題文の C プログラムのステップデータにおいて、問題文で使用方法の指定された計算機リソースの値を照合することで、答案のステップデータを正しいステップデータ、システムにより誤りが検出されたステップデータ、どちらともいえないステップデータに分類可能である。また、それらのステップデータにおいて照合した計算機リソースに影響を与える命令系列を求めることで、答案の命令を正しい命令、誤っている命令、どちらともいえない命令に分類できる。学習者が原因を特定し、それを修正し、その答案がシステムにより正解と判定されることで、命令の依存関係に関する知識を定着させる。しかし、これらの作業を正しく効果的に行えずに、つまづいてしまう学習者が存在する。

本機能は、答案と正解例のステップデータの照合において、問題文で使用方法の指定された計算機リソースの値に差異の現れたステップデータとその計算機リソースをスライシング基準とするダイナミックスライスと、それ以前に照合した差異のないステップデータと計算機リソースをスライシング基準とするダイナミックスライスを求め、前者から後者を除外することで、ダイナミックダイスを求める。そして、このダイナミックダイス、差異の現れた計算機リソース、及び照合した計算機リソースを誤っている可能性の高い命令系列として学習者に提示する。

また、このダイナミックダイスは、ダイス中に誤った命令が存在するか、正しい命令が不足しているか、それらの両方かに分類できる。このため学習者は、誤った命令をダイスから探し、正しい命令の挿入箇所をプログラムから探すことになる。そこで、本機能は、正しい命令の挿入箇所の探索範囲を絞り込むために、静的チャンクを正しいチャンク、誤っている可能性のあるチャンク、及び誤っているチャンクに分類する。正しいチャンクは、チャンクの出力が全て照合リソースであり、それらが答案と正解例とで一致する。誤っている可能性のあるチャンクは、チャンクの出力に照合

リソース以外の計算機リソースが含まれており、照合リソースは答案と正解例とで一致する。誤っているチャンクは、従来法で答案と正解例とで不一致の計算機リソースを出力とするステップデータの素となる命令を含むものである。したがって、探索範囲は誤っている可能性のあるチャンクと誤っているチャンクに絞り込まれる。本機能は、誤っている可能性のあるチャンク、及び誤っているチャンクを、誤りの原因を含む静的チャンクとして学習者に提示する。

学習者は、答案のアセンブラプログラム及び問題文の C プログラムからステップデータを求め、それらから振舞いを表す箇所を求める。両者を比較し、差異の現れた計算機リソースと差異の現れなかった計算機リソースの各々に影響のある命令系列を求め、前者の命令系列から後者の命令系列を除外することで誤っている可能性の高い命令系列を求める。この命令系列とシステムの提示した誤っている可能性の高い命令系列とを比較することで、命令の依存関係に関する未定着箇所を発見する。また、システムの提示した誤っている可能性の高い命令系列と、誤っている可能性のあるチャンク、及び誤っているチャンクにより、命令語・計算機リソース誤りの原因を特定する。

図 6 は図 2 に示す命令語・計算機リソース誤りのプログラムに対する誤り潜在域表示である。「答案プログラムの実行の流れ」の列では、プログラムの命令と静的チャンクが表示され、「実行結果」の列では各静的チャンクでの最後の命令の実行完了時における問題文で使用方法の指定された計算機リソースのうち、命令語・計算機リソース誤りが検出されるまでに最後の命令が実行完了した静的チャンクまでの計算機リソースの値が表示されている。「正解例プログラムの実行の流れ」の列の静的チャンクの並びは、問題文で指定された制御構造を満たすものである。「正解例の実行結果」の列の計算機リソースの値は、答案プログラムのもと同じ意味である。

図中 (1)～(4) の指す枠内の文字列は赤字である。赤

答案プログラムの実行の流れ	実行結果	正解例プログラムの実行の流れ	正解例の実行結果
PUSH 0, GR7		関数の開始処理	GR7 0x04
CALL GR7		関数の読み出し	GR1 0x04
ADDI GR1, GR2		順次処理	GR2 0x04
LSP GR7		関数の終了処理	
POP GR7			
RET			

図 6 誤り潜在域表示の例
Fig. 6 An example of potential fault region.

字の命令系列（図中 (1)）は誤っていることを意味し、赤字のチャンク名（図中 (2)）は誤っているチャンク若しくは誤っている可能性のあるチャンクであることを意味する．これは、図中 (1) の修正か図中 (2) へ新しい命令の追加により誤りを修正できることを意味する．また、図中 (3) は誤っている変数とその値、図中 (4) は正解例における同変数とその値である．図 5 による命令語・計算機リソース誤りの原因特定の手続きと比較すると、赤字に誤り原因の潜在域が限定されていることと、それらが自身の答案プログラムに直接的に示されていることにより、誤り原因の絞込みが簡単である．

