山口大学大学院東アジア研究科
博士論文

C言語プログラムの類似度評価に関する研究

2016年3月

包 胡日查

学位論文内容要旨

学位論文題目：C 言語プログラムの類似度評価に関する研究

指導教員　　：葛　崎偉　教授

申請者名　　：包　胡日査

平成 24 年度入学
山口大学大学院　東アジア研究科　博士後期課程

プログラムソースコードの類似性を判定することはプログラム本体の重複コードや無駄なコードを減らせ，さらに知的財産の分野では，著者の知識財産を守ることも意味している．C 言語プログラムは手続き言語であり，実行順で関数の呼び出し関係ができ，関数内部でも順に実行されるので全プログラムを 1 つの木グラフとして表現し，木グラフのノード節点は C ソースコードで表現できる．

木構造は XML ファイルや遺伝子の進化過程など，様々なオブジェクトの表現に用いられている．このような木で表現されたオブジェクトを検索したり分類したりするには，木の類似性判定が必要である．プログラミング教育の現場において，教師は学生から提出される大量のプログラムを目視により妥当性を判断し，それぞれのプログラムに適した評価をする必要がある．

二つの C 言語プログラムがどれぐらい類似しているか，文字列を厳密に比較して算出するのではなく，プログラムを構文木に変換し，変換された木グラフの類似度を用いてプログラムの類似度とする．構文木に変換して比較することは，計算コストを下げるだけではなく，木の特性を生かした比較ができる．これにより，文字列比較ではできなかったプログラム構造の比較が可能になり，また関数の呼び出し関係まで比較することができる．我々は C 言語プログラム同士の類似度を測ることを目的し，C 言語プログラムの類似度の求める方法としては，New Tree Overlapping 手法と Depth Matching 手法を提案する．

本論文は以下のように第 1 章から第 6 章までで構成される．

1 章では研究の背景を紹介し，関連の研究状況を概説しており，また研究の目的について述べる．

2 章では，研究に関連する基礎的な定義や概念を記述する．まず，木グラフ (Tree Graph) の定義や構文木（Syntax Tree）や二部グラフのマッチング（Bipartite Matching）について説明した上で，それらの基本的な性質を示す．次に文字列編集距離と木間の編集距離について説明し，最後に既存のプログラム類似性判定手法の SMMT について紹介する．

3 章では，C 言語プログラムを構文木化する手法を提案する．まずは C 言語ソースプログラムに対する前処理を行う．次に，C 言語プログラムの構成パターンを分類し，それぞれのパターンを表現する構文木部品を作成する．更に，これらの部品を用いて C 言語プログラム全体を一つの木グラフに変換する．本論文では，提案した構文木化の方法を用いて作成した CCX ソフトの説明や，具体例を用いた変換結果の紹介も行う．

4 章では，構文木グラフの類似度を求める手法を提案する．既に提案された Tree Overlapping 手法を紹介した上で，すべての共通部分木を用いて構文木の類似度を測る New Tree Overlapping（NTO）方法を提案する．また，木グラフの深さを基準にノード間の最大マッチングを計算することによって構文木の類似度を測る Depth Matching（DM）手法を提案する．

5 章では，サンプルプログラムを用いて提案した二つの手法と既存の SMMT 手法との比較実験を行い，その結果について議論する．サンプルプログラムは，学生のレポートやテキストに掲載のプログラムや研究用のプログラムなどである．実験結果から，提案の DM 手法が最も良く，その有効性が確認できた．

6 章では，本論文で得られた結果をまとめており，今後の展望や課題について述べる．

A Study on Similarity Evaluation of C Language Programs

by

Huricha Bao

Thesis submitted to the Graduate School of East Asian Studies
of Yamaguchi University,
in partial fullfillment of the requirements for the degree of
Doctor of Philosophy

Advisory Committee:

Professor Qi-Wei GE, Chairman/Advisor
Professor Takashi NARITOMI
Professor Takamasa FUKUDA
Associate Professor Shingo YAMAGUCHI

謝辞

本研究の遂行並びに博士論文の作成にあたり，多くのご支援とご指導を賜りました．研究活動全般に御指導と御高配を賜りました山口大学大学院東アジア研究科葛崎偉教授に甚大なる謝意を表します．葛崎偉教授から，丁寧かつ熱心なご指導を賜り，挫けそうなときに励まして頂きました．先生の支援，促進，鞭撻および助言は，個人・学術的な成長において莫大な価値があり，毎日の思いやりのある指導に特に感謝します．私が非常に興味を持っているプログラムソフトウエアとグラフの新しい学術分野をより深く学ぶことができました．過去の６年，葛先生が熱心に私の研究と論文にしっかり指導を頂き，多数の研究交流で自分の研究が改善され，洗練されました．私は非常に幸運で，日本での研究の全体にわたって葛先生の有用な助言を受け研究促進することができました．

私にいつも丁寧にアドバイスをして頂いた副指導教員として山口大学大学院東アジア研究科成富敬教授から多分野に渡り有益な助言を頂きました．また，副指導教員である山口大学大学院東アジア研究科福田隆眞教授から日頃より多くのご助言や励ましの言葉を頂きました．ここで，深く感謝いたします．

学位論文審査において，貴重なご指導とご助言を頂いた山口大学大学院理工学研究科山口真悟准教授に心より感謝いたします．本研究を進めるにあたり，多くのご支援とご指導を頂いた山口大学教育学部中田充教授に深く感謝いたします．

特別に葛先生の奥様呉靭先生（山口短期大学）に感謝します．呉先生より多くの有益な助言やノウハウやアドバイスを頂き，特にシミュレーション実験の指導と，研究促進のための丁寧で熱心な指導を頂きました．

さらに，私は研究室での皆さまとの交流を通して日本で勉強と生活を有意義で価値のある経験にすることができました．中田研究室の長岡さんと木村さんと葛研究室の西田さん熱心的に本論文の日本語の修正をして頂き，深く感謝します．

最後に，私の両親と兄に心の支えになり絶えず支援してくれたことに感謝します．

彼らの愛および叱咤激励なしでは，私はこの学位論文を遂行することができなかったと思います.

表記法/Notation

$x \in X$:	x は X 集合の要素である.
$x \notin X$	x は X 集合の要素ではない.
$X \cup Y$:	X と Y の和集合である.
$X \cap Y$:	X と Y の交差集合である.
$\sum x_i$:	x_i すべての要素の合計である.
ϕ:	空集合である.
N:	自然数の集合である.
$\|X\|$:	X 集合の要素数である.
$\max X$:	X 集合の最大要素.
$\min X$:	X 集合の最小要素.
$X \leftarrow \{x_1, x_2, \cdots, x_n\}$:	X 集合に $x_1, x_2, \cdots, x_n$ の要素を追加である.
$L \leftarrow t_1 t_2 \cdots t_n$:	L の左からの順に子供 $t_1, t_2, \cdots, t_n$ を追加である.
$x \leftarrow x'$:	変数 x に x' の値を割り当てる.

目次

論文概要 **3**

謝辞 **5**

表記法 **8**

目次 **9**

図目次 **11**

表目次 **18**

1 はじめに **1**

1.1 研究の背景と動機 1

1.2 類似プログラム判別技術の研究状況 3

1.3 論文の目的と構成 7

2 本研究の基本概念 **9**

2.1 グラフの定義 9

2.1.1 木 9

2.1.2 構文木 11

2.1.3 木間写像と最大共通部分木 11
2.2 二部グラフの定義 13
2.3 編集距離 14
2.3.1 文字列編集距離 14
2.3.2 木編集距離の定義 15
2.4 SMMT 手法 17

3 C 言語プログラムの構文木化手法の提案 19
3.1 プログラムの構文木化の目的 19
3.2 プログラムの構文木の作成 21
3.2.1 プログラムの前処理 21
3.2.2 構文木の構成部品 22
3.2.3 構文木の作成 23
3.3 XML 形式の構文木の作成およびその構造 24

4 木グラフの類似度を求める手法の提案 31
4.1 Tree Overlapping 手法 31
4.2 New Tree Overlapping（NTO）手法の提案 33
4.3 Depth Matching（DM）手法の提案 44

5 シミュレーション結果と考察 55
5.1 各手法によるシミュレーションと評価 55
5.2 考察 60

6 おわりに 65
6.1 研究の成果 65
6.2 本研究の展望 67

6.3　今後の課題　. 69

参考文献 **71**

図目次

1.1 書き換えたサンプルプログラム例 2
1.2 木構造が持つ XML で表現されたオブジェクト例 5

2.1 木 (Tree) 9
2.2 木の構造 9
2.3 木の中に含まれる部分木 10
2.4 プログラムの構文木一例 11
2.5 木間写像と最大共通部分木 12
2.6 二部グラフ 13
2.7 二部グラフのマッチング例 14
2.8 weight と write の編集距離 15
2.9 ノードの挿入 16
2.10 ノードの削除 16
2.11 ノードの置換 16
2.12 SMMT 対応例 17

3.1 C プログラムの関数関連図 20
3.2 サンプルプログラムの前処理 22
3.3 構文木部品一覧 23

3.4 作成された構文木 25
3.5 標準化 C プログラムを XML 木構造に変換される CCX ソフト 25
3.6 可視化ツール Graphviz 26
3.7 中間プログラム表現可視化ツール CoVis 27
3.8 サンプル XML 構文木の可視化 28
3.9 XML 構文木のサンプル（左半分が構文木，右半分は属性やテキストの値） 29

4.1 Tree Overlapping 手法例 33
4.2 サンプルプログラム P 35
4.3 プログラム P の関数関係 35
4.4 サンプルプログラム P から変換された構文木グラフ T 36
4.5 構文木グラフ T_1 と T_2 37
4.6 ケース (i) NTO により得られた最大共通部分木 Rectangle1 39
4.7 ケース (i) NTO による更新された部分木とその最大共通部分木 Rectangle2 40
4.8 ケース (i) NTO に更新後のすべての共通部分木 Rec3,Rec4,Rec5 . . . 41
4.9 ケース (i) NTO による最終更新された部分木 41
4.10 ケース (ii) NTO により得られた最大共通部分木 Rectangle1 42
4.11 ケース (ii) 更新された部分木 42
4.12 ケース (ii) NTO 手法により更新された部分木とその最大共通部分木 Rectangle2 43
4.13 ケース (ii) NTO による最終更新された部分木 43
4.14 ケース (iii) NTO による最大共通部分木 44
4.15 ケース (iii) 更新された木 45

4.16 Depth Matching 手法例 46
4.17 ケース (i) DM 手法によるレベルごとの写像 47
4.18 ケース (i) 二部グラフ $G^1G^3G^6$ の最大ノンクロスマッチング $M_1M_3M_6$ 48
4.19 ケース (ii) DM 手法によるレベルごとの写像 49
4.20 ケース (ii) 二部グラフ $G^1G^3G^6$ の最大ノンクロスマッチング $M_1M_3M_6$ 50
4.21 ケース (iii) DM 手法によるレベルごとの写像 51

5.1 ケース (i) 関数名だけ変更した場合の類似度結果 59
5.2 ケース (ii) グローバル変数が変更した場合の類似度結果 59
5.3 ケース (iii) 自作関数名とグローバル変数が変更した場合の類似度結果 60
5.4 ケース (iv) ローカル変数を変更した場合 60
5.5 ケース (v) ローカルとグローバル変数両方変更した場合の類似度結果 61

6.1 ACCS 会員ソフトウエア企業により報告 [89] 67
6.2 世界ソフトウエア違法コピー調査 2011 による結果 (日本)[90] 68
6.3 世界ソフトウエア違法コピー調査 2011 による結果 (地域別)[90] 69
6.4 abstract.java（1） 81
6.5 abstract.java（2） 82
6.6 FunSplit.java（3） 83
6.7 FunSplit.java（4） 84
6.8 FunSplit.java（5） 85
6.9 FunSplit.java（6） 85
6.10 FunSplit.java（7） 85
6.11 CProgramsXML.cs（1） 86
6.12 CProgramsXML.cs（2） 87

6.13 CProgramsXML.cs（3） 88
6.14 CProgramsXML.cs（4） 89
6.15 CProgramsXML.cs（5） 90
6.16 CProgramsXML.cs（6） 91
6.17 CProgramsXML.cs（7） 92
6.18 CProgramsXML.cs（8） 93
6.19 CProgramsXML.cs（9） 94
6.20 CProgramsXML.cs（10） 95
6.21 CProgramsXML.cs（11） 96
6.22 CProgramsXML.cs（12） 97
6.23 CProgramsXML.cs（13） 98
6.24 CProgramsXML.cs（14） 99
6.25 CProgramsXML.cs（15） 100
6.26 CProgramsXML.cs（16） 101
6.27 CProgramsXML.cs（17） 102
6.28 CProgramsXML.cs（18） 103
6.29 CProgramsXML.cs（19） 104
6.30 CProgramsXML.cs（20） 105
6.31 CProgramsXML.cs（21） 106
6.32 CProgramsXML.cs（22） 107
6.33 CProgramsXML.cs（23） 108
6.34 CProgramsXML.cs（24） 109
6.35 CProgramsXML.cs（25） 110
6.36 CProgramsXML.cs（26） 111

6.37 CProgramsXML.cs (27) . . . 112
6.38 CProgramsXML.cs (28) . . . 113
6.39 CProgramsXML.cs (29) . . . 114
6.40 CProgramsXML.cs (30) . . . 115
6.41 CProgramsXML.cs (31) . . . 116
6.42 NTO-Method-XML.cs (1) . . . 117
6.43 NTO-Method-XML.cs (2) . . . 118
6.44 NTO-Method-XML.cs (3) . . . 119
6.45 NTO-Method-XML.cs (4) . . . 120
6.46 NTO-Method-XML.cs (5) . . . 121
6.47 NTO-Method-XML.cs (6) . . . 122
6.48 NTO-Method-XML.cs (7) . . . 123
6.49 NTO-Method-XML.cs (8) . . . 124
6.50 NTO-Method-XML.cs (9) . . . 125
6.51 NTO-Method-XML.cs (10) . . . 126
6.52 NTO-Method-XML.cs (11) . . . 127
6.53 NTO-Method-XML.cs (12) . . . 128
6.54 NTO-Method-XML.cs (13) . . . 129
6.55 NTO-Method-XML.cs (14) . . . 130
6.56 NTO-Method-XML.cs (15) . . . 131
6.57 DM-Method-XML.cs (1) . . . 132
6.58 DM-Method-XML.cs (2) . . . 133
6.59 DM-Method-XML.cs (3) . . . 134
6.60 DM-Method-XML.cs (4) . . . 135
6.61 DM-Method-XML.cs (5) . . . 136

表目次

3.1 前処理一覧 21

4.1 構文木 T の各ノードラベルとプログラムの対応ソースコード 1-1 . . . 37
4.2 構文木 T の各ノードラベルとプログラムの対応ソースコード 1-2 . . . 38
4.3 T_1 と T_2 の各ノードのラベル 46

5.1 サンプルプログラム 1-1 55
5.2 サンプルプログラム 1-2 56
5.3 NTO と DM と SMMT 手法による計算された類似度 1-1 57
5.4 NTO と DM と SMMT 手法による計算された類似度 1-2 58
5.5 NTO 手法による C 言語コース i のレポート間類似度結果 61
5.6 DM 手法による C 言語コース i のレポート間類似度結果 62
5.7 SMMT 手法による C 言語コース i のレポート間類似度結果 62
5.8 NTO 手法による C 言語コース ii のレポート間類似度結果 63
5.9 DM 手法による C 言語コース ii のレポート間類似度結果 63
5.10 SMMT 手法による C 言語コース ii のレポート間類似度結果 63

第 1 章

はじめに

1.1 研究の背景と動機

近年，インターネットやソフトウェア技術の発展と共に，人々の生活に大きな変化が起こり，情報は生活の隅々にまで浸透している．情報教育は現代社会を生き抜く力を養う基本的素養の一つになっており，平成 15 年文部省教育課程審議会により高等学校の普通科には情報科目も新設した [1]. 一方，C 言語は現在最も普及しているプログラミング言語 [2] であり，情報系の大学生にとって重要な科目になっている．Windows アプリケーションや UNIX などの OS の記述にも利用され，特に組み込みシステムにおいて最も多く使われている．

