変数の出現方法に着目したプログラムの段落分けの手法の提案

2020SE040 森龍成
指導教員：吉田敦

1　はじめに

プログラムの理解において，一度に全体を理解することが難しいため，意味的なまとまりごとに分けて理解する必要がある．理解支援ツールは，プログラムを意味のある単位に分割して支援をしようとするが，意味的なまとまりの特定が難しく，制御構造を単位とする近似的な分割が一般的に使用されている．しかし，制御構造に基づくため，ツールが提示する構造と読み手が考える構造の整合性をとることが難しい．手続き型プログラムでは，ファイルやバッファなどのリソースを確保し，それに対して一連の操作を行うことが一般的であり，リソースの確保とそれに対する操作が理解する上での 1 つのまとまりと捉えられる．この理解する上でのまとまりのことを本研究では段落と呼ぶ．本研究では，理解支援ツールがより意味のまとまり単位での分割ができるよう，これらの考え方を前提に段落分けの手法を提案する．

2　関連研究

倉田の研究 [1] は，プログラマが挿入した空白に基づく文のまとまりを段落と呼び．そのまとまり具合の評価方法を提案している．段落の切り分けがプログラムの処理に基づかない点で本研究とは異なる．

メソッド抽出 [2] はプログラムを分割するという点で類似しているが，この研究は切り出したい部分の指定から始まるが，本研究はプログラム全体の分割という違いがある．

3　プログラムの段落分けの手法の提案

3.1　段落の定義

本研究では，リソースの確保とそれに対する一連の操作を意味のまとまり，すなわち段落と捉えるが，特定の鍵となるリソースの確保の発見が難しい．そこで，リソースを広く捉え，変数の初期化とその変数の値の更新・参照の文の集合を基本単位とし，段落は複数の基本単位が結合したものとする．各文が他の変数の参照を含む場合，基本単位間ではその文を共有し，複数の基本単位の結合が生じる．

この定義に基づくと，関数は 1 つの段落とみなせる．関数が 1 つの機能を実現すると考えると，一般的には特定の変数を中心に操作が行われ，その変数は関数全体で使用される．この変数が基本単位群を広く結びつけるので，その変数を無視することで，関数を複数の部分に分割できる．それぞれの部分では，その中にのみ出現する変数による基本単位群が結びつくので，意味のまとまりができ，これも段落にあたる．この定義では，段落内の文が連続して出現するとは限らず，また，段落自体が内部に他の段落を含むことがある．

3.2　段落分けの手法の概要

本研究では，段落を入力とすると，その内部を段落分けする手法を提案する．なお，最初は関数を段落とみなして入力とし，再帰的に適用するものとする．

分割をする際には，基本単位を広く結びつける変数を特定する．段落がそのような変数を対象とした操作で構成されるとすれば，段落の前方と後方に出現すると考えられ，広く分散して出現する変数（以下，「分散的な変数」）がそのような変数となる．段落の基本単位を構成する文をノードとしたプログラム依存関係グラフ（以下，PDG）から分散的な変数の一連の操作に関わるデータ依存関係を除去することで，その関係で結びついていた内部の段落を分離する．分散的な変数が複数存在する場合，複数の変数で分割を行う．変数の初期化とその初期値に関わる操作は初期値からのデータ依存関係にある文のリストでもあり，PDG における 1 つのパスを構成する．それらが組み合わさったものが段落となるので，そのパスの削除により分割を実現する．PDG を分割して得られる部分グラフが段落となる．また，制御構造内はこの手法を再帰的に適用することで分割できる．PDG の定義については下村 [3] に基づく．

3.3　分散的な変数の決定方法

分散的な変数の決定には，指定されたプログラムにおける文番号の総数に占める変数の最大文番号と最小文番号の差の割合をすべての変数において計算し，基準値を上回った変数を分散的な変数と決定する．文番号は，プログラム内の文に出現順に番号を付け，制御文の中の文も含めて番号を付ける．便宜上，文には宣言を含む．

式を以下に示す．

$$\alpha = \frac{(v_{max} - v_1)}{s_{max}} \times 100$$

$$\beta = \frac{(v_{max} - v_2)}{s_{max}} \times 100$$

- v_i : 指定された変数が i 番目に出現する文番号
- v_{max} : 指定された変数の最大の文番号
- s_{max} : 最大の文番号

α は指定された変数の出現をすべて考慮した式であり，初期化が実際に使用される位置より離れることを計算に考慮しない式が β である．α は，初期化が離れている場合，値が大きくなり，実際には分散していないのに分散しているとみなされる場合があるため，β を用意する．