4. システムの実現法

本論文では、データ構造をタプルで表記し、タプルの各要素の参照はドット表記を用いる．例えば、データ構造 $X = (y, z)$ の変数 w において、 y を参照する場合には $w.y$ と表す．また、配列の添字は 1 から始まる．そして、次の関数を定義する．文字を要素とする配列 A の要素 $A[i]$ から要素 $A[j]$ ($i \leq j$ とする) を i から j へ向かって連結した文字列を返す関数を $\text{substr}(A, i, j)$ と定義する．文字列 str_1 の末尾に文字列 str_2 を連結した文字列を返す関数を $\text{strjoin}(\text{str}_1, \text{str}_2)$ と定義する．配列 A の末尾要素の添字を返す関数を $\text{tail}(A)$ と定義する．

4.1 静的チャンク抽出法

プログラムの識別子及び命令数は問題文によって異なるため、CASL II の命令で命令系列条件を記述すると、その数が膨大になるという問題点がある．本論文では、以下の要件を満たすパターンマッチングによりこの問題の解決を図る．

[要件 1] パターン文字列内の特定の部分にマッチした命令系列を抽出できること．抽出したい命令系列の特徴として、その前後の命令を特徴として利用することが想定され得る．例えば、分岐の命令系列の終端は、分岐命令から分岐命令のオペランドに指定されたラベルをもつ命令の直前の命令までである．図 7 では、1～4 行目が分岐処理チャンクとなる．

[要件 2] 任意かつ同一の識別子が複数箇所に出現する命令系列を抽出できること．例えば、分岐の命令系列では、分岐命令のオペランドと、分岐先の命令のラベルが任意かつ同一の文字列となる．図 7 では、ラベル L1 は学習者により自由に命名されたもので、2, 3, 5 行目に出現している．

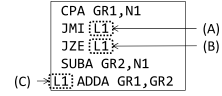


図 7 分岐処理の命令系列の例

Fig. 7 An example of instructions for selection processing.

要件 1 を次のように関数 $f1$ で表す．CASL II の識別子と区切り文字を 1 トークンとする正規表現のパターン文字列を要素とする配列を R とする． (a, b) が返されるとき、 $\text{substr}(R, 1, \text{tail}(R))$ の実行結果をプログラムの表現文字列 P の先頭から末尾に向かって最短マッチングし、最初にマッチした文字列において $\text{substr}(R, x, y)$ にマッチしている文字列が $\text{substr}(P, a, b)$ となる．

$$f1(P, R, x, y) = \begin{cases} (a, b) & (\text{マッチ成功}) \\ NULL & (\text{マッチ失敗}) \end{cases}$$

例えば、図 7 を含むプログラムから分岐処理の命令系列 (1～4 行目) を抽出することを考える． $R[j]$ が 1 行目の CPA に、 $R[k]$ が 4 行目の N1 に、 $R[l]$ が 5 行目の GR2 にマッチするパターン文字列を要素とする配列を R とする．プログラムの表現文字列 P において、 $P[a] = C$ (1 行目 CPA の C) で $P[b] = 1$ (4 行目 N1 の 1) のとき、 $f1(P, R, j, k)$ は (a, b) を返す．

要件 2 を次のように関数 $f2$ で表す． P と R は関数 $f1$ と同じ意味であり、 $R[x]$ と $R[y]$ にマッチした文字列同士が等しいことを指定する整数 x と y (ただし、 $x < y$) による組 (x, y) を要素とする集合を G とする．

$$f2(P, R, G) = \begin{cases} true & (G \text{ の全ての要素 } g \text{ に対して、} f1(P, R, g.x, g.x) \\ & = f1(P, R, g.y, g.y) \text{ の場合}) \\ false & (\text{それ以外}) \end{cases}$$

例えば、図 7 を含むプログラムから分岐処理の命令系列 (1～4 行目) を抽出することを考える． $R[j]$ が図 7 中 (A) の識別子に、 $R[k]$ が (B) の識別子に、及び $R[l]$ が (C) の識別子にマッチするパターン文字列を要素とする配列 R を考える．このとき、 G に (j, k) 、 (j, l) を要素として与えることで、(A) と (B) と (C) の識別子が異なるとき $f2(P, R, G)$ は $true$ を返さずに $false$ を返す．

要件 1 と 2 を満たすパターンマッチ関数 match を

次のように定義する． P, R, x, y は関数 $f1$ と， G は関数 $f2$ と同じ意味である． (a, b) が返されるとき， $f1(P, R, x, y)$ の実行結果 $F1$ において ($F1.a = a$ かつ $F1.b = b$) が真である．

$$\text{match}(P, R, x, y, G) = \begin{cases} (a, b) & (G \text{ の全ての要素 } g \text{ に対して, } x \leq g.x < g.y \leq y \text{ かつ } f2(\text{substr}(P, F1.a, F1.b), R, G) = \text{true} \text{ の場合}) \\ \text{NULL} & (\text{それ以外}) \end{cases}$$