C 言語のプログラミング教育の現場において，プログラムの実行結果のみならず，教師は学生から提出される大量のプログラムを目視により妥当性を判断し，それぞれのプログラムに適した評価を行う必要がある [3]．目視によるプログラムの評価は，時間がかかるだけではなく，場合によっては適正な評価が難しいことも考えられる. また，学習者が他人のプログラムをコピーした上で，関数名や変数名だけを置き換えて（図 1.1）提出するような場合，プログラムの目視からこれに気づくことは難しい．さらに，他人のソースプログラムを簡単に書き変えて自分の提出物としているケースがある [4, 5, 6]．したがって，プログラムの類似性を自動的に測ることが望まれている．

一般的にプログラムの作成は，要求定義，システム設計（基本設計），プログラム設計（詳細設計）の過程を経て行われ，併せてプログラム取扱説明書（マニュアル）等の関係資料が作成される．そして，これらの過程において作成される要求仕様書，システム設計仕様書，フローチャート等プログラム設計仕様書や，プログラ

ム取扱説明書等のいわゆるドキュメント類はプログラムと共にソフトウェアと呼ばれる [7]．プログラムの作成で，もっとも大切な過程はプログラム設計で，プログラマの独自のもので，これを守るにはソフトウェア著作権 [8, 9] である．

昭和 60 年の法律改正でプログラムも著作物として保護されることとなった．したがって，他人のプログラムを無断で複製することは著作権侵害となる [10]．プログラムは作成者の学術的思想の創作であることは，従来の著作物と相違ないが，プログラム言語による表現そのものがデジタル化されコンピュータを機能させることから，産業財としての特性を有するようになってきている [11]．プログラミング教育の現場において，レポートや宿題などが学生個人の著作物として見なされ，個人のアイデアやオリジナルの発想があり，無断で複製や複製した上で書き換える行為も著者権侵害になる．したかって，プログラム間の類似性を自動的に評価する技術が望まれる．この技術は大学をはじめとした教育機関やソフトウェア企業などで求められている．

```
1
2  #include <stdio.h>
3  main()
4  {
5      int a,b,sum;
6      a=10;
7      b=24;
8      sum=add(a,b);
9      //関数addを呼び出す.
10     printf("sum= %d\n",sum);
11 }
12
13 int add(int x,int y)
14 {
15     int  z;
16     z=x+y;
17     return(z);
18 }
```

```
1
2  #include <stdio.h>
3  main()
4  {
5      int i,j,sum;
6      i=10;
7      j=24;
8      sum=count(i,j);
9      //関数countを呼び出す.
10     printf("sum= %d\n",sum);
11 }
12
13 int count(int x,int y)
14 {
15     int  z;
16     z=x+y;
17     return(z);
18 }
```

図 1.1: 書き換えたサンプルプログラム例

1.2 類似プログラム判別技術の研究状況

プログラムソースコードの類似性判定研究が古くから行われてきた．例えば，1976年，Purdue 大学の Ottenstein ら [12] が最初に応用属性計数法 (attribute counting) を用いて類似度計算を提案した（An Algorithmic Approach to the Detection and Prevention of Plagiarism）．Ottenstein らが初めて Halstead プログラム類似計測方法を用いてプログラムの類似性を判別し Fortran プログラムの類似度測定システムを開発した．M.Halstead はプログラムの長さから 4 つの基本パラメータを測ることを提案した．ただし，構造化プログラム言語の作成は単純に属性計数法では色々なプログラム情報が無くなり，類似性の判別が難しくなることは明らかである．

1992 年，Michael Wise らは Plague の基に YAP ツールを開発した．先に YAP1 を，次に YAP2 と最終バージョン YAP3 を開発した [13, 14]．YAP1 と YAP2 がプログラムの重複検測ため主に使われ，YAP3 はプログラムだけではなくテキストファイルにも使われる．ただし，YAP は diff コマンドと同様に，プログラムの構造の情報はほとんど反映されない．1994 年，Alex Aiken らは Berkeley で MOSS（Measure of Software Similarity）[15] を開発した．このソフトは，先に前処理を行い，次に関数名や変数や型や（identifier）や文字列などの類似度を比較する．具体的な方法とツールは公開されていない．MOSS が LCS の最長類似度エリアを見つけ出せる．

2007 年，大連理工大学の張らは LCS（Longest Common Subsequence）をもちいて C 言語プログラムの類似度を検測する手法を提案した [16]．この手法ではソースコードを文字列として文字列の最長共通部分列に基づいて類似度を求める．芝浦工業大学の福田らは教育支援システム [17, 18, 19] を開発した．これらのシステムでは各プログラムの行数と各行の文字数ををカウントし，行数順にソートし，それらの一致部分を比較する．[16] の手法でも，プログラムの一行ごとに類似度を求め，それらを基にプログラム全体の類似度を求めている．ここで，ラインごとの類似度の精度により少しだけ書き換えたパラメータの場合全部一致と取られてしまう恐れがある．

小田 悠介らは，プログラム間の類似性の定量化手法 [20] の段階的感度比較法を提案した．二つのプログラムを比較する方法には，例えばソースコードの先頭から末

尾まで文字が完全に一致するか調べる方法，構文解析の結果得られる構文構造が一致するか調べる方法，同じ入力に対する出力が一致するかどうかだけ調べる方法など，いくつかの方法が考えられる．これら複数の比較法を，厳密性が高い（比較の感度が高い）と考えられるものから順に適用していくと，どこかで一致することになる．プログラム同士が初めて一致した比較レベルをプログラム間の距離とする．次に示す 5 種類の比較法を比較レベルとして定義した．

1. 比較レベル 0: 一文字ごとの完全比較
2. 比較レベル 1: トークン列の比較
3. 比較レベル 2: 変数名，計算式を標準化して比較
4. 比較レベル 3: 制御構造を標準化して比較
5. 比較レベル 4: 入出力だけの比較

以上の研究では，プログラムの文字列が完全に一致している部分を比較によって比較したり，プログラムの制御文とパラメータを取り出して比較を行う．プログラムにとって処理の流れ或いは構造（関数・メソッド）も重要な要素であり，構造の比較はしなければならない．ここまでに挙げた研究を踏まえた上，我々は，C 言語プログラムを構文木に変換し，構文木を比較することでプログラムの類似度を求めることを考える．C 言語ではどんなに複雑で大きなプログラムであってもそれを小さな関数に分け，組み立てることで効率よくプログラムを構成でき，関数の呼び出し関係で簡潔な構文木が作成できる．構文木を作成する事で，木グラフの特性を生かしてプログラムの構造の比較が可能になる．そして，それぞれの構文木のノードに対応する文の記述をノードラベルとして，二つのプログラムから変換された構文木のラベルを比較することで，プログラムのソースコードの比較が出来る．

UNIX の diff[21] コマンドを利用した文字列の比較や差分情報の生成などによるソースコードの類似性評価の研究が行われている [22, 23]．しかし，文字列比較は計算コストが高く，近年増加している大規模なソフトウェアシステムには向かないため，自己組織化マップを用いた類似度測定の研究も試みられている [24]．また，C 言語のソースコードを対象に，異なる部分を構文比較によって見つけ出す Cdiff[25, 26] や構文木の比較によるソースコードの差分表示 [27] などの研究が行われている．大規模なシステム或いはソフトの無駄なコードと容量が減らせるため，抽象構文木を使

用して，大規模なコンピュータプログラムのソースコードからコードクローンを検出方法が提案された [28, 29, 30]．また，ほぼ同じ目的とした既にある CCFinderX[31] ではソースコードを字句解析してトークンの列に直してからコードクローン検出を行う．このツールではパラメータ置換が処理され，コードライン順のトークンに着目して比較される．これらの研究では，大きなシステムからの効率的コードクローン検出が有効であるが，結果は明確目的ではない．

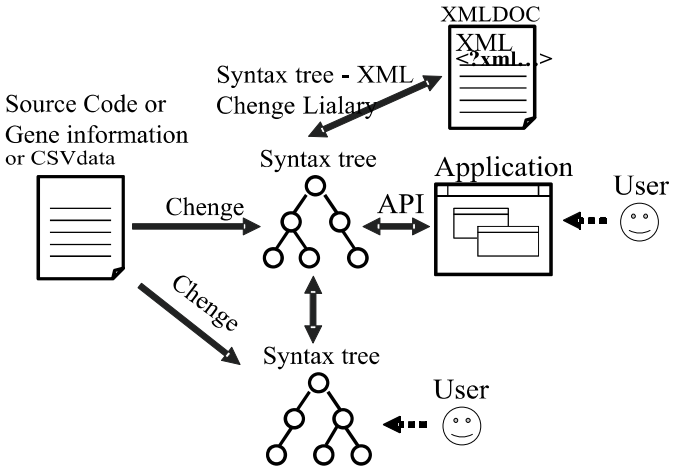


図 1.2: 木構造が持つ XML で表現されたオブジェクト例

木構造はプログラム構造を解析して類似性判定技術だけで使用されるのではなく，XML ファイルや遺伝子の進化過程など，様々なオブジェクトの表現に用いられている [32, 33, 34]（図 1.2）．このような木で表現されたオブジェクトを検索したり分類したりするには，木の類似性評価が必要である．これまで，二つの木の類似性を評価する手法として，編集距離を用いた様々な手法が提案されている．文字列の編集距離アラインメントによる木編集距離の近似 [35, 36] や複数観点を用いた木類似度算出法によるソースコードの類似構造の抽出などの研究 [37] がある．また，木編集距離を利用した木データの構造と内容の類似性を反映する分類手法 [38] や木の DP マッチングを利用した DTD 類似度の考察も行われてきている [39, 41, 42, 43]．

プログラムと文字列の類似性判別技術はソフトウェア開発効率を飛躍的に向上するための手法として，プログラムの再利用に活用されている．再利用とは既存の類

似プログラム部品を参照し，一部手直しをして利用することがある．インターネットの普及により大量のソースコードが容易に入手可能になり，これらのソースコードは，開発者にとって有益な再利用部品である可能性がある．再利用を活用するためには，過去に開発された類似しているソフトウェア部品に関する情報を入手することが必要である [44, 45, 46, 47, 48, 49]．ソースプログラムの類似度比較と類似部分の抽出はソフトウェア開発者から望まれる．

1.3 論文の目的と構成

本研究では，二つのC言語プログラムがどれぐらい類似しているか，文字列を厳密に比較して算出するのではなく，プログラムを構文木に変換し，その木グラフを用いて比較することで類似度を求めることを検討する．構文木に変換して比較することで，計算コストを下げるだけではなく，木の特性を生かした比較が出来る．これにより，文字列比較ではできなかったプログラム構造の比較が可能になり，また関数の呼び出し関係まで比較することができる．書き換えたプログラム同士がどのぐらい類似しているかを量的に測ることが本研究の目的である．プログラムの無断複製などの問題に対して，(1) 倫理教育，(2) 規則制定，(3) 不正判別技術，といった三つの観点から解決することが考られるが，ここでは，我々は (3) 不正判別技術を提案する．

本論文は以下のように構成される:

1章では研究の背景を紹介し，関連の研究状況を概説しており，また研究の目的について述べる.

2章では，研究に関連する基礎的な定義や概念を記述する．まず，木グラフ (Tree Graph) の定義や構文木（Syntax Tree）や二部グラフのマッチング（Bipartite Matching）について説明した上で，それらの基本的な性質を示す．次に文字列編集距離と木間の編集距離について説明し，最後に既存のプログラム類似性判定手法のSMMTについて紹介する．

3章では，C言語プログラムを構文木化する手法を提案する．まずはC言語ソースプログラムに対する前処理を行う．次に，C言語プログラムの構成パターンを分類し，それぞれのパターンを表現する構文木部品を作成する．更に，これらの部品を用いてC言語プログラム全体を一つの木グラフに変換する．本論文では，提案した構文木化の方法を用いて作成したCCXソフトの説明や，具体例を用いた変換結果の紹介も行う．

4章では，構文木グラフの類似度を求める手法を提案する．既に提案されたTree

Overlapping 手法を紹介した上で，すべての共通部分木を用いて構文木の類似度を測る New Tree Overlapping（NTO）方法を提案する．また，木グラフの深さを基準にノード間の最大マッチングを計算することによって構文木の類似度を測る Depth Matching（DM）手法を提案する．

5 章では，サンプルプログラムを用いて提案した二つの手法と既存の SMMT 手法との比較実験を行い，その結果について議論する．サンプルプログラムは，学生のレポートやテキストに掲載のプログラムや研究用のプログラムなどである．実験結果について考察し，提案手法の有効性を確認する．

6 章では，本論文で得られた結果をまとめており，今後の展望や課題について述べる．

付録では本研究でシミュレーションを行うため，JAVA と C ＃で作られた各スクリプトファイル（実用プログラム）を紹介する．

第 2 章

本研究の基本概念

本節では，本研究の基礎的な概念およびそれらの拡張について示す．まず，木グラフ（Tree Graph）の定義及び構文木（Syntex Tree）と二部グラフのマッチング（Bipartite Matching）について説明し，それらの基本的な性質も示す．次に文字列編集距離と木間の編集距離について説明して，最後に既存のプログラム類似性判定手法の SMMT について紹介する．

2.1 グラフの定義

2.1.1 木

木 (Tree) とは，グラフの種類のひとつで，連結で閉路をもたない無向グラフのことである（図 2.1）[50, 51]．

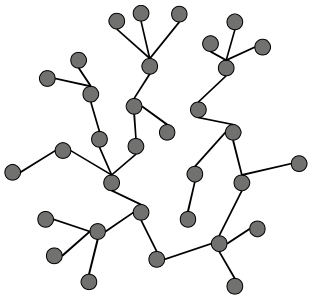


図 2.1: 木 (Tree)

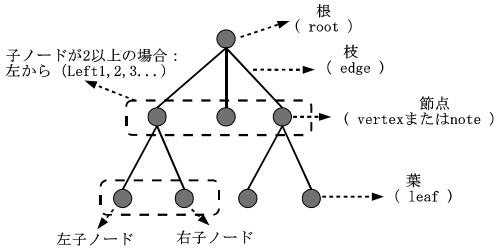


図 2.2: 木の構造

木の用語定義は以下のように示す．

- 木の各頂点をノードという．
- 木の特別な一つの頂点を根といい，根の指定された木を根つき木という．

- (根以外) 次数 1 の点を葉という．
- 根からの道の長さを深さという．
- 最大の道の長さを高さという．

以下では，木構造 [52] について説明する．木の中に含まれる木を**部分木** (subtree) という（図 2.3)．木を構成する頂点のことをノードといい,（図 2.2）木 T に含まれるノードの集合を $V(T)$ と表す．1 つのノード r を**根** (root) として選んだ木を**根付き木** (rooted tree) といい，r を $r(T)$ と表す．また，根付き木 T の頂点 v を根とする部分木を $T(v)$ と表す．根付き木の根から頂点 v への経路上の頂点を v の**先祖** (ancestors) といい，v の先祖を u とすると，このときの関係を $v < u$ と表記する．u が v を先祖に持つならば，u は v の**子孫**（descendant）であるという．また，特に v に隣接する先祖を，v の**親** (parent) といい，$par(v)$ と表記する．u が v の親のとき，v は u の**子** ($child$) といい，i 番目の子を u_i と表記する．また，頂点 v の子の数を，v の**次数** (degree) といい，$deg(v)$ と表記する．次数が 0 の頂点を特に**葉** (leaf) といい，木 T の葉の数を $leaves(T)$ と表記する．

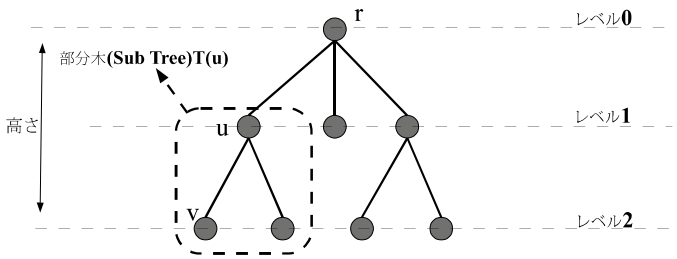


図 2.3: 木の中に含まれる部分木

各頂点がアルファベット $\sum$ でラベル付けされている木を，**ラベル付き木** (labeled tree) という，また，各頂点の子に左右の順序を指定している根付き木を，**根付き順序木** (rooted ordered tree) といい，順序を指定していない根付き木を，**根付き無順序木** (rooted unordered tree) という [53]．本研究では，ラベル付き根付き順序木を順序木，あるいは，単に木という．