2 つの式の選択は次のように行う．2 つの式の値の差（以下，外れ値の影響値）を求め，影響値以上の際は β，それ以外は α を用いる．基準値と影響値は予備実験を行い，基準

値は α は 65，β は 60 とした．また，影響値を 25 とした．

3.4 適用例

図 1 のプログラム [4] を用いて説明する．分散的な変数を特定するために出現するすべての変数に対して式を用いて計算を行い，表 1 に計算した値，用いた式，基準値，分散的な変数の可否を示す．下の 2 つの変数は出現が 1 回であるので，値を算出できず，判定外とした．計算した結果，変数 p のみ分散的な変数と判別できる．変数 p が関わるデータ依存関係を除去前の PDG を図 2 を示し，除去するエッジを赤で示す．また，除去後の PDG を図 3 に示す．図 2 において，残った部分グラフを枠線で囲んで明示し，対応する箇所を図 1 にも明示する．それぞれ prefix_lead_space，prefix，prefix_length を求める処理として独立しており，段落として成り立つ．

```
static void set_prefix (char *p)
{
1   char *s;

2   prefix_lead_space = 0;
3   while (*p == ' ')
    {
4     prefix_lead_space++;
5     p++;
    }
6   prefix = p;
7   prefix_full_length = strlen (p);
8   s = p + prefix_full_length;
9   while (s > p && s[-1] == ' ‘)
10    s--;
11  *s = '¥0’;
12  prefix_length = s - p;
}
```

図 1　サンプルプログラム

表 1　分散的な変数の決定結果

変数名	値	式	基準値	決定
p	75	α	65	○
s	33.3	β	60	×
prefix_full_length	8.3	α	65	×
prefix_lead_space	16.6	α	65	×
prefix	出現 1 回			-
prefix_length	出現 1 回			-

4　評価

3 つのサンプルプログラム [4] に対して適用し，手動での段落分けの結果と比較したところ，一致した．

5　考察

分散的な変数の決定方法の改善として，値の差を用いるのではなく外れ値の分析をしてそれを除いた計算が考えられる．プログラムを静的解析して，特定の変数において離れて存在している変数が存在すれば，その変数を外れ値として計算を行うことでより分散的な変数の決定の精度を高められる．

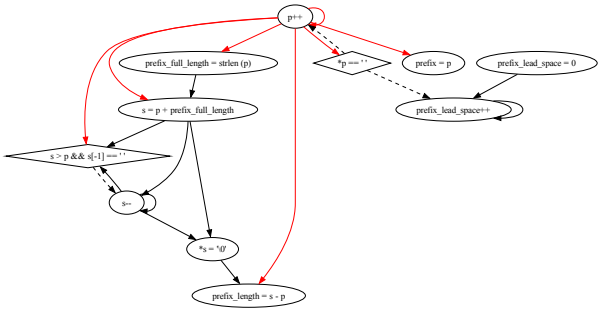

図 2　サンプルプログラムの PDG

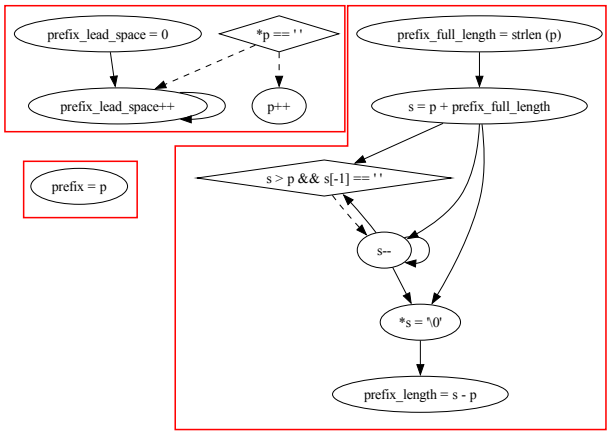

図 3　分散的な変数が関わるエッジを除去した PDG

6　おわりに

本論文では変数に着目したプログラムの段落分けの手法を提案した．外れ値の分析や PDG におけるエッジの削除，プログラムの分割の自動化が今後の課題となる．

参考文献

[1] 倉田優輝：変数の依存関係を用いた段落の評価指標の提案—プログラム可読性の評価への応用をめざして—，三重大学大学院 工学研究科 博士前期課程 情報工学専攻 コンピュータソフトウェア研究室 36p(2020).

[2] 丸山勝久：基本ブロックスライシングを用いたメソッド抽出リファクタリング，立命館大学理工学部情報学科，pp.1625–1637 (2002).

[3] 下村隆夫：プログラムスライシング技術と応用，pp.1–23，共立出版株式会社 (1995).

[4] “Coreutils - GNU core utilities.”，<https://www.gnu.org/software/coreutils/coreutils.html>(参照 2023-08-05).