関数 match の実現には，perl [6] のパターンマッチングを次のように用いる． G の要素を g とするとき， $R[g.x]$ をグループ (g_x とする) に指定し， $R[g.y]$ を g_x へのグループの後方参照により指定した R を全ての g において作成する．次に， $r_{xy} = \text{substr}(R, x, y)$ の実行後に， r_{xy} をグループ (g_{xy} とする) に指定する．そして， $\text{regex} = \text{strjoin}(\text{strjoin}(\text{substr}(R, 1, x-1), r_{xy}), \text{substr}(R, y+1, \text{tail}(R)))$ の実行後に， regex を最短一致に指定したパターン文字列 regex' を作成する． P に対して regex' のマッチングを行い， g_{xy} へのグループの後方参照の値 J を求める． J が空文字列であるならば NULL を戻り値として返す．そうでないならば， $\text{substr}(P, a, b) = J$ となる a と b を見つけ， (a, b) を戻り値として返す．

また，答案から静的チャンクを抽出するためには，次の点を満たす必要がある．

[要件3] 制御構造の実装の誤った答案からも静的チャンクを抽出できること．答案には，制御構造が正しく実装されたものだけでなく，誤って実装されたものもある．このため，正しい制御構造の命令系列条件だけでは，静的チャンクを抽出できない答案が存在する．

[要件4] 複数の命令系列条件にマッチする命令系列があったとき，抽出する命令系列を選択できること．

要件3への対応のため，システム運用時に教師が新しい静的チャンク条件を定義できるようにした．3.2で述べたように，本システムに定義されている静的チャンク条件は，システム開発時に定義された六つと，システム運用時に定義された三つである．

要件4への対応のため，静的チャンク条件に優先度の項を設ける．命令系列がある二つの静的チャンク条件に重複してマッチしたとき，その命令系列は優先度の高い方の静的チャンクとし，優先度が同じであれば両方の静的チャンクとする．

本演習で作成されるプログラムは，構造化プログラ

ミングに基づいており，また，あるルーチンから別のルーチンへジャンプ命令で飛ぶことが許可されていない．したがって，複数のルーチンにわたった静的チャンクを求める必要がないため，ルーチンごとに独立して求めることでアルゴリズムを単純化する．ここで，CASL II のルーチン（メインルーチンまたはサブルーチン）は，START 命令から END 命令までの命令の系列で，ルーチンのボディは，ルーチン中の機械語命令とマクロ命令の系列とする．

下記のアルゴリズムにより，ルーチンボディの表現文字列 P' から静的チャンクを抽出する．静的チャンク条件は，静的チャンク名 name と，命令系列条件としての関数 match の引数 R, x, y, G ，及び優先度 pri により， $(\text{name}, R, x, y, G, \text{pri})$ の六つ組で管理される．静的チャンクは，静的チャンク条件 cc 及び命令系列の文字列表現 $\text{substr}(P', a, b)$ により， (cc, a, b) の三つ組で管理される．順次処理の静的チャンク条件を seq とする． seq 以外の静的チャンク条件を優先度の高い順に格納した配列を CC とする．また，抽出された静的チャンクはリスト SC に格納される．

S1. CC が空でないならば先頭要素を取り出し cc とし，そうでないならば S8 へ進む．

S2. $j = 1$ を実行する．

S3. $j \leq \text{tail}(P')$ が偽であるならば S1 へ戻る．

S4. $\text{match}(\text{substr}(P', j, \text{tail}(P')), cc.R, cc.x, cc.y, cc.G)$ の実行結果 M が NULL であるならば S1 へ戻る．

S5. $a' = M.a + j - 1$ と $b' = M.b + j - 1$ を実行する．

S6. SC の全ての要素 sc に対して， $(b' < sc.a$ または $sc.b < a')$ が真（例えば，図8では， sc_1, sc_2 ，及び sc_3 が抽出されたチャンクであり， (a'_1, b'_1) と (a'_3, b'_3) はこの条件に真となる）であるならば，または SC の要素を sc として $(sc.a \leq a' \leq sc.b \leq b'$ または $a' \leq sc.a \leq sc.b \leq b'$ または $a' \leq sc.a \leq b' \leq sc.b)$ が真（図8の (a'_2, b'_2) はこの条件に真となる）となる全ての sc において $sc.pri = cc.pri$ が真であるならば， (cc, a', b') を SC の末尾に追加する．

S7. $j = b' + 1$ を実行して，S3 へ戻る．

S8. 次の操作によって， CC のどのチャンク条件も満たさない最長の文字列が $\text{substr}(P', k, l)$ の実行結果となる k と l を求める．例えば，図9では， sc_1, sc_2 ，及び sc_3 が抽出されたチャンクであり， $\text{substr}(P', x_1, y_1)$ ， $\text{substr}(P', x_2, y_2)$ ， $\text{substr}(P', x_3, y_3)$ ，及び $\text{substr}(P', x_4, y_4)$ の実行結果は sc_1 と重複している

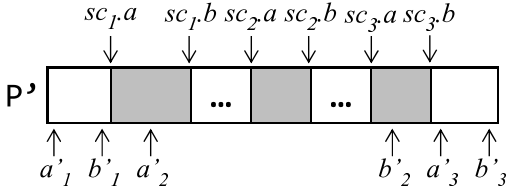


図 8 静的チャンクの抽出例

Fig. 8 An example of extracting static chunks.