2.1.2 構文木

構文木とは，プログラミング言語の構文を表わす木構造のことである [54]．木グラフで表現したプログラムを構文木として見なすことが出来る（図 2.4）．それぞれのノードに対応する文の記述をノードラベルとすれば，二つのプログラムから変換された構文木のラベルを比較することで，プログラムの類似性を測ることが出来る．

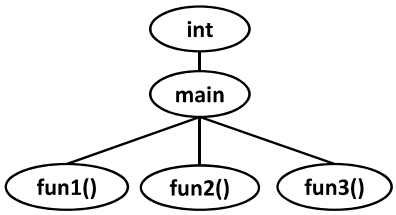


図 2.4: プログラムの構文木一例

2.1.3 木間写像と最大共通部分木

木の間ノード編集操作 [55, 56, 57, 58] を繰り返して木 T_1 を木 T_2 へ編集することを $T_1 \rightarrow T_2$ と表す．木の変換を変換中に行われるノード編集の集合として表現できる．これを木間写像 M を用いて $M(T_1, T_2)$ と表す．ここで，$T_i(T_j)$ は木 T の $i(j)$ 番目のノードであり，$M(T_1, T_2)$ におけるノード編集ペア $(T_1[i], T_2[j])$ は以下の性質をみたす．

(1) $i_1 = i_2$ のとき $j_1 = j_2$．(1 対 1 対応)
(2) $T_1[i_1]$ が $T_1[i_2]$ の左側存在するとき $T_2[j_1]$ は $T_2[j_2]$ の左側存在．(兄弟の性質保存)
(3) $T_1[i_1]$ が $T_1[i_2]$ の祖先であるとき $T_2[j_1]$ は $T_2[j_2]$ 先祖である．(先祖の性質保存)

ノード編集操作 (x，y) においてかかるコストを関数 c(x，y) で表す．写像 $M(T_1$ と $T_2)$ において木 T_1 から削除されるノードの集合を D_M，木 T_2 に挿入されるノードの集合を I_M，木 T_1 の要素を木 T_2 の要素へ置換するノードの集合を S_M としたとき，$M(T_1, T_2)$ における編集コスト $\mathrm{cost}(M, T_1, T_2)$ を以下のように M における

全置換操作におけるコスト，全削除操作におけるコスト，全挿入操作におけるコストの和として次の式で定義 [59] できる．

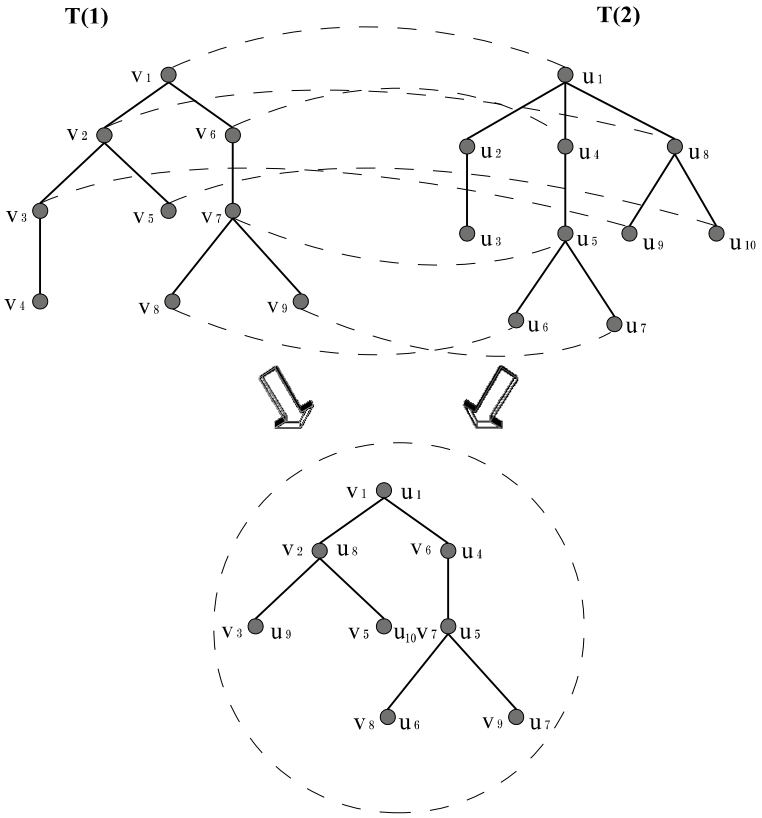


図 2.5: 木間写像と最大共通部分木

$$cost(M) = \sum_{(i,j)\in S_M} \mathrm{c}(T_1[\mathrm{i}],T_2[\mathrm{j}]) + \sum_{(i,j)\in D_M} \mathrm{c}(T_1[\mathrm{i}],\lambda) + \sum_{(i,j)\in I_M} \mathrm{c}(\lambda,T_2[\mathrm{j}])$$

最大共通部分グラフ問題は，複数のグラフに共通の連結部分グラフのうち，辺の数が最大であるものを求めるという問題である．いままで色々の手法が提案されて来た [60, 61, 62, 63]．最大共通部分木求める並列アルゴリズム [64]，二つの木の最大共通部分グラフを求めるアルゴリズム [65]．ここで 2.5 のように最大共通部分木が求めると図 2.5(点線に含まれた部分) のように得られる．

2.2 二部グラフの定義

グラフ $G = (V, E)$ で，V が適当な V_1 と V_2 に分割でき，E のどの辺も一方の端点は V_1 に属し，他方の端点は V_2 に属するとき，G は二部グラフ（bipartite graph）という（図 2.6 の (a)）．完全二部グラフ (complete bipartite graph) とは，V_1 の各点が V_2 の各点とちょうど 1 本の辺で結ばれている二部グラフである（図 2.6 の (b)）．二部グラフの辺集合 M がマッチングであるとは, M に属するどの 2 辺も隣接していないと言うことである（図 2.6 の (c)）．

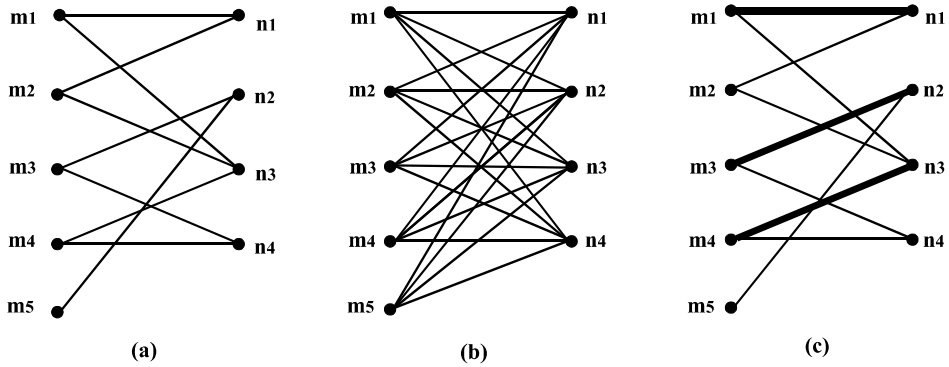


図 2.6: 二部グラフ

二部グラフとマッチング問題

二部グラフ G=(V, E) において，M が E の部分集合で，かつ M のどの 2 辺も共通の端点をもたないとき，M を G のマッチング (matching) という．辺の本数が最大のマッチングを最大マッチング (maximum matching) という．あるマッチング M のどの 2 辺も交差していないならば M を G の**ノンクロスマッチング** (Noncross matching) という．本研究では，ノンクロスマッチングをマッチングという [66, 67, 68]．

図 2.7 について，(a) は辺 m_1,n_1 と m_2,n_1 が共通の端点が n_1 持つのでマッチングではない．(b) の点 m_5 と n_4 がマッチングが取れるため，最大マッチングではないが，はマッチングである．(c) は最大マッチングであるが，辺 m_3,n_3 と m_4,n_2 はクロ

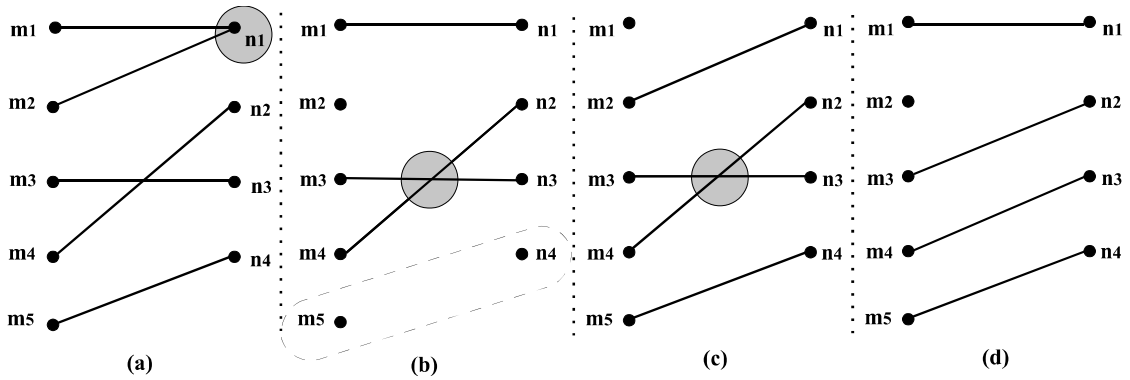


図 2.7: 二部グラフのマッチング例

スしているため，ノンクロスマッチングではない．(d) はノンクロス最大マッチングである．

2.3 編集距離

2.3.1 文字列編集距離

二つの文字列の近さを計るために片方の文字列に次の操作を繰り返し適用して，もう一方の文字列を得るための操作回数の最小値を，その二つの文字列編集距離という [69, 70]．使用可能な操作は以下の 3 種類である [71, 72]．

1. 削除：一つの文字を取り除く．
2. 挿入：一つの文字を新たに付き加える．
3. 置換：一つの文字を別の文字で置き換える．

例えば,「$weight$」と「$write$」の編集距離を考えるとき，次のように操作を繰り返し適用すると，この二つの編集距離が 4 以下であることが図 2.8 のように分かる．

- $weight$
- $weighte$(挿入：e)
- $wrighte$(置換：$e \rightarrow r$)

- $writhe$(削除：g)
- $write$(削除：h)

		1	2	3	4	5	6
	0	W	e	i	g	h	t
1	W	0	1	2	3	4	5
2	r	1	1	2	3	4	5
3	i	2	2	1	2	3	4
4	t	3	3	2	2	3	3
5	e	4	4	3	3	3	**4**

図 2.8: weight と write の編集距離

また，3 回以下の操作の適用では「$weight$」を「$write$」にすることができないので，「$weight$」と「$write$」の編集距離は 4 となる．

2.3.2 木編集距離の定義

木編集距離は木構造に対する距離の指標の一種で，ある一方の木 T_1 に対しノードの追加，削除，置換などの操作を行い，もう一方の木 T_2 と同じ構造に変形させるのに必要な操作のコストをその二つの木の距離とする手法である [73, 74]．即ち T_1 を T_2 に変換する最小編集操作回数（最小コスト）である．木編集距離の三つの操作ノードの挿入，削除，置換に対して，以下のようにそれぞれ述べる [59]．

- ノード挿入

葉としてノードが挿入する場合，葉の部分にそのままノードを新たに付け足すだけである．また，葉ではなく図 2.9 のように要素 y のノードを u_i とノードを u_j の間 (節) に挿入するときは u_{i+1}〜u_{j-1} がノード y の子供となる．

- ノード削除

削除対象のノードが葉の場合，そのノードのみを削除する．葉ではなく削除対象のノードに子供がいる場合も図 2.10 のように削除対象のノードのみを削除し，その子供は削除したノードの親の子供になる．

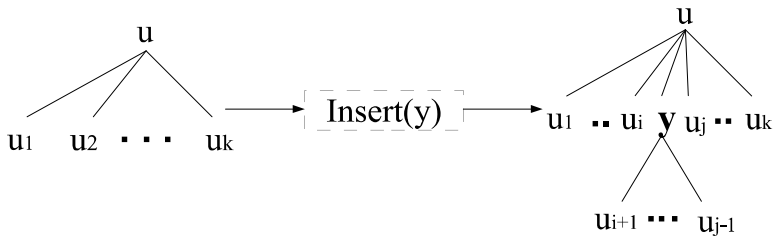


図 2.9: ノードの挿入

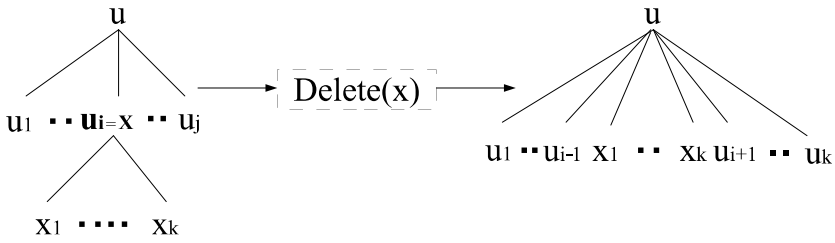


図 2.10: ノードの削除

- ノード置換

置換操作はノードのラベルを編集するだけで，木の構造は変化しない図 2.11．ここで置換操作とは異なる要素へ置換することだけではなく，同じ要素へ置換する場合も含む．

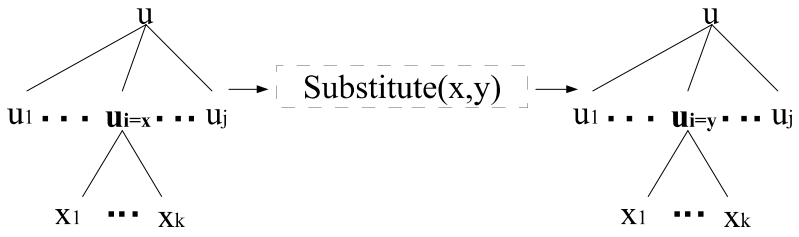


図 2.11: ノードの置換

ここで，空ノード λ を導入することによって挿入，削除，置換ノード編集操作を $(x,\ y)$ のノードペアの形で統一的に表現できる．$(x,\ y)$ はノードをノード y に入れ替えるという操作で，$x{\neq}\lambda$ かつ $y{\neq}\lambda$ のときがノード置換操作で，

$x{=}\lambda$ かつ $y{\neq}\lambda$ のときが挿入操作で，
$x{\neq}\lambda$ かつ $y{=}\lambda$ のときが挿入操作を意味する．

2.4 SMMT 手法

二つのソースファイル P と Q について，それぞれの要素の集合は，$P{=}\{p_1, \cdots, p_n\}$ と $Q{=}\{q_1, \cdots, q_m\}$ と書く，ここで，各 p_i はファイル P の各行である．二つのファイル P と Q に対し，対応 $R{\subseteq}P\mathrm{x}Q$ が得られるとする．図 2.12 のように対応 R がファイルの差分抽出ツール diff により与えられる一致部分である．P と Q の R に対する類似度 $S(0 \leq S \leq 1)$ を以下のように定義する [75]．

$$S(P,Q) = \frac{\mid \{p_i \mid (p_i, q_j) \in R\} \mid + \mid \{q_i \mid (p_i, q_j) \in R\} \mid}{\mid P \mid + \mid Q \mid} \tag{2.1}$$

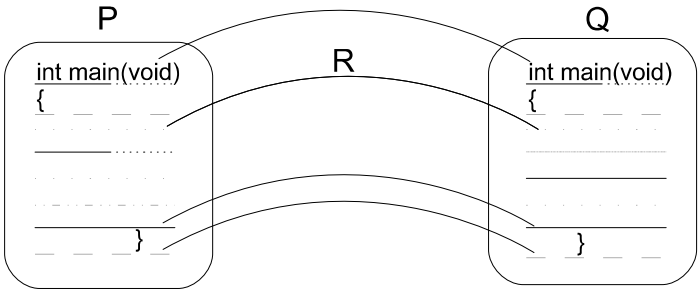


図 2.12: SMMT 対応例

第 3 章

C 言語プログラムの構文木化手法の提案

本章では，C 言語プログラムを構文木化手法を提案する．まずは C 言語ソースプログラムについて前処理を行う．次に我々は C 言語プログラム構成のパターンを分析し，分析したパターンをすべて表現できる構文木部品を提案し，これらの部品を使い C 言語プログラム全体を木グラフに変換する CCX ソフトを提案する [76]．最後に変換手法について例を用いて説明する.

3.1　プログラムの構文木化の目的

C 言語は手続き型言語であり，基本的には記述した順番にデータが処理されていくプログラム言語である．C 言語プログラムは一つ以上の関数から構成される [77, 78]．それぞれの処理に対応する関数を作成することで，複雑な処理に対しても関数の組み立てによって効率よくプログラムを構成することができる．C 言語のプログラムを構文化の目的としては，木の特性を生かして類似度の比較を行いたいからである．この木の特性というのはノードに対する親子関係を見ることで関数で呼び出された関数の比較が可能である．つまり，関数の呼び出し関係を比較することが可能になる．具体的には木の部分木に焦点を当ててその中に含まれるノードの個数を用いて比較ができるようになる．

C 言語のプログラムを構文化の方法としては，C 言語プログラムは，(1) main 関数を根とする，(2) 関数中の実行文などを記述順にその子供とする，(3) 関数呼出しがあれば，それに対応する子供を作成し，その実行文などについては (2) と同様の処理を行う（関数の展開），という処理を行うことで一つの木グラフとして表現することができる．関数を展開する時の留意点としては，一度展開した関数は二度と

は展開しない．このため関数を展開する前にすでに展開したものかどうか判別する必要がある．他には，代入文などで関数が呼び出される場合もある．その場合代入文の子に関数を置く必要がある．さらに関数の引数が関数だった場合も引数を関数の子にする．

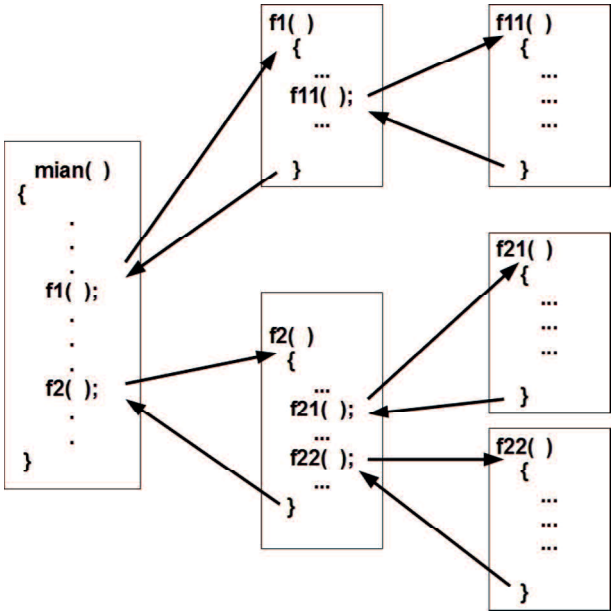


図 3.1: C プログラムの関数関連図

C 言語プログラム全体の構成は図 3.1 に示すような関数同士の関連図で表現することができる．関数内の実行文や制御構造をあらかじめそれぞれ「部品」として作成しておく．なお，関数については，ライブラリ関数とプログラム中に定義される関数（自作関数と呼ぶ）があるが，我々は，ライブラリ関数は部品化して扱い，自作関数については，最初に呼び出された箇所のみに展開することとする．これにより，与えられたプログラムに対して，唯一の木構造表現が可能となる．

3.2 プログラムの構文木の作成

本節では，まずはプログラムに対する前処理について説明し，次に構文木の構成部品について提案する．最後に構成した構文木 XML 形式の構造について説明する．

3.2.1 プログラムの前処理

前処理とは，与えられたプログラムについて，我々の構文木作成において必要でないものや書き換える必要のある部分に対する処理である．構文木作成に必要でないものとして，#include のようなプリプロセッサ文，関数宣言，構造体の宣言，コメント文，空行などがある．これらの部分については，あらかじめ削除することにする．また，プログラムの実行や結果に影響のない部分（例えば，複数行に渡って書かれた実行文，複数のスペースが入っている実行文など）については，書き換えて正規化する．以上のような前処理の一覧を 3.1 に示す．図 3.2(1) に示すプログラムに対し，3.1 の前処理を行うと，図 3.2(2) に示すようなプログラム表記になる．

表 3.1: 前処理一覧

前処理 i	プリプロセッサ文の削除.
前処理 ii	関数外で定義された構造体やグローバル変数の削除.
前処理 iii	コメントや空行の削除.
前処理 vi	関数宣言の削除.
前処理 v	複数のスペースを 1 つにまとめる．ただし，文字リテラルとしてのスペースは例外とする.
前処理 vi	ブロック（”{”と”}”）の処理 :”{”と ”}”を単独の行に書き換え，制御文のブロックについては，”{”と ”}”がなければ追加する.
前処理 vii	全ての実行文を 1 行にまとめる.

プログラムA

```
/*
  階乗を計算するプログラ
ム
*/

#include<stdio.h>
int main(void);
int kaijo(int);

//メイン関数
int main(void)
{
     int a;
     a = kaijyo(5);
     printf("5! =%d\n", a);
     return 0;
}

//自作関数
int kaijyo(int n)
{
     if(n == 1){
          return 1;
     }else{
          return n * kaijyo(n-
1);
     }
}
```

(1) 与えられたプログラム

プログラムB

```
int main(void)
{
int a;
a = kaijyo(5);
printf("5! =%d\n", a);
return 0;
}
int kaijyo(int n)
{
if(n == 1)
{
return 1;
}
else
{
return n * kaijyo(n-1);
}
}
```

(2) 前処理されたプログラム表記

図 3.2: サンプルプログラムの前処理

3.2.2 構文木の構成部品

C 言語プログラムの関数は大きく分けると，変数宣言文，式（代入文を含む），制御文，関数呼出文といった実行文から構成される．これらの実行文に対応した部品を作り，記述順に関数のノードの子供としてつなげば，その関数に対応した構文木の作成ができる．図 3.3 は我々が提案する構文木の構成部品一覧である．これらの 8 種類パターンは C 言語のすべての関数を構成でき，これらの部品は，それぞれの実行文を構成する個々の要素を表現できるように作成している．図 3.3 のパターン 4 については，if()else だけを示しているが，if() の場合はパターン 6 の no ノードがなくなり，if()elseif の場合は，elseif 文はパターン 4 の no の位置に現れ，その位置から

新たな if 構文が展開されることになる．

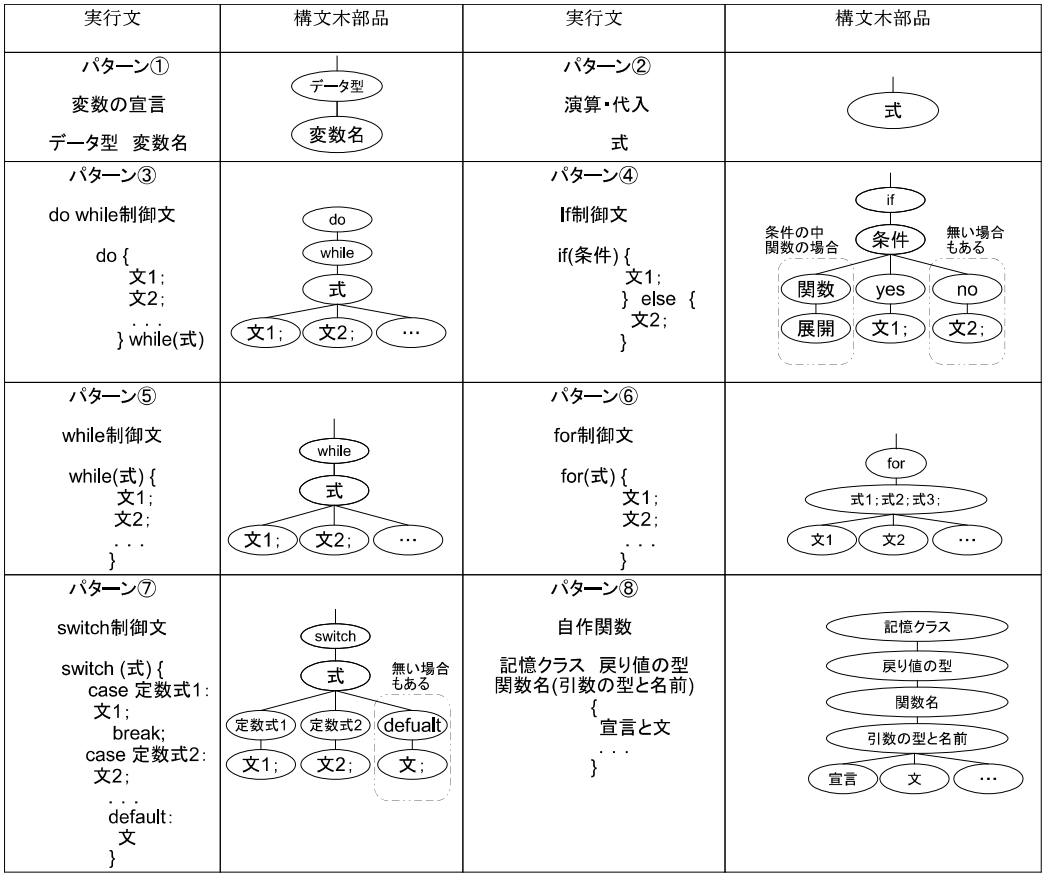


図 3.3: 構文木部品一覧

3.2.3 構文木の作成

関数には，ライブラリ関数と自作関数があり，我々は自作関数だけを展開するように構文木を作成する．また，一つの関数はプログラムの複数の位置で呼出される可能性がある．呼び出される全ての位置で関数を展開すると，構文木のサイズが大きくなり，再帰呼び出しの場合は構文木のサイズが無限になる。そのために，自作関数を 1 回だけ展開するようにする必要がある．自作関数を 1 回だけ展開し，かつ

その位置を一意に決定するために，次の手順で処理を行う．

0° プログラムの main 関数を選択する．

1° 関数内の全実行文を順に解析し，これまで呼び出したことのない自作関数なら，それを FIFO キューに入れ，呼び出した位置も記録する (FIFO とは，データの格納と取り出しに関する方式のひとつで，最初に格納したデータから取り出す方式のことである)．

2° キューが空ならば，停止。そうでなければ，キューの先頭から一つの関数を取り出し，1° へ．

以上で得られた自作関数の呼出しの位置を利用し，それぞれの関数の構文木をつなぎ合わせば，プログラム P に対応する構文木 $T = (N, E, L)$ は一意に定まる．ここで，N は木グラフ T のノード集合であり，E は木グラフ T の辺集合である．また，L はノードのラベル関数であり，それぞれのノードに対応するプログラムの記述である．このように生成される構文木は根付き順序木であることに注意されたい．図 3.2 (1) に示すプログラムに対し，3.1 の前処理を行い，さらに，図 3.3 の構文木部品を用いて，以上で述べた構文木作成の処理を行うと，図 3.4 のような構文木グラフが生成される．

3.3 XML 形式の構文木の作成およびその構造

我々は，3 節で述べた構文木作成の方法に基づいて，与えられた C 言語プログラムの XML 形式の構文木を生成するソフトウェアを開発した．このソフトは前述のように前処理された C 言語ソースコードを XML 形式のデータに変換するものであり，我々はこのソフトを CCX と名付ける (図 3.5)．

XML は文書やデータの意味や構造を記述するためのマークアップ言語の一つである [79, 80, 81]．マークアップ言語は,「タグ」と呼ばれる特定の文字列で地の文に情報の意味や構造，装飾などを埋め込んでいく言語のことで，XML はユーザが独自のタグを指定できることから，マークアップ言語を作成するためのメタ言語とも言

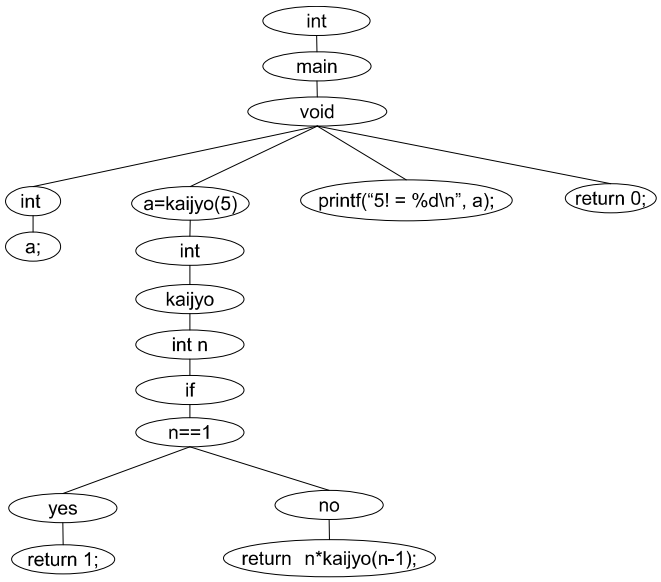


図 3.4: 作成された構文木

われる．

図 3.5: 標準化 C プログラムを XML 木構造に変換される CCX ソフト

CCX の実行例および生成された XML 形式の構文木について説明する．CCX は 1

つのファイルにまとめられた前処理後の C 言語ソースコードを入力とする．図 3.2(2) に示す前処理されたソースコードを CCX で実行すると，その対応する XML 形式の構文木は図 3.9 に示したものとなる．ただし，図 3.9 には main 関数に対応する部分だけを表示している．図 3.9 の XML 形式の構文木には，funcType，funcName，funcArgType，funcContents といったノード（要素）が含まれる．funcType は関数型に対応するノード，funcName は関数名に対応するノード，funcArgType は関数の引数の型と名前に対応するノードである．これ以外のすべてのノードの名前（タグ名）は funcContents である．また，#text は，XML の要素のテキストであり，ノードに対応する具体的なソースコードである．

構文木のノードに関する情報は属性として表される。図 3.8 は図 3.9 の XML 構文木に対応する木グラフである．図に示すように属性 level は木グラフにおける根からの階層数，属性 label は順序木とみなす木グラフにおいて深さ優先順による探索時の順番を表す．但し，図 3.9 中の一番上の funcType ノードが図 3.8 の根に対応し，この根の level は 0，label は 1 である．numofchildren はノードの子供の数を表す属性と定義する．例えば，図 3.8 中の属性 label が 4 のノードについて，その level（根からの階層数）は 3 であり，#text にはソースコードとして文字列「int」が含まれ，属性 numofchildren は 1 である．