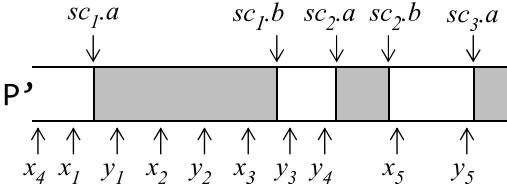


図 9 順次処理チャンクの抽出例

Fig. 9 An example of extracting a sequence processing chunk.

ので求めたい文字列ではなく、 $\text{substr}(P', x_5, y_5)$ の実行結果はどのチャンクとも重複していない（かつ、最長であるとする）ので求めたい文字列である。 SC の全ての要素 sc に対して、 $\neg(x \leq sc.b \leq y)$ かつ $\neg(x \leq sc.a \leq y)$ かつ $\neg(sc.a \leq x \leq sc.b)$ かつ $\neg(sc.a \leq y \leq sc.b)$ かつ $(x \leq y)$ が真となり、かつ $\text{substr}(P', x, y)$ の実行結果が最長となる x と y を探す。

S9. x と y を見つけられたならば、 (seq, x, y) を SC の末尾に追加し、S8 へ戻る。

S10. SC を返す。

4.2 誤り原因絞込み法

ある命令 i をステップ実行して得られたステップデータ s があるとき、 i を $\text{Ins}(s)$ で表す。命令 i の属する静的チャンクの集合を $\text{SChunk}(i)$ で表す。ステップデータ s が得られたステップ実行回を $\text{SI}(s)$ で表す。チャンク先頭のステップデータ h 、チャンク末尾のステップデータ t 、素となる静的チャンク sc により、 (h, t, sc) の三つ組で動的チャンクを管理する。ステップデータがキュー S に格納されているとき、動的チャンクの集合 DC を求める関数 $\text{DChunk}(S)$ は以下のとおりである。

S1. DC を空にする。

S2. S が空でないならば先頭要素を取り出し s とし、そうでないならば S8 へ進む。

S3. $\text{SChunk}(\text{Ins}(s))$ の要素 sc において、 $\text{Ins}(s)$ が sc の先頭命令となる sc の集合 SC を求める。

S4. $hs = s$ 及び $ts = s$ を実行する。

S5. S が空でないならば先頭要素を取り出し s に格納

し、S6 へ進む。そうでないならば S7 へ進む。

S6. $(\text{SChunk}(\text{Ins}(s)) \cap SC)$ が空である、または $(\text{SI}(s) < \text{SI}(ts))$ であるならば、 SC の全ての要素 sc に対して $DC = DC \cup \{(hs, ts, sc)\}$ を実行して S3 へ戻る。そうでないならば、 $ts = s$ を実行して S5 へ戻る。

S7. SC の全ての要素 sc に対して $DC = DC \cup \{(hs, ts, sc)\}$ を実行する。

S8. DC を返す。

動的チャンクの集合 DC の要素 dc 、ステップデータ s において、 $dc.t = s$ を満たす dc の集合を $\text{DCE}(DC, s)$ で表す。動的チャンクの集合 DC とステップデータのキュー S において、 S の先頭から順に出現する s に対する $\text{DCE}(DC, s)$ を格納するキュー E を求める関数 $\text{DCEQ}(DC, S)$ は以下のとおりである。

S1. キュー E を空にする。

S2. S が空でないならば先頭要素を取り出し s とし、そうでないならば S5 へ進む。

S3. $dce = \text{DCE}(DC, s)$ を実行し、 dce が空でないならば、 E の末尾に dce を追加する。

S4. S2 へ戻る。

S5. E を返す。

計算機リソース名 n とその値 v によるデータ構造を $\text{Resrc} = (n, v)$ と定義する。動的チャンクの出力を Resrc にて管理し、動的チャンク dc の出力の集合を $\text{DCOut}(dc)$ で表す。ステップデータ s 以前 (s を含む) で変数 w に最後に値を設定したステップデータを $\text{Def}(s, w)$ で表す。正解例の静的チャンク sc の出力において照合する計算機リソース名の集合を $\text{CmpRN}(sc)$ で表す。制御構造誤りの評価における答案と静的チャンクの並びの比較において、正解例の静的チャンク sc と比較した答案の静的チャンクを $\text{CmpSC}(sc)$ で表す。スライシング基準 C におけるダイナミックスライスに含まれる命令の集合を $\text{DSlice}(C)$ で表す。

答案と正解例のステップデータがキュー S_S と T_S に格納されているとき、誤っているダイナミックダイスを構成する命令の集合 EI 、及び誤っている可能性のある静的チャンクと誤っている静的チャンクの集合 EC の組 (EI, EC) を求めるアルゴリズムは以下のとおりである。

S1. EC, EI 、誤っている可能性のある静的チャンクの集合 UC 、及び正しい命令の集合 CI を空にする。

S2. $S_E = \text{DCEQ}(\text{DChunk}(S_S), S_S)$ 及び

$T_E = DCEQ(DChunk(T_S), T_S)$ を実行する.