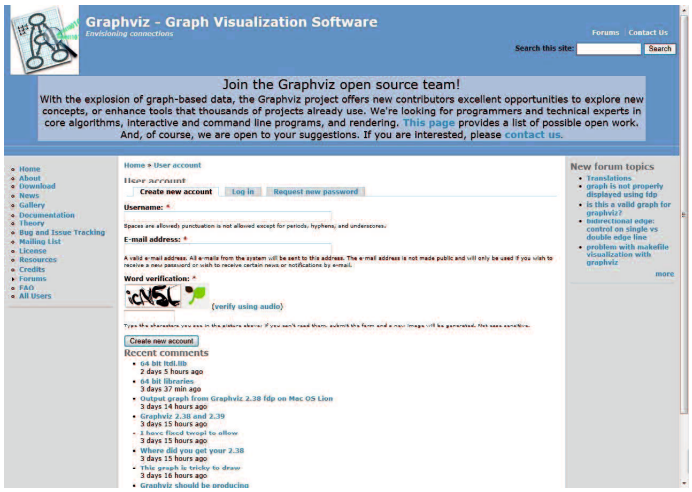


図 3.6: 可視化ツール Graphviz

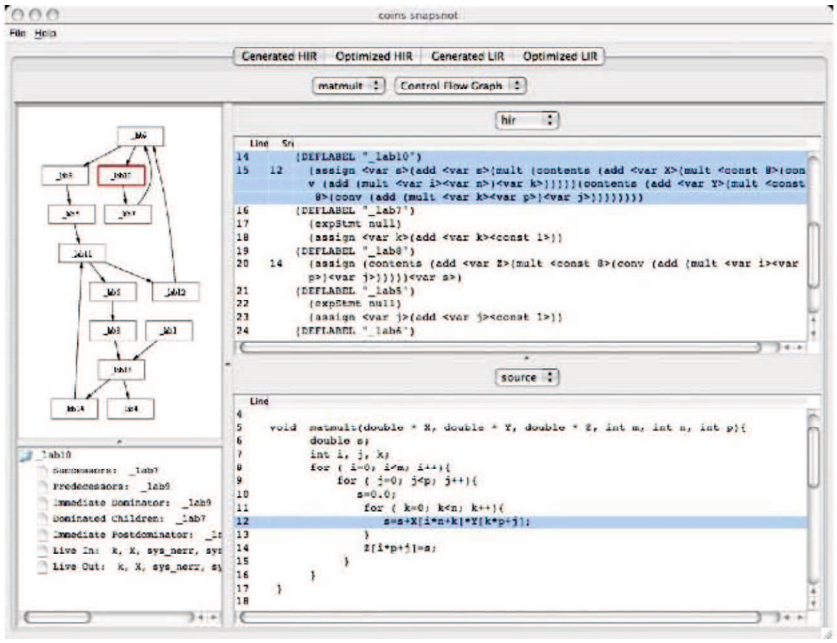


図 3.7: 中間プログラム表現可視化ツール CoVis

XML で記述された木構造のデータを図 3.8 のように可視化するフリーソフトウェアが多く開発されている．ここでは，中間プログラム表現可視化ツール [82]CoVis（図 3.7）と XML を可視化ツール [83]Graphviz（図 3.6）を提供する．それらのソフトウェアと我々が開発した CCX と合わせて利用すれば，C 言語プログラムを容易に視覚的に表現することができる．図 3.2 に示すサンプルプログラムは，二つの関数 main と kaijyo からなり，プログラムはは図 3.8 で示す構造になる．図 3.8 から分かるように，kaijyo 関数は属性 label が 6 のノードで呼び出されており，プログラム全体の構造は根付き順序木である図 3.9 のようになる．このように，我々のソフトは複雑なソースコードを簡潔なグラフに変換でき，関数内の構造や関数間の呼び出し関係，またプログラム全体の実行の流れを陽に表現することができる．

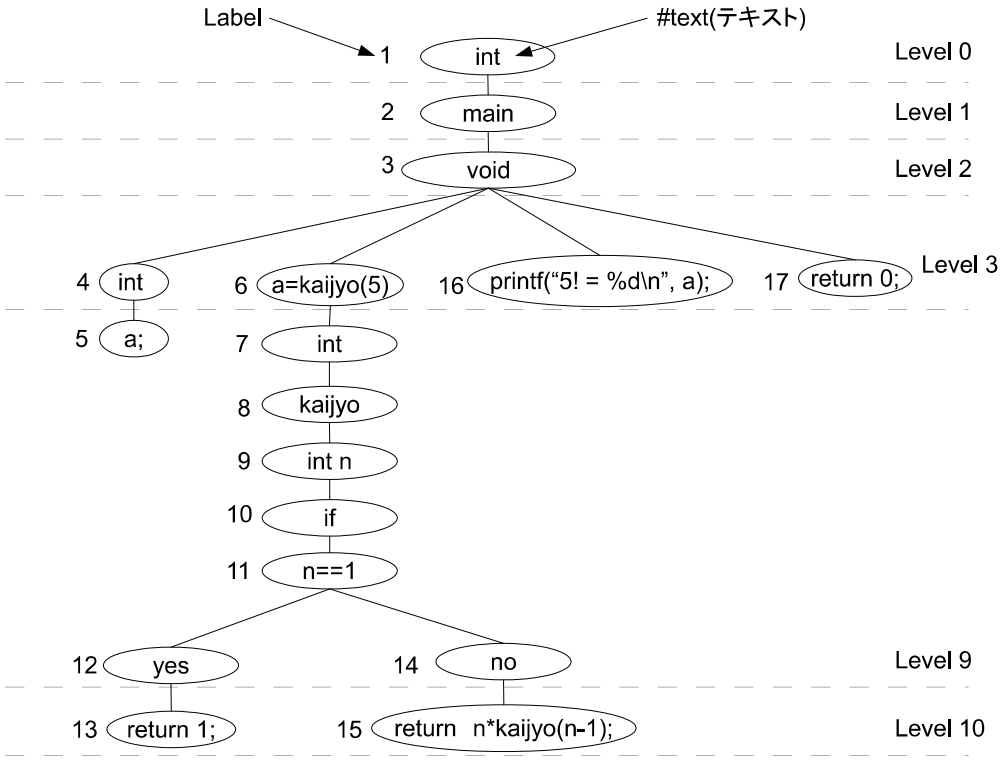


図 3.8: サンプル XML 構文木の可視化

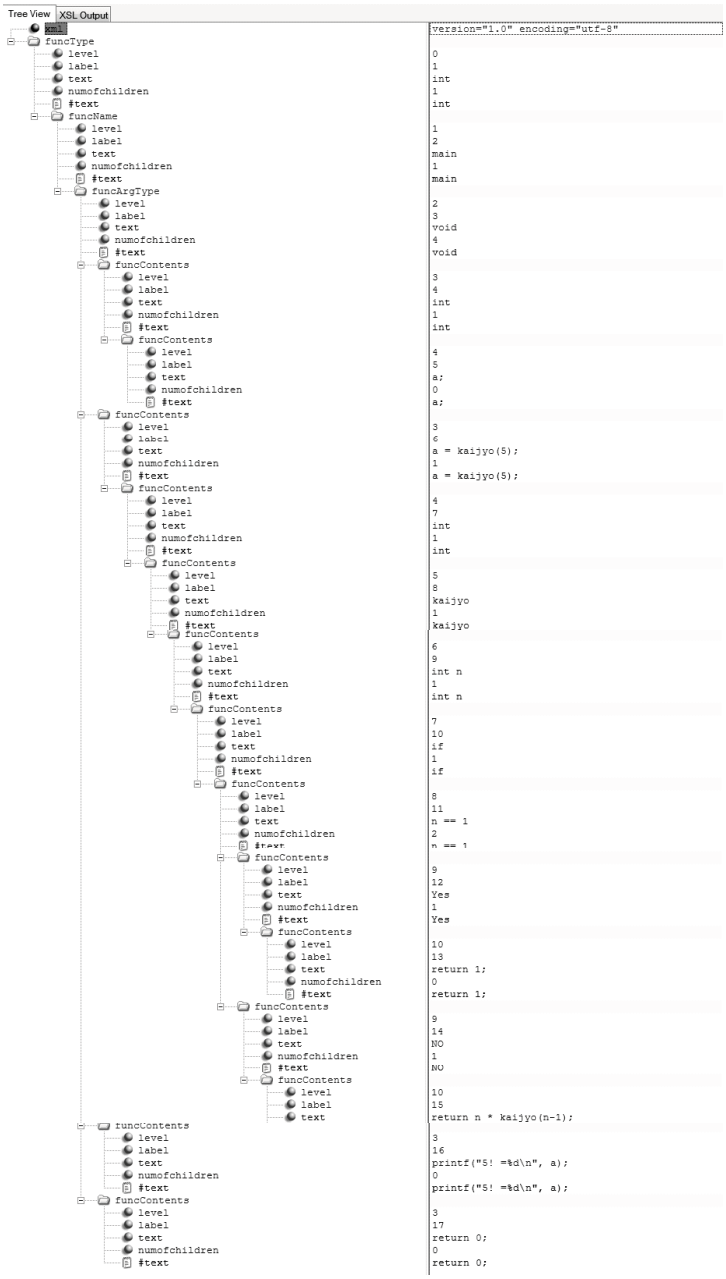


図 3.9: XML 構文木のサンプル（左半分が構文木，右半分は属性やテキストの値）

第 4 章

木グラフの類似度を求める手法の提案

本章では構文木グラフの類似度を求める手法を提案する．ここで，既に提案された Tree Overlapping[84] 手法を紹介し，これを踏まえた上で，すべての共通部分木を求める新たな New Tree Overlapping（NTO）と Depth Mathing（DM）手法を紹介し，サンプルプログラムを用いて我々の提案した二つの手法で実験を行うことで具体的な方法を説明する．

4.1 Tree Overlapping 手法

T_1=(N_1, E_1, l_1) と T_2=(N_2, E_2, l_2) が二つの木グラフとする．l_1 と l_2 はノードのラベル関数であり，$l_1(x)$ は T_1 のノード x のラベルを表す．$TreeOverlapping$ の方法による T_1 と T_2 の類似度 S_{TO}(T_1,T_2) は次のように計算される．まず，(i) T_1 のノード n_1 と T_2 のノード n_2 が同じ位置に重なるように，T_1 と T_2 を重ね合わせることを考えた時の「重なった導出規則」の数 $C_{TO}(n_1, n_2)$ を計算し，次に，(ii) $S_{TO}(T_1, T_2)$ を次の式で計算する．

$$S_{TO}(T_1, T_2) = \max_{n_1 \in N_1} \max_{n_2 \in N_2} C_{TO}(n_1, n_2) \tag{4.1}$$

$C_{TO}(n_1, n_2)$ の計算は次の式に従って行う．

$$C_{TO}(n_1,n_2)=\left|\left\{(m_1,m_2)\;\middle|\;\begin{array}{ll}\wedge & m_1\in nonterm(T_1)\\ \wedge & m_2\in nonterm(T_2)\\ \wedge & (m_1,m_2)\in L(n_1,n_2)\\ \wedge & prod(m_1){=}prod(m_2)\end{array}\right\}\right|$$

ただし，

- m_1，m_2 は T_1，T_2 の任意のノード．
- i はノード n の i 番目の子ノード．
- $nonterm(T)$ は T の非終端ノードの集合．
- $ch(n,i)$ はノード n の i 番目の子ノード．
- $prod(n)$ はノード T_1 の子ノードの導出規則，(n に子ノード (左から順に)x,y,z が存在する場合，$prod(n){=}(l_j(n){\rightarrow}l_j(x)l_j(y)l_j(z))(j{=}1,2)$．
- $L(n_1,n_2)$ は「n_1 と n_2 を同じ位置にして，T_1 と T_2 を重ね合わせた時の．重なるノードの組」の集合．

$L(n_1,n_2)$ は次の条件を満たす．

(1) $(n_1,n_2)\in L(n_1,n_2)$．
(2) $(m_1,m_2)\in L(n_1,n_2)$ ならば，$(ch(m_1,i),ch(m_2,i))\in L(n_1,n_2)$.
(3) $(ch(m_1,i),ch(m_2,i))\in L(n_1,n_2)$ ならば，$(m_1,m_2)\in L(n_1,n_2)$.
(4) (2)-(3) を再帰的に適用してもたどり着けないものは L(n_1,n_2) に含まれない．

ここでは，図 4.1(1) に示した T_1 と T_2 を例として,Tree Overlapping 手法を用いた類似度計算の具体例を示す． まず，$C_{TO}((T_1,A),(T_2,A))$ を求めると，T_1 と T_2 のノード A が重なるように T_1 と T_2 を重ねあわせたのが図 4.1(2) である．$C_{TO}((T_1,A),(T_2,A))=2$ となる．同様に，$C_{TO}((T_1,H),(T_2,H))$ を求めると，T_1 と T_2 のノード H が重なるように T_1 と T_2 を重ねあわせたのが図 4.1(3) である．$C_{TO}((T_1,A),(T_2,A))$=1 となる．Tree Overlapping 手法によって，$C_{TO}(T_1,T_2)$=2 と得られる．

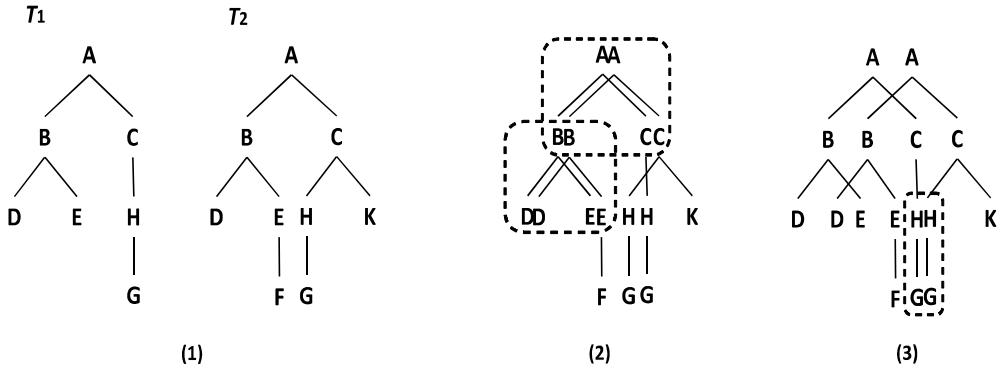


図 4.1: Tree Overlapping 手法例

Tree Overlapping 手法では，$S_{TO}(T_1, T_2)$ が得られると同時に，T_1 と T_2 を重ね合わせた時に重なる最大のノードの組を得ることも可能である．つまり，$P_{TO}(n_1, n_2)$ を次のものとし，

$$P_{TO}(n_1, n_2)= \left| \left\{ (m_1, m_2) \;\middle|\; \begin{array}{ll} \wedge & m_1 \in nonterm(T_1) \\ \wedge & m_2 \in nonterm(T_2) \\ \wedge & (m_1, m_2) \in L(n_1, n_2) \\ \wedge & prod(m_1)=prod(m_2) \end{array} \right\} \right|$$

$|P_{TO}(n_1, n_2)|=S_{TO}(n_1, n_2)$ を満たす $P_{TO}(n_1, n_2)$ を求めればよい．言い換えれば，二つの木が重ね合わせられ，それぞれ n_1 と n_2 を根とする最大の共通部分木を求めることができる．

4.2 New Tree Overlapping（NTO）手法の提案

ここで，我々は $|P_{TO}(n_1, n_2)|=S_{TO}(n_1, n_2)$ を満たす $P_{TO}(n_1, n_2)$ を求めればよい．言い換えれば，二つの木が重ね合わせられ，それぞれ n_1 と n_2 を根とする最大の共

通部分木を求めることができる．しかし，この方法を適用して二つのプログラムの類似性を判定するには次のような問題を解決する必要がある．

(1) 二つのプログラムの類似性を評価するには，重ね合わせられる最大の共通部分木だけでなく，それ以外の重ね合わせられる共通部分木や終端ノードも考慮に入れなければならない．
(2) プログラムの関数名はエディタ等のツールで簡単に置き換えられる．

このような書き換えだけで得られたプログラムは類似していると判定されるべきである．しかし，Tree Overlapping 法では関数名を異なる名前にするだけで，重ね合わせられる最大部分木がかなり小さくなる．以上のことを考えて，我々は二つのプログラム P_1 と P_2 の類似度を次のように提案する．

$$S(P_1, P_2) = |N_c|/\max\{|N_1|, |N_2|\} \tag{4.2}$$

ここで，T_1=(N_1, E_1, l_1) と T_2=(N_2, E_2, l_2) はプログラム P_1 と P_2 のそれぞれの木構造グラフであり，N_c は T_1 と T_2 が重ね合わせられる全ての共通部分木のノードの集合である．N_c は次のような方法で求める．

0° T_1 と T_2 に対し，自作関数のノードラベルをすべて λ に置き換えて T_1 と T_2 を更新し，$N_c \leftarrow \phi$ とする．

1° 更新された T_1 と T_2 に対し，$|P_{TO}(n_1, n_2)|$=$S_{TO}(T_1, T_2)$ を満たす $P_{TO}(n_1, n_2)$ を求める．$P_{TO}(n_1, n_2)$=ϕ ならば，停止．

2° $N_1' \leftarrow \{n_1\} \cup \{n | n \in IS(x), (x, y) \in P_{TO}(n_1, n_2)\}$ と $N_2' \leftarrow \{n_2\} \cup \{n | n \in IS(y), (x, y) \in P_{TO}(n_1, n_2)\}$ を計算する．さらに T_1 と T_2 において，N_1' と N_2' に含まれる全てのノードをそれぞれ n_1 と n_2 に縮約するように T_1 と T_2 を更新する．ただし，$IS(x)$ は x の子ノードの集合である．

3° $N_c \leftarrow N_c \cup N_1'$ を更新し，1° へ．

同じ図 4.1 に示した T_1 と T_2 を例として,New Tree Overlapping [85] 手法を用いた類似度計算すると，図 4.1(2) のように $C_{TO}((T_1, A), (T_2, A)) = 2$ と，図 4.1(3) のように $C_{TO}((T_1, H), (T_2, H))$=1 になる．NTO 手法によって，$C_{TO}(T_1, T_2)$=3 と得られる．