S3. S_E が空または T_E が空ならば $S12$ へ進み, そうでないならば T_E の先頭要素を取り出し T_DC とし, S_E の先頭要素を取り出し S_DC とする.

S4. T_DC が空でないならば要素の一つを取り出し tdc とし, そうでないならば $S3$ へ戻る.

S5. S_DC の要素 sdc において, $sdc.sc = CmpSC(tdc.sc)$ が真となる sdc を求める.

S6. $DCOut(sdc)$ の要素 sr において, $sr.n$ の集合 PRN を求める.

S7. $DCOut(sdc)$ の要素 sr , $DCOut(tdc)$ の要素 tr , $CmpRN(tdc.sc)$ の要素 crn において, $(sr.n = crn$ かつ $tr.n = crn$ かつ $sr.v \neq tr.v)$ が真となる crn の集合 CRN を求める.

S8. CRN が空でないならば, CRN のある要素 crn に対して, $EI = DSlice((Def(sdc.e, crn), crn)) \setminus CI$ を実行し, $S11$ へ進む.

S9. $CmpRN(tdc.sc)$ が PRN の部分集合であるならば, $S9-1 \sim S9-4$ を実行する.

S9-1. $DCOut(sdc)$ の要素 sr , $DCOut(tdc)$ の要素 tr , $CmpRN(tdc.sc)$ の要素 cr において, $(sr.n = cr$ かつ $tr.n = cr$ かつ $sr.v = tr.v)$ が真となる cr の集合 CRE を求める.

S9-2. CRE の全ての要素 cre に対して, $CI = CI \cup DSlice((Def(sdc.e, cre), cre))$ を実行する.

S9-3. $CmpRN(tdc.sc)$ が PRN の真部分集合であるならば, $sdc.sc$ を UC へ加える.

S9-4. $S4$ へ戻る.

S10. $DCOut(sdc)$ の要素 sr において, $sr.n \notin CmpRN(tdc.sc)$ が真となる sr を見つけ, $EI = DSlice((Def(sdc.e, sr.n), sr.n)) \setminus CI$ を実行する.

S11. EC に $sdc.sc$ を加える.

S12. $EC = EC \cup UC$ を実行する.

S13. (EI, EC) の組を返す.

図 6 の答案に関する表示のための (EI, EC) を求めることを考える. 図 2 の正解例と命令語・計算機リソース誤りのプログラムにおいて, 答案の命令が先頭から順に $si1, si2, \dots, si10$ で, 答案と正解例の静的チャンクが $(ssc1, ssc2, ssc3, ssc4)$ と $(tsc1, tsc2, tsc3, tsc4)$ で, スタックポインタに「0x100」, メモリ番地「0x100」に「0x120」, 「0x101」に「0x4」, 「0x102」に「0x3」, 「0x120」に「0xF000」, $GR7$ に「0x110」をセットして実行して得られたステップデータが $S_S = (ss1, ss2, ss3, ss4, ss5, ss6, ss7, ss8)$ と $T_S = (ts1, ts2,$

$ts3, ts4, ts5, ts6, ts7, ts8)$ であるとする. 照合リソースは, $tsc1$ にて「 $GR7$ 」, $tsc2$ にて「 $GR1$ 」と「 $GR2$ 」, $tsc4$ にて「 $GR1$ 」と「 $GR7$ 」であるとする. このとき, 上記のアルゴリズムにより (EI, EC) は下記のように求められる.

1. $S1$ を実行する.

2. $S2$ にて,

$$S_E = (\{(ssc1, ss2, \{(GR7, 0xff)\})\}, \\ \{(ssc2, ss4, \{(GR1, 0x120), \\ (GR2, 0x4)\})\}, \\ \{(ssc3, ss5, \{(GR1, 0x124)\})\}, \\ \{(ssc4, ss8, \{(GR7, 0x110)\})\})$$

及び

$$T_E = (\{(tsc1, ts2, \{(GR7, 0xff)\})\}, \\ \{(tsc2, ts4, \{(GR1, 0x4), \\ (GR2, 0x3)\})\}, \\ \{(tsc3, ts5, \{(GR1, 0x7)\})\}, \\ \{(tsc4, ts8, \{(GR7, 0x110)\})\})$$

が求まる.

3. $S3$ にて, $T_DC = \{(tsc1, ts2, \{(GR7, 0xff)\})\}$ 及び $S_DC = \{(ssc1, ss2, \{(GR7, 0xff)\})\}$ が求まる.

4. $S4$ にて, $tdc = (tsc1, ts2, \{(GR7, 0xff)\})$ が求まる.

5. $S5$ にて, $sdc = (ssc1, ss2, \{(GR7, 0xff)\})$ が求まる.

6. $S6$ にて, $PRN = \{GR7\}$ が求まる.

7. $S7$ にて, $CRN = \{\}$ が求まる.

8. $S9-1$ にて, $CRE = \{GR7\}$ が求まる.

9. $S9-2$ にて, $CI = \{si2\}$ が求まる.

10. $S9-3$ にて, $UC = \{\}$ が求まる.

11. $S4$ を実行する.