サンプルプログラムの紹介

ここで，図 4.2 はC 言語サンプルプログラム P である．プログラム P が 5 つの関数からなり，関数間の呼び出し関係は図 4.3 のように表している．図 4.4 はプログラム P から作られた構文木 T を表している．Table 4.1 と Table 4.2 は構文木 T のそれぞれのノードに対応しているC プログラムのソースコードである．

```
main()
{
strprintf_a_b();
strchange_a_b();
printf("\nchenge\n\n");
strprintf_a_b();
toupper_a_b();
printf("\n chenged\n\n");
strprintf_a_b();
}
void strprintf_a_b()
{
printf("a[]=%s\n",a);
printf("b[]=%s\n",b);
}
```

```
void strchange_a_b()
{
char temp[80];
int i;
for(i=0;a[i]!=`\0`;i++){
temp[i]=a[i];
}
temp[i]=`\0`;
for(i=0;b[i]!=`\0`;i++){
a[i]=b[i];
}
a[i]=`\0`;
for(i=0;temp[i]!=`\0`;i++){
b[i]=temp[i];
}
b[i]=`\0`;
}
```

```
void toupper_a_b()
{
int i;
for(i=0;a[i]!=`\0`;i++)
{
a[i]=to_upper(a[i]);
}
for(i=0;a[i]!=`\0`;i++)
{
b[i]=to_upper(b[i]);
}
}
```

```
char to_upper(char c)
{
char cc;
if((c>=`a`)&&(c<=`z`))
{
cc=c-`a`+`A`;
}
else
{
cc=c;
}
return cc;
}
```

図 4.2: サンプルプログラム P

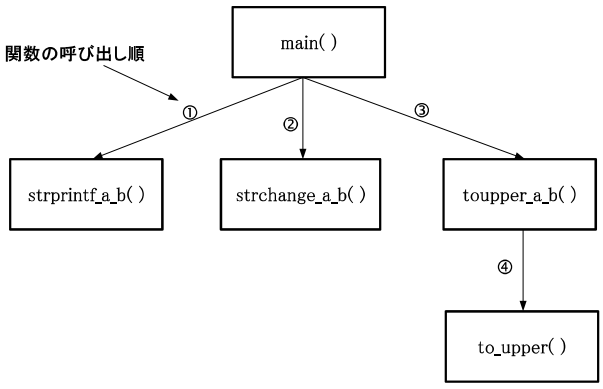


図 4.3: プログラム P の関数関係

図 4.5 は 2 つのプログラムから作られた構文木を表している.T_1 は図 4.2 に示したプログラム P の構文木であり,T_2 は, 図 4.2 のプログラム P に対して, 以下三つの変換操作から成られた新たなプログラムの対応構文木である．

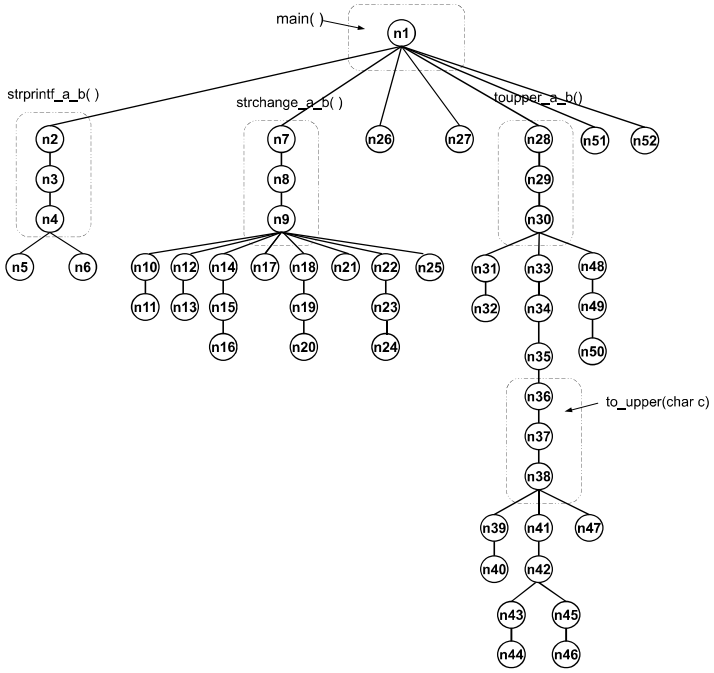


図 4.4: サンプルプログラム P から変換された構文木グラフ T

(i) 変数 i を j に変更.

(ii) 自作関数 “$strprintf\_ab$” が “fun_1” に, “$strchange\_ab$” が “fun_2” に，“$toupper\_ab$” が “fun_3” に，“$toupper$” が “fun_4” ” に変更.

(iii) (i) と (ii) を両方変更.

このような, プログラムの構造自体はそのままである．二つの提案手法における, これら 2 つの構文木の類似度はそれぞれ自作関数名だけ，引数名だけと引数と自作関数両方変換した上の 3 つの場合の類似度の計算過程は以下に示す.

NTO によるサンプルの計算

ケース (i) 自作関数名だけ変換した場合

まず，NTO 手法を用いて図 4.5 を実験する．T_1 と T_2 に対して，関数だけ変換し

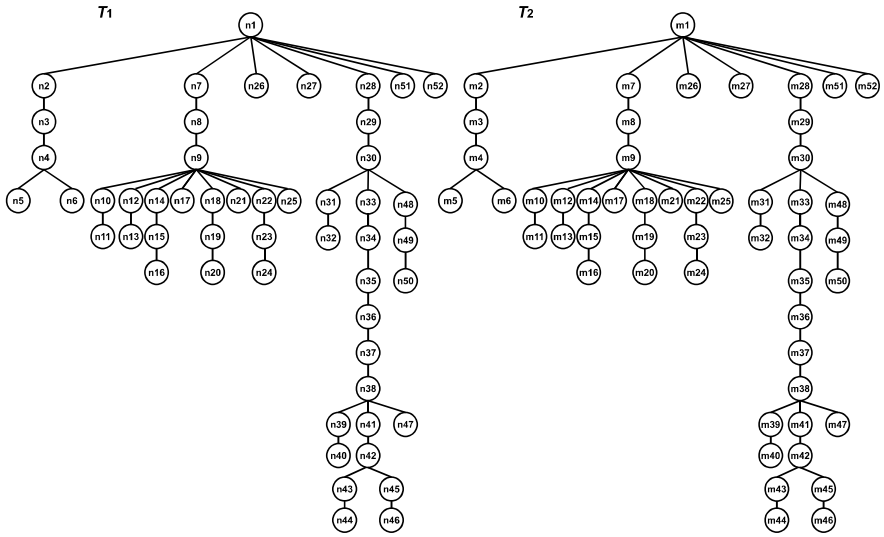


図 4.5: 構文木グラフ T_1 と T_2

表 4.1: 構文木 T の各ノードラベルとプログラムの対応ソースコード 1-1

n1	main	n2	strprintf_a_b
n3	void	n4	strprintf_a_b
n5	printf(”a[]=%s\n”,a)	n6	printf(”b[]=%s\n”,b)
n7	strchange_a_b	n8	void
n9	strchange_a_b	n10	char
n11	Temp[80]	n12	int
n13	i	n14	for
n15	i=0;a[i]!=’\0’;i++	n16	temp[i]=’\0’

た場合は以下のように計算する．二つの構文木 T_1 と T_2 に対し，$S_{TO}(T_1, T_2)$ を求めると，$|P_{TO}(n_1, m_2)|$=$S_{TO}(T_1, T_2)$ を満たす $P_{TO}(n_1, m_2)$ が得られる（図 4.6 の矩形で囲っている部分（Rectangle1）である）．つまり，n_1 と m_2 を重ね合わせたときに，共通部分木が最大になる．

表 4.2: 構文木 T の各ノードラベルとプログラムの対応ソースコード 1-2

n17	temp[i]='\0'	n18	for
n19	i=0;b[i]!=i++	n20	a[i]='\0'
n21	a[i]='\0'	n22	for
n23	i=0;temp[i]!='\0';i++	n24	b[i]=temp[i]
n25	b[i]='\0'	n26	printf("\n chenge \n")
n27	strprintf_a_b	n28	toupper_a_b
n29	void	n30	toupper_a_b
n31	int	n32	i
n33	for	n34	i=0;b[i]!='\0';i++
n35	a[i]=to_upper(a[i])	n36	char
n37	to_upper	n38	char
n39	char	n40	cc
n41	if	n42	(c≥'a')&&(c≤'z')
n43	yes	n44	cc=c-'a'+'A'
n45	no	n46	cc=c
n47	return cc	n48	for
n49	i=0;b[i]!='\0';i++	n50	b[i]=to_upper(b[i])
n51	printf("chenge\n")	n52	strprintf_a_b

ここで，求められた最大共通部分木を親ノードに縮約する．縮約された部分木に対して更新すれば，図 4.7 のような部分木になる．同じように重ね合わせた最大共通部分木を求めると図 4.7 の点線で囲っている部分 rectangle2 が求められる．従って，すべての共通部分木を同じように求めると図 4.8 のように Rec3，Rec4 と Rec5 が求

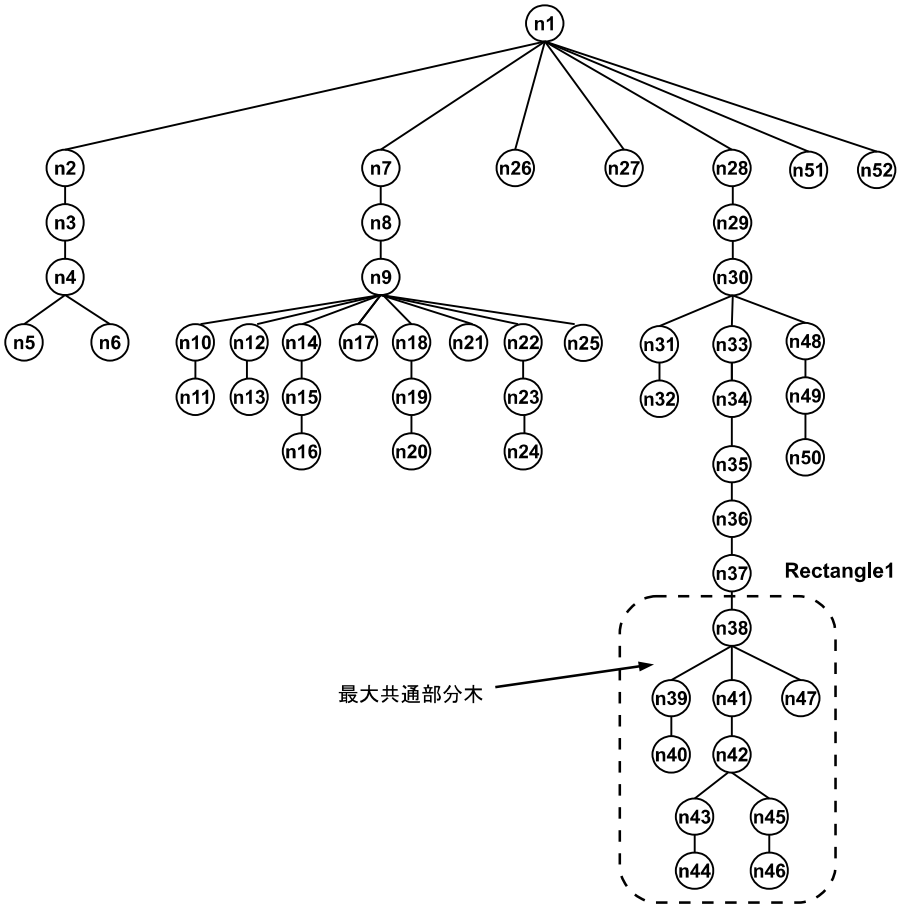


図 4.6: ケース (i) NTO により得られた最大共通部分木 Rectangle1

められる．

T_2 におけるノード集合 N_c は，N_c=$\{m_{10}, m_{11}, m_{12}, m_{13}, m_{14}, m_{15}, m_{16}, m_{18}, m_{19}, m_{20}, m_{22}, m_{23}, m_{24}, m_{31}, m_{32}, m_{33}, m_{34}, m_{48}, m_{49}\}$．図 4.5 の T_1 と T_2 に対する $S_{TO}(T_1, T_2)$ の計算結果，$S_{TO}(T_1, T_2)$=0 であるため，$P_{TO}(n, m)$=ϕ となる．すべての共通部分木を求められ，更新後の部分木が図 4.9 のようになる．

結果として，関数名だけ変換した場合，図 4.5 の二つのプログラムの類似度は

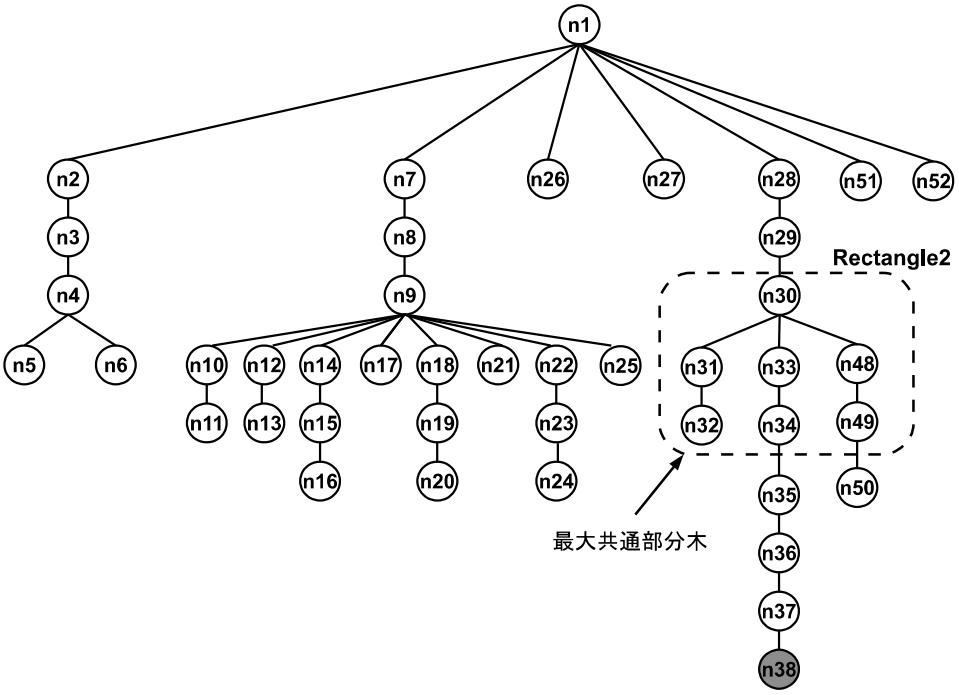


図 4.7: ケース (i) NTO による更新された部分木とその最大共通部分木 Rectangle2

$S(P_1, P_2) = |N_c|/\max\{|N_1|, |N_2|\}$=29/52 = 55.8%

ケース (ii) 変数名だけ変換した場合

まず，NTO 手法を用いて図 4.5 を実験する．T_1 と T_2 に対して，引数を変換した場合以下のように計算を行う．二つの構文木 T_1 と T_2 に対し，$S_{TO}(T_1, T_2)$ を求めると，$|P_{TO}(n_1, m_2)|$=$S_{TO}(T_1, T_2)$ を満たす $P_{TO}(n_1, m_2)$ が得られる．(図 4.10 の矩形で囲っている部分（Rectangle1）である）．つまり，n_1 と m_2 を重ね合わせたときに，共通部分木が最大になる．

求められた最大共通部分木を親ノードに縮約する．縮約された部分木に対して更新すれば，図 4.11 のような部分木になる．同じように重ね合わせた最大共通部分木を求めると図 4.12 の点線で囲っている部分 Rectangle2 が求められる．