12. $S3$ にて,

$$T_DC = \{(tsc2, ts4, \{(GR1, 0x4), (GR2, 0x3)\})\}$$

及び

$$S_DC = \{(ssc2, ss4, \{(GR1, 0x120), (GR2, 0x4)\})\}$$

が求まる.

13. $S4$ にて,

$$tdc = (tsc2, ts4, \{(GR1, 0x4), (GR2, 0x3)\})$$

が求まる.

14. S5 にて,

$sdc = (ssc2, ss4, \{(GR1, 0x120), (GR2, 0x4)\})$
が求まる.

15. S6 にて, $PRN = \{GR1, GR2\}$ が求まる.

16. S7 にて, $CRN = \{GR1, GR2\}$ が求まる.

17. S8 にて, $EI = \{si3\}$ が求まる. これは, $DSlice$ の結果が $\{si2, si3\}$ となり, ここから CI を取り除くことによる.

18. S11 にて, $EC = \{ssc2\}$ が求まる.

19. S12 にて, $EC = \{ssc2\}$ が求まる.

20. S13 にて, $(\{si3\}, \{ssc2\})$ が返される.

5. プロトタイプシステム

図 10 のように, 提案システム APAS を実装し, プログラミング演習支援システム CAPES [3] とアセンブリ構造誤り検出システム ASADS [2] と結合して, 提案機能を含むアセンブラプログラミング演習システムを構成した. 提案システム APAS の実装には Apache Tomcat [7], JavaServer Pages [8] を用いた. 我々は, この演習システムを名古屋工業大学工学部情報工学科 2 年生を対象とした授業「システムプログラム」において 2009 年度～2011 年度にわたり運用し, 延べ 242 人の受講者が利用した. 図中, Questions は問題, Simulator commands はシミュレータ動作コマンド系列, Symbol table はシンボル対応表である.

演習前に, 教師は CAPES に問題, 正解例, シミュレータ動作コマンド系列, シンボル対応表, 静的チャンク条件, 照合リソース, 拡張構造パターン抽出スクリプトを登録しておく. このうち, APAS のために新たに用意する情報は静的チャンク条件, 照合リソース, 及び拡張構造パターン抽出スクリプトである. 演習中

に, ウェブブラウザを通じて学習者が問題を CAPES に要求すると, CAPES は要求された問題をウェブブラウザ上に表示する. 学習者が CAPES に答案を投稿すると, ASADS が答案と正解例をステップ実行し, ステップデータを生成する. その後, シンボルテーブルに基づき, ステップデータ中のアセンブラシンボルを問題文の C プログラムの高級言語シンボルに置き換えたシンボル置換え後ステップデータ系列を生成する. そして, APAS は答案の誤りに応じて, 静的チャンク表示, 振舞い表示, 誤り潜在域表示を生成し, 学習者のウェブブラウザ上に表示する.

6. 評価実験

本実験は, 静的チャンク表示, 振舞い表示, 誤り潜在域表示が学習者による誤り原因の特定に役立つかを評価することを目的としている. 名古屋工業大学工学部情報工学科 2 年生を対象とした授業「システムプログラム」において, 本システムのアンケート評価を行った. この授業は 4 回のアセンブラプログラミング演習を含んでおり, 受講生は本システムを利用して 13 問の問題を解く.

第 4 回目の演習後にて評価アンケートを実施した. 質問は「誤り原因を探すのに役立ったか」を問うものである. 質問への回答は {1:役に立たなかった, 2:あまり役に立たなかった, 3:どちらともいえない, 4:やや役に立った, 5:役に立った} の 5 段階評価である. また, これに加えてアンケートは各々の質問への回答理由の記述欄をもつ. 評価結果を表 1 に示す.

評価アンケートの結果が, 評価値が 4 または 5 の被験者は, 静的チャンク表示で 30 人 (48%), 振舞い表示で 51 人 (70%), 誤り潜在域表示で 22 人 (65%) であること, 及び提出された答案から全ての学習者が最終的に正しく修正できていることから, 問題点に対して一定の効果があるといえる. このほか, 講義内容の多くが定着していないために誤りを特定できない学習

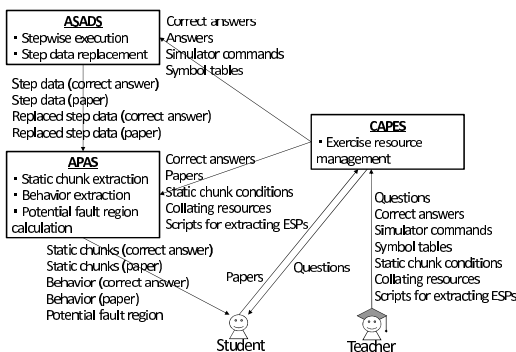


図 10 プロトタイプシステムの構成
Fig. 10 Structure of prototype system.

表 1 評価結果
Table 1 Questionnaire results.

評価値	シミュレータのステップ実行機能	静的チャンク表示	振舞い表示	誤り潜在域表示
1	4	10	4	4
2	4	11	6	6
3	4	12	12	3
4	31	21	30	11
5	29	9	21	11
合計	72	63	73	34

者は、誤り原因の特定にシミュレータのステップ実行を活用できる状態に至っていないため、シミュレータのステップ実行機能の評価にて1, 2, または3を回答すると考えられる。この被験者12人(17%)のうち、4または5が静的チャック表示で4人(33%), 振舞い表示で9人(75%), 誤り潜在域表示で3人(25%)となり、提案した表示は講義内容の未定着箇所の発見に一定の効果があるといえる。

振舞い表示へのコメントとして「自分のプログラムのどこが違うか、処理の順番が正しいかなどを知るのに役に立ちました。」という肯定的なものがあつた。これは、振舞い表示がその目的である「問題文の要求する振舞いを示す」に対して有効であつたことを示している。また、振舞い表示への否定的なコメントとして、「示している箇所を理解できれば参考になるが、間違つた箇所がどの場所か分からないことが多かった。」があつた。この回答者は振舞い表示への評価は3であつたが、誤り潜在域表示への評価は4であつた。これは、誤り潜在域表示がその目的である「誤り原因の潜在域の絞込み」に対して有効であつたことを示している。誤り潜在域表示への肯定的なコメントとして「命令単位で教えてくれるのは非常に役立った。」があつた。これは、ダイナミックスライスにより誤りの原因を絞り込んでいることが、学習者の誤り原因特定に役立ったことを示唆している。

7. 関連研究

PELAS [9], CHASE [10], FIND [11] は、論理エラー（本研究の命令語・計算機リソース誤りに相当するもの）の原因をユーザとの対話を通じてプログラム解析により効率的に絞り込む。この作業にはプログラムの正しい振舞いを命令単位で理解している必要があるため、これらのシステムは文法や命令の動作を理解しているプログラミングに慣れたユーザ向けであるといえる。本研究の対象者はアセンブラプログラミングの初学者であるため、これらのシステムは適さない。

研究[12], [13] は事例ベース推論処理を利用して、アセンブラプログラミング演習において答案の自動正誤判定を行うシステムである。研究[12] は事例ごとに教師がアドバイスを設定しておく方法、研究[13] は答案と近似した事例との差異からアドバイスを自動生成する方法である。これらは、事例が増えることでより適切なアドバイスを得られるようになるという長所がある。一方、システムの運用に必要となる事例を収集す

る必要があり、リアルタイムでの運用や、毎年異なる課題を出す演習での運用には適さない。

ALPUS [14] は、論理エラーの答案プログラムに対応した知識をパターンマッチングにより知識ベースから探し出し、その知識に基づいたアドバイスを生成する。論理エラーの原因は教師により事前に発見され知識化されている。このため、プログラムの修正箇所を細かく指摘することができる。しかし、論理エラーの原因は問題ごと、答案ごとに様々であるため、原因を演習に先立って想定し、知識化しておくことは教師への負担が大きいといえる。一方、本システムは学習者による命令語・計算機リソース誤りの原因特定の支援を振舞い表示と誤り潜在域表示で行う。このため、教師が事前に様々な誤り原因を想定し、知識化しておく必要はない。また、3.2で述べたように、「典型的な実装誤りをもつ命令系列」の静的チャック条件を定義することで、静的チャック表示にて命令語・計算機リソース誤りの原因となる命令系列を静的チャックとして表示できる。しかし、静的チャック表示は制御構造誤りの原因特定の支援を目的としたものであり、命令語・計算機リソース誤りの原因特定の支援に使用される必要はない。

研究[15] は、C プログラミング演習においてコンパイルのメッセージに対するアドバイスを生成するシステムである。システム[16], [17] はコンパイル時誤りに対するアドバイスを生成するデバッグ支援システムである。一方、本研究は命令語・計算機リソース誤りの原因特定を支援するものである。

8. む す び

本研究では、誤り原因を特定できないため、演習をあきらめてしまう学習者の存在を問題点としてきた。誤り原因を特定できない原因を、多くの講義内容の未定着、答案の分析ミス、及びCプログラムの分析ミスと考えた。

このため、講義内容の未定着箇所の発見を支援するアプローチ、答案の分析を支援するアプローチ、及び問題文の分析を支援するアプローチをとり、三つの機能を有する答案プログラム分析システムを開発した。静的チャック抽出機能の開発のために、CASL II プログラムから perl のパターンマッチングにより静的チャックを抽出する方法を提案した。振舞い抽出機能の開発のために、プログラミング演習システム[2]の構造パターン抽出法を拡張した拡張構造パターン抽出法を提案した。誤り原因絞込み機能の開発のために、ダイナ

ミックスライス, 静的チャンク, 及び動的チャンクを用いて, 誤っているダイナミックダイス, 及び誤っている可能性のある静的チャンクと誤っている静的チャンクを求める方法を提案した. 評価実験では, 学習者による誤り特定及び講義内容の未定着箇所の発見に対する提案表示の一定の効果が示された.

今後の課題は, 自動正誤判定と提案表示のための教師の負担低減である. 振舞いと制御構造の特徴を指定するという点で, 構造パターン抽出スクリプトと, 静的チャンク条件及び照合リソースは近いと思われる. このため, これらを相互変換する機能が統合的な記述方法の開発を行いたい.