T_2 におけるノード集合 N_c は，N_c=$\{m_1, m_2, m_3, m_4, m_5, m_6, m_7, m_8, m_9, m_{26},$

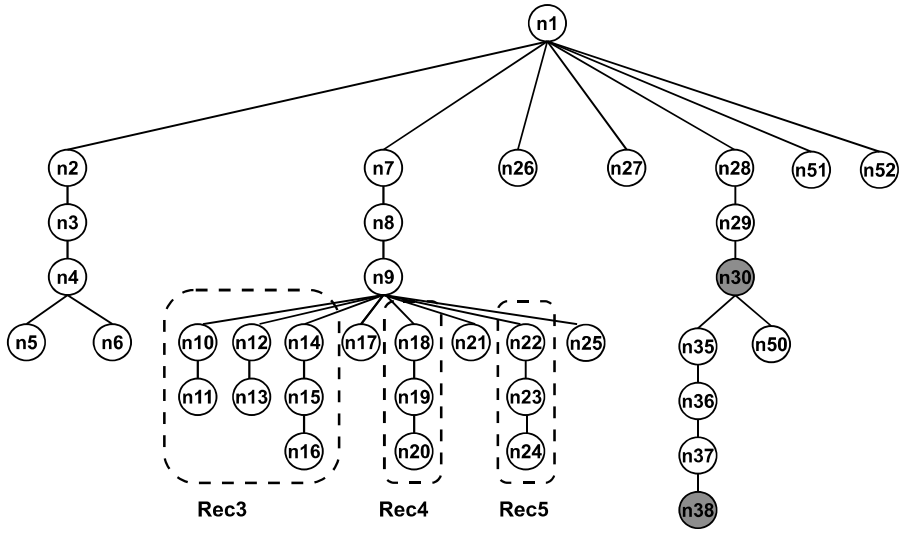


図 4.8: ケース (i) NTO に更新後のすべての共通部分木 Rec3,Rec4,Rec5

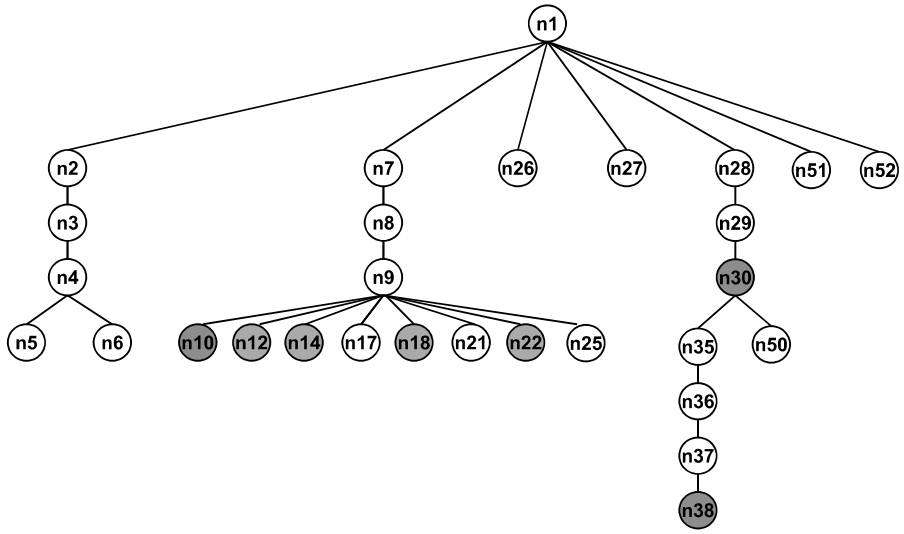


図 4.9: ケース (i) NTO による最終更新された部分木

$m_{27}, m_{28}, m_{29}, m_{30}, m_{51}, m_{52}, m_{38}, m_{39}, m_{40}, m_{41}, m_{42}, m_{43}, m_{44}, m_{45}, m_{46}, m_{47}\}$. 図4.5 の$T_1$と$T_2$に対する$S_{TO}(T_1, T_2)$の計算結果，$S_{TO}(T_1, T_2)$=0であるため，$P_{TO}(n, m)$=$\phi$ となる．すべての共通部分木を求められ，更新後の部分木が図4.13のようになる．それで，図4.5の二つのプログラムの類似度は$S(P_1, P_2) = |N_c|/\max\{|N_1|, |N_2|\}$=26/52 =

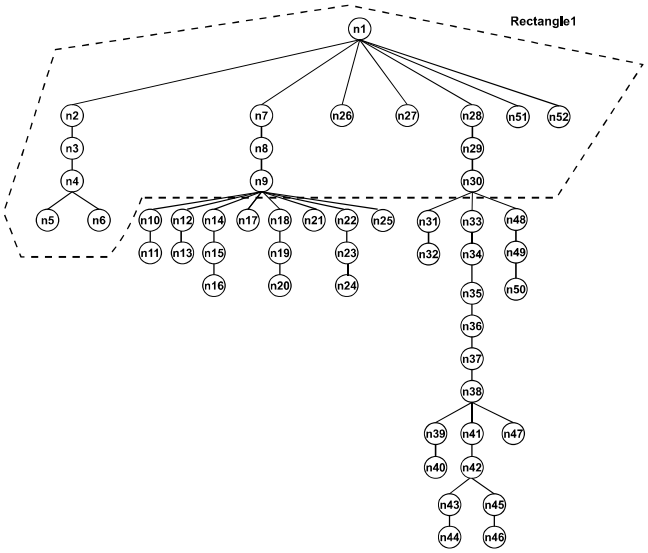


図 4.10: ケース (ii) NTO により得られた最大共通部分木 Rectangle1

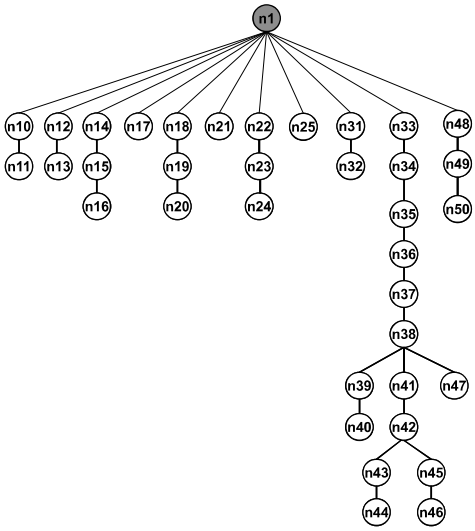


図 4.11: ケース (ii) 更新された部分木

50%.

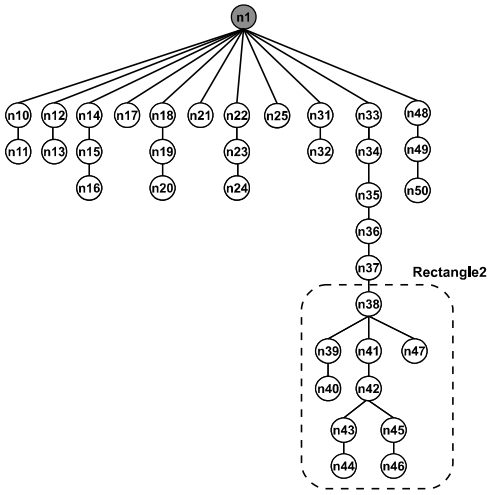


図 4.12: ケース (ii) NTO 手法により更新された部分木とその最大共通部分木 Rectangle2

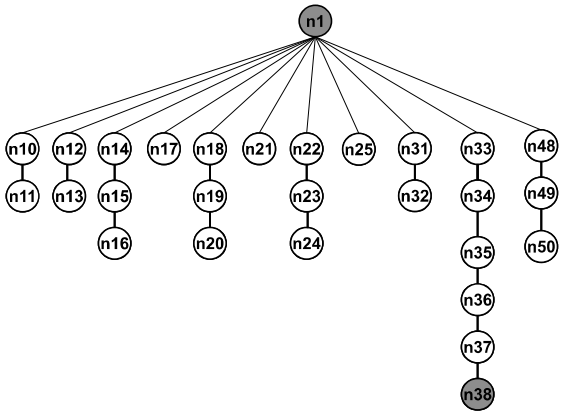


図 4.13: ケース (ii) NTO による最終更新された部分木

ケース (iii) 自作関数と変数名両方変換した場合

NTO 手法を用いて図 4.5 の T_1 と T_2 に対して，関数と引数名両方変換した場合以下のように計算を行う．$|P_{TO}(n_1, m_2)|$=$S_{TO}(T_1, T_2)$ を満たす $P_{TO}(n_1, m_2)$ が得られる．（図 4.14 の矩形で囲っている部分（rectangle1）である）．つまり，n_1 と m_2 を重ね合わせたときに，共通部分木が最大になる．T_2 におけるノード集合 N_c は，

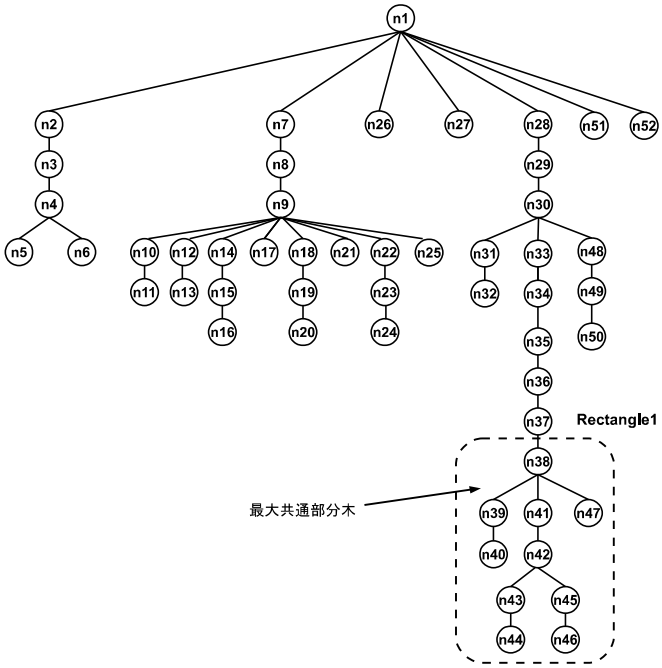


図 4.14: ケース (iii) NTO による最大共通部分木

N_c=$\{m_{38}, m_{39}, m_{40}, m_{41}, m_{42}, m_{43}, m_{44}, m_{45}, m_{46}, m_{47}\}$. 図 4.5 の T_1 と T_2 に対する $S_{TO}(T_1, T_2)$ の計算結果，$S_{TO}(T_1, T_2)$=0 であるため，$P_{TO}(n, m)$=ϕ となる．すべての共通部分木を Rectangle1 しか求められないため，更新後の部分木が図 4.15 のようになる．二つのプログラムの類似度は $|N_2|\}$=10/52 = 19.2% になる．

4.3 Depth Matching（DM）手法の提案

本節では，構文木の類似性判定に用いる類似度算出である Depth Matching[86, 87] 手法を提案する．まず，提案手法について述べた上で，2 つの木を例として類似度計算の具体例を示す．

木グラフ T_1=(N_1, E_1, l_1)，T_2=(N_2, E_2, l_2) を考える．ここで，ルートノードからの距離（深さ・レベル）を k で表す．k^{T_1} は T_1 中の最も深い位置にあるリーフノードの深さである．l_1 と l_2 は，それぞれの木に関するノードのラベル関数であり，$l_1(x)$ は

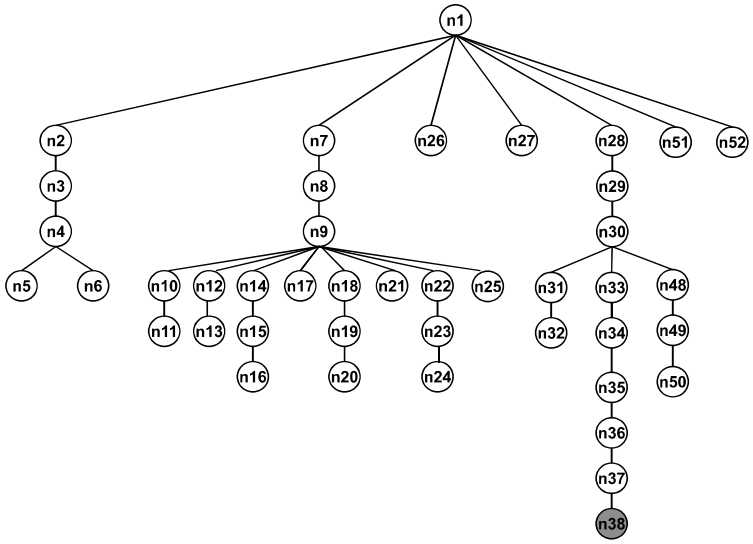


図 4.15: ケース (iii) 更新された木

T_1 のノード x のラベルを表す．以下の手順に沿って，T_1 と T_2 の類似度 $S(T_1, T_2)$ を計算する．

(i) T_1 について，深さが同じノードの集合 N_1^k を作成する（$0 \leq k \leq k^{T_1}$）．同様に N_2^k を作成する．

(ii) $0 \leq k \leq min\{k^{T_1}, k^{T_2}\}$ について (1)〜(2) を繰り返す．

(ii)-(1) N_1^k と N_2^k について，それぞれのノードの順番と位置を記録した上で，2 部グラフ G^k=(V^k, E^k) を作る．ここで，V^k=$N_1^k \cup N_2^k$ であり，$(v_i, u_j) \in E^k$ は $l(v_i)$=$l(u_j)$ を表す．

(ii)-(2) 2 部グラフ G^k のノードの最大マッチング M_k を求める．その際，同じレベルのノードについて，その順番に矛盾が生じないようにする（ノンクロスマッチング）．

(iii) T_1 と T_2 の各レベルでのマッチングの節点数の合計 NM_T を以下の式に基づいて計算する．但し，NM_k は G^k における最大マッチングにおける節点数であり，$NM_k = |M_k| \times 2$ である．

$$NM_T = 2 \times \sum_{k=0}^{K_{min}} |M_k| \tag{4.3}$$

表 4.3: T_1 と T_2 の各ノードのラベル

T_1 のノード	v1	v2	v3	v4	v5	v6	v7	v8	v9
ラベル	a	b	c	z	e	f	g	i	j

T_2 のノード	u1	u2	u3	u4	u5	u6	u7	u8	u9	u10
ラベル	a	x	c	r	e	f	g	y	i	j

(iv) 次式に基づいて $S(T_1, T_2)$ を計算する．

$$S(T_1, T_2) = \frac{NM_T}{|N_1| + |N_2|} \tag{4.4}$$

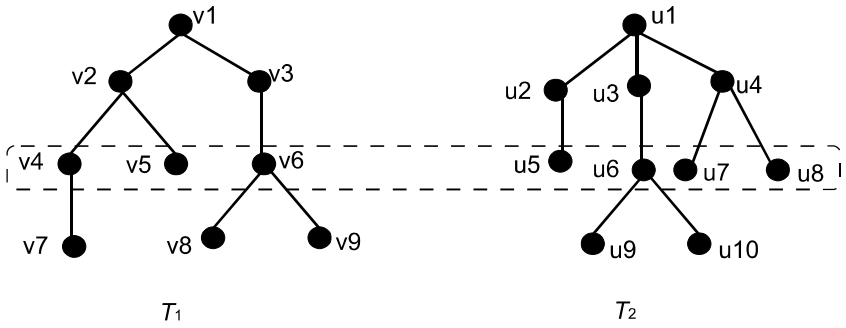


図 4.16: Depth Matching 手法例

図 4.16 に示した T_1 と T_2 を例として，Depth Matching 手法を用いた類似度計算の具体例を示す．なお，T_1 と T_2 の各ノードのラベルは表 4.3 に示したとおりである．この例において，レベル 2（深さ 2）のノードの 2 部グラフ G^2=(V^2, E^2) は，V^2=$\{v4, v5, v6, u5, u6, u7, u8\}$，$E^2$=$\{(v5, u5), (v6, u6)\}$ である．そのため，最大マッチングは M_2=$\{(v5, u5), (v6, u6)\}$ であり，その節点数 NM_2 は 4 となる．

同様に，深さ 0〜4 までの最大マッチングにおける節点数を求めると，それぞれ NM_0=2，NM_1=2，NM_3=4 になる．(iii) により，木グラフ T_1 と T_2 のマッチングに