文 献

- [1] IPA 独立行政法人情報処理推進機構 IT 人材育成本部情報処理技術者センター, “「試験で使用する情報技術に関する用語・プログラム言語など」 Ver2.1 (2011 年 10 月版),” http://www.jitec.ipa.go.jp/1.13download/shiken_yougo_ver2.1.pdf, pp.3-8, 2011.
- [2] 宮地恵佑, 高橋直久, “構造誤り検出機能を有するアセンブラプログラミング演習支援システムの実現と評価,” 信学論 (D), vol.J91-D, no.2, pp.280-292, Feb. 2008.
- [3] 中島秀樹, 細川宜秀, 高橋直久, “プログラミング学習のための QA サイクル: 受講者の習得度に応じた問題自動提示メカニズム,” 信学論 (D-I), vol.J88-D-I, no.2, pp.439-450, Feb. 2005.
- [4] 下村隆夫, プログラムスライシング技術と応用, 共立出版, 東京, 1995.
- [5] H. Agrawal and J. Horgan, “Dynamic program slicing,” SIGPLAN Notices, vol.25, no.6, pp.246-256, 1990.
- [6] The Perl Programming Language, <http://www.perl.org/>
- [7] Apache Tomcat, <http://tomcat.apache.org/>
- [8] JavaServer Pages Technology, <http://www.oracle.com/technetwork/java/javaee/jsp/index.html>
- [9] B. Korel, “PELAS — Program error-locating assistant system,” IEEE Trans. Softw. Eng., vol.14, no.9, pp.1253-1260, 1988.
- [10] T. Shimomura, “Bug localization based on error-cause-chasing methods,” Trans. IPSJ, vol.34, no.3, pp.53-64, 1993.
- [11] 下村隆夫, “FIND: デバッグ自動化支援システム,” 信学論 (D-I), vol.J79-D-I, no.7, pp.446-456, July 1996.
- [12] 渡辺博芳, 荒井正之, 武井恵雄, “事例に基づく初等アセンブラプログラミング評価支援システム,” 情処学論, vol.42, no.1, pp.99-109, 2001.
- [13] 渡辺博芳, 高井久美子, 荒井正之, 武井恵雄, “初等アセンブラプログラミングにおけるニアミスプログラムに対するアドバイス提示法,” 情処学研報, vol.2003, no.49, pp.31-38, 2003.
- [14] H. Ueno, “Concepts and methodologies for knowledge-based program understanding — The

ALPUS's approach,” IEICE Trans. Inf. & Syst., vol.E78-D, no.9, pp.1108-1117, Sept. 1995.

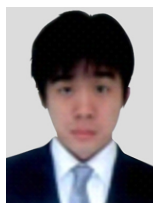
- [15] 高本明美, 藤井美知子, 泉 直利, 田中 稔, “誤り原因を指摘するプログラム学習支援システムとその学習効果,” 教育システム情報学会誌, vol.17, no.4, pp.533-540, 2001.
- [16] 海尻賢二, 吉田美樹雄, “統計的手法によるデバッグ支援システムの構築,” CAI 学会誌, vol.11, no.2, pp.51-62, 1994.
- [17] 大西 淳, 島崎真昭, “CONSULT/C: コンパイル時のエラーに関するプログラム相談手法/システム,” 情処学研報, vol.1989, no.11, pp.185-192, 1989.
(平成 24 年 9 月 12 日受付, 25 年 1 月 7 日再受付)



立岩佑一郎 (正員)

2008 名古屋大学大学院情報科学研究科博士課程了 (情報科学博士). 同年, 名古屋工業大学大学院工学研究科助教となり, 現在に至る. 仮想マシン技術, ネットワーク, 情報教育等に興味をもつ. ACM, IEEE, 教育システム情報学会, 情報処理学会各

会員.



吉田 裕一

2008 名工大・工・情報工学卒. 2010 同大大学院工学研究科情報工学専攻修士課程了. 同年, パナソニックアドバンステクノロジー (株) に入社, 現在に至る. 現在, デジタルテレビ, BD レコーダの組込みシステム開発に従事.



山本 大介

2003 名大・工・電気電子・情報卒. 2008 同大大学院情報科学研究科博士課程了. 2006~2008 日本学術振興会特別研究員. 2008~2012 名古屋工業大学工学研究科助教. 2012 同准教授, 現在に至る. 情報処理学会, 日本データベース学会, 人工知能学会各会員. 博士 (情報科学).



高橋 直久 (正員)

1974 電通大・応用電子卒. 1976 同大大学院応用電子工学専攻修士課程了. 同年日本電信電話公社 (現 NTT) 武蔵野電気通信研究所入所. 1987 工博 (東京工業大学). 2001 名古屋工業大学電気情報工学科教授. 2004 名古屋工業大学大学院工学研究科ながれ領域 (情報工学科/情報工学専攻担当) 教授. 現在, 時空間情報処理, e ラーニングシステム, ネットワーク診断システムなどの研究に従事.