おける総ノード数 NM_T は 12 になり，(iv) で木間の類似度 $S(T_1, T_2)$ は $12/19 = 0.631$ (63.1%) になる．

DM によるサンプルの計算

ケース (i) 自作関数名だけ変換した場合

DM 手法を用いて図 4.5 を実験する．関数だけ変換した場合，提案手法における，これら 2 つの構文木の類似度の計算過程は以下に示す．まず，2 つの構文木について，図 4.17 に示したように同じレベルの節点から構成される 2 部グラフ $G^0, G^1, ..., G^{13}$ for k=0, 1, ..., 13 を構成する．次に，図 4.17 に示したすべての 2 部グラフ G^k について，最大マッチング M_k $(0 \leq k \leq 13)$ を求める．．ここでレベル $1, 3, 6$ の最大マッチングを示す（図 4.18）．すべての二部グラフ G^k について，最大マッチング M_k $(0 \leq k \leq 13)$ を求められる．

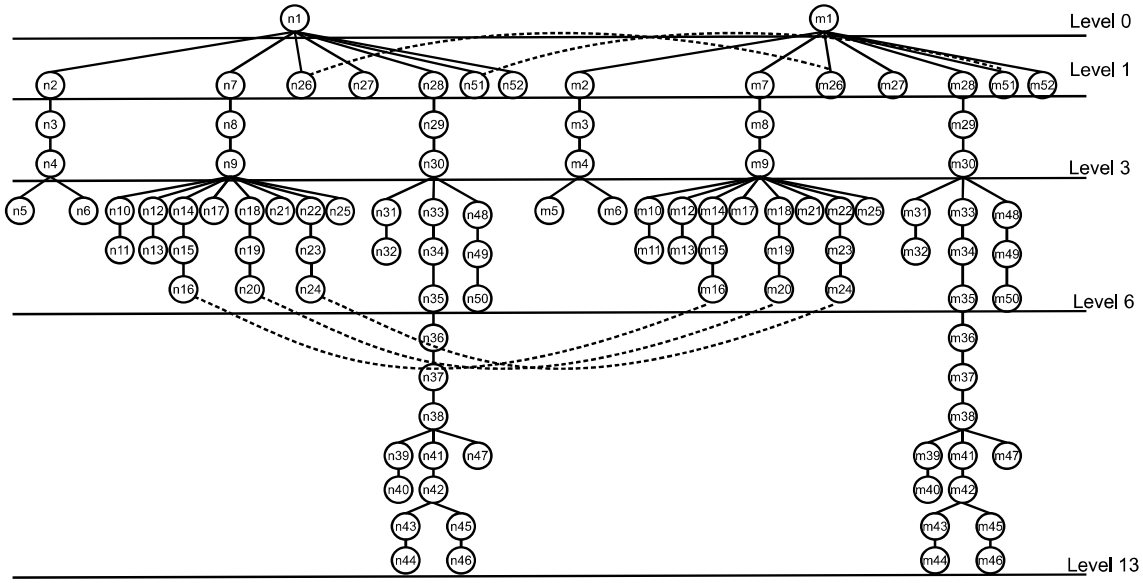


図 4.17: ケース (i) DM 手法によるレベルごとの写像

- 二部グラフの最大ノンクロスマッチングの集合を以下のように示す:

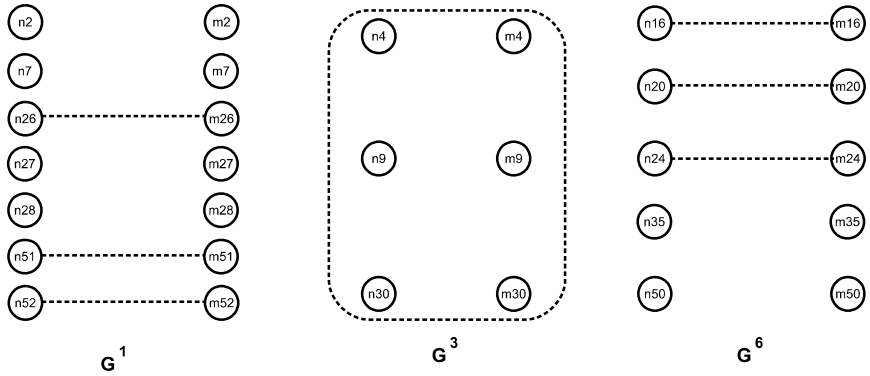


図 4.18: ケース (i) 二部グラフ $G^1G^3G^6$ の最大ノンクロスマッチング $M_1M_3M_6$

M_0=$\{(n1, m1)\}$,
M_1=$\{(n26, m26), (n51, m51), (n52, m52)\}$,
M_2=$\{(n3, m3), (n8, m8), (n29, m29)\}$,
M_3=ϕ,
M_4=$\{(n5, m5), (n6, m6), (n10, m10), (n12, m12), (n14, m14), (n17, m17), (n18, m18),$
$(n21, m21), (n22, m22), (n25, m25), (n31, m31), (n33, m33), (n48, m48)\}$,
M_5=$\{(n11, m11), (n13, m13), (n15, m15), (n19, m19), (n23, m23), (n32, m32),$
$(n34, m34), (n49, m49)\}$,
M_6=$\{(n16, m16), (n20, m20), (n24, m24)\}$,
M_7=$\{(n36, m36)\}$,
M_8=ϕ,
M_9=$\{(n38, m38)\}$,
M_{10}=$\{(n39, m39), (n41, m41), (n47, m47)\}$,
M_{11}=$\{(n40, m40), (n42, m42)\}$,
M_{12}=$\{(n43, m43), (n45, m45)\}$,
M_{13}=$\{(n44, m44), (n46, m46)\}$.

前節 (iii) より, 各レベルでのマッチングの節点数の合計 NM_T は 42 となる.2 つの構文木の節点数 $|N_1|$ と $|N_2|$ は両方とも 52 であり,2 つの構文木, すなわち, プログラムの類似度は $S(T_1, T_2)$ は,(iv) より 42/52=80.8%となる.

ケース (ii) 変数名だけ変換した場合

DM 手法を用いて図 4.5 を実験する．関数だけ変換した場合，DM 手法における, これら 2 つの構文木の類似度の計算過程は以下に示す. まず，2 つの構文木について, 図 4.19 に示したように同じレベルの節点から構成される 2 部グラフ $G^0, G^1, ..., G^{13}$ for k=0, 1, ..., 13 を構成する. 次に, 図 4.19 に示したすべての 2 部グラフ G^k について, 最大マッチング M_k $(0 \leq k \leq 13)$ を求める．ここでレベル $1, 3, 6$ の最大マッチングを示す（図 4.20）．すべての二部グラフ G^k について，最大マッチング M_k $(0 \leq k \leq 13)$ を求められる．

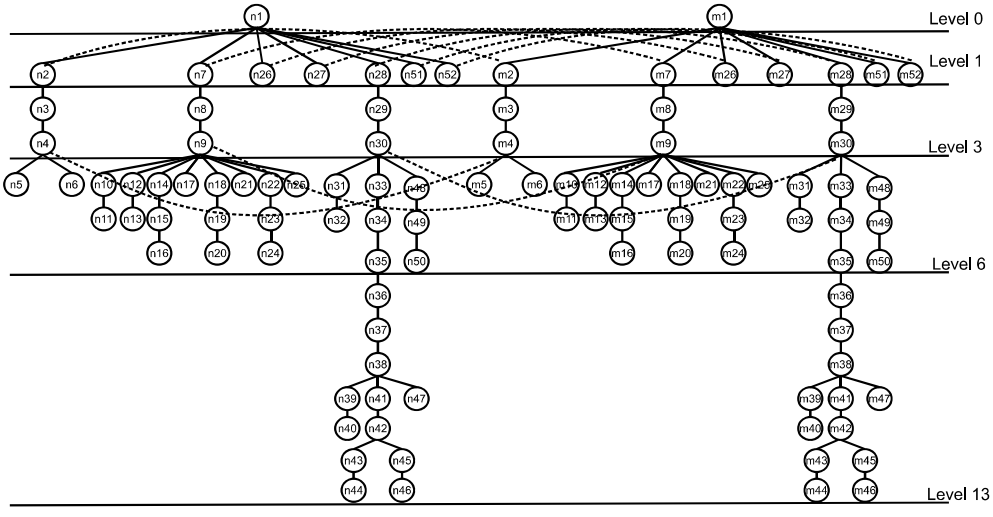


図 4.19: ケース (ii) DM 手法によるレベルごとの写像

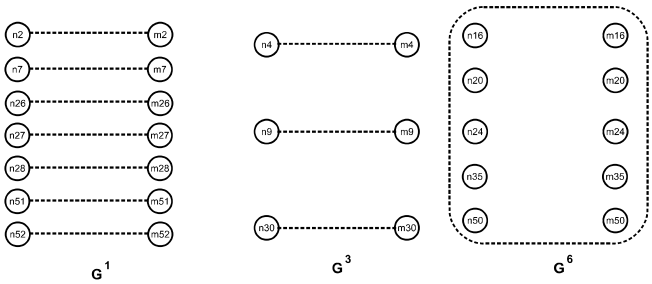


図 4.20: ケース (ii) 二部グラフ $G^1G^3G^6$ の最大ノンクロスマッチング $M_1M_3M_6$

- 二部グラフの最大ノンクロスマッチングの集合を以下のように示す:

M_0=$\{(n1, m1)\}$,
M_1=$\{(n2, m2), (n7, m7), (n26, m26), (n27, m27), (n28, m28), (n51, m51), (n52, m52)\}$,
M_2=$\{(n3, m3), (n8, m8), (n29, m29)\}$,
M_3=$\{(n4, m4), (n9, m9), (n30, m30)\}$,
M_4=$\{(n5, m5), (n6, m6), (n10, m10), (n12, m12), (n14, m14), (n18, m18), (n22, m22)\}$,
M_5=$\{(n11, m11)\}$,
M_6=ϕ,
M_7=$\{(n36, m36)\}$,
M_8=$\{(n37, m37)\}$,
M_9=$\{(n38, m38)\}$,
M_{10}=$\{(n39, m39), (n41, m41), (n47, m47)\}$,
M_{11}=$\{(n40, m40), (n42, m42)\}$,
M_{12}=$\{(n43, m43), (n45, m45)\}$,
M_{13}=$\{(n44, m44), (n46, m46)\}$.

前節 (iii) より, 各レベルでのマッチングの節点数の合計 NM_T は 34 となる.2 つの構文木の節点数 $|N_1|$ と $|N_2|$ は両方とも 52 であり,2 つの構文木, すなわちプログラム

の類似度は $S(T_1, T_2)$ は,(iv) より 34/52=65.3%となる.

ケース (iii) 自作関数と変数名両方変換した場合

DM 手法を用いて図 4.5 の T_1 と T_2 に対して，関数と引数名両方変換した場合以下のように計算を行う．DM 手法における, これら 2 つの構文木の類似度の計算過程は以下に示す. まず，2 つの構文木について, 図 4.21 に示したように同じレベルの節点から構成される 2 部グラフ $G^0, G^1, ..., G^{13}$ for k=0, 1, ..., 13 を構成する．次に, 図 4.21 に示したすべての 2 部グラフ G^k について, 最大マッチング M_k $(0 \leq k \leq 13)$ を求める．すべての二部グラフ G^k について，最大マッチング M_k $(0 \leq k \leq 13)$ を求められる．

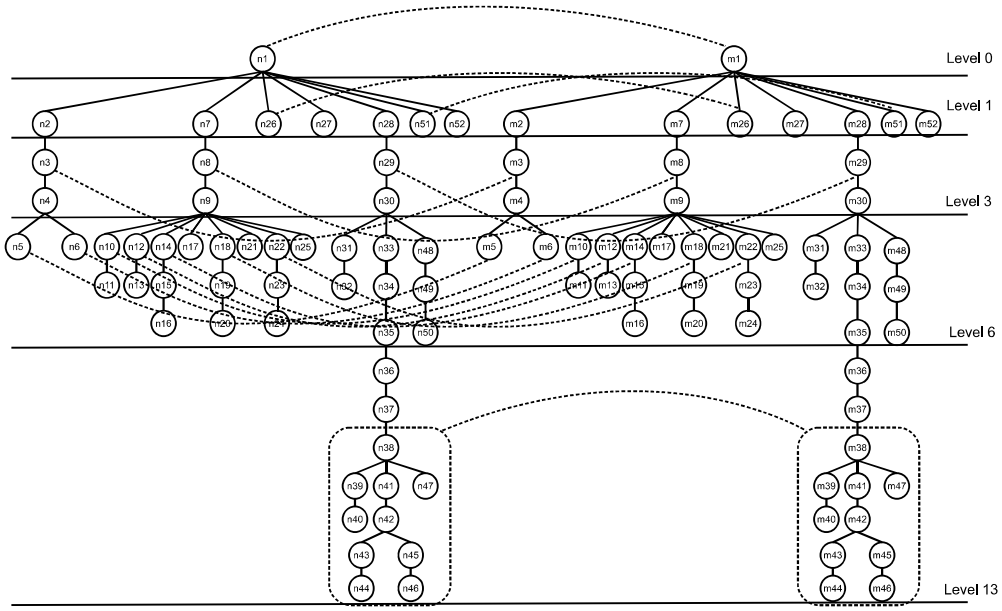


図 4.21: ケース (iii) DM 手法によるレベルごとの写像

- 二部グラフの最大ノンクロスマッチングの集合を以下のように示す:

M_0={$(n1, m1)$},

M_1={$(n26, m26), (n51, m51)$},
M_2={$(n3, m3), (n8, m8), (n29, m29)$},
M_3=ϕ,
M_4={$(n5, m5), (n6, m6), (n10, m10), (n12, m12), (n14, m14), (n18, m18), (n22, m22),$
$(n25, m25), (n31, m31), (n33, m33), (n48, m48)$},
M_5={$(n11, m11)$},
M_6=ϕ,
M_7={$(n36, m36)$},
M_8=ϕ,
M_9={$(n38, m38)$},
M_{10}={$(n39, m39), (n41, m41), (n47, m47)$},
M_{11}={$(n40, m40), (n42, m42)$},
M_{12}={$(n43, m43), (n45, m45)$},
M_{13}={$(n44, m44), (n46, m46)$}.

自作関数と引数両方とも変換した場合はマッチングされるノードは図 4.21 のようになる．前節 (iii) より, 各レベルでのマッチングの節点数の合計 NM_T は 29 となる．2 つの構文木の節点数 $|N_1|$ と $|N_2|$ は両方とも 52 で 2 つのプログラムの類似度は $S(T_1, T_2)$ は,(iv) より 29/52=55.8%となる.

以上では，図 4.2 の C 言語オリジナルプログラム P に対して，自作関数や引数の名前などを書き換えたプログラムを作成し，それぞれ対応する構文木に変換した．変換された二つの構文木に対して，提案した NTO と DM 手法を用いて三つのケースでそれぞれ類似度の計算を行った．計算した結果：NTO 手法では，ケース (i) 自作関数のみすべて変換した場合の類似度は 55.8%であり，ケース (ii) すべての引数名を変換した場合の類似度は 50%であり，ケース (iii) 自作関数と引数名を両方変換した場合の類似度は 19.2%である．DM 手法では，ケース (i) 自作関数のみすべて変換した場合の類似度は 80.8%であり，ケース (ii) すべての引数名を変換した場合の類似度は 65.3%であり，ケース (iii) 自作関数と引数名を両方変換した場合の類似度は 55.8%である．各ケースでの計算結果を比較すると DM 手法が NTO 手法より類

似度が高かった．書き換えたプログラムとオリジナルプログラム同士は類似度が高くなるべきであり，より高い類似度が求まる事が望ましい．

第 5 章

シミュレーション結果と考察

本章で，提案した手法と既存の SMMT 手法を用いて実際の学生レポートやテキストプログラムや研究用プログラムなどでシミュレーションを行い，実験結果について考察と評価し，提案手法の有効性を確認する．

5.1 各手法によるシミュレーションと評価

ここでは，前章で紹介した NTO，DM と既存の SMMT 手法を評価するためのシミュレーション実験について述べる．

表 5.1: サンプルプログラム 1-1

No.	行数	関数		引数	
		関数名変換	変換箇所数	引数名変換	変換箇所数
1	56	2	6	1	3
2	103	4	10	1	15
3	93	1	2	2	12
4	76	1	2	1	9
5	176	2	4	2	17
6	45	1	2	2	8
7	40	2	4	1	9

表 5.2: サンプルプログラム 1-2

No.	行数	関数		引数	
		関数名変換	変換箇所数	引数名変換	変換箇所数
8	289	6	16	1	42
9	75	2	4	1	14
10	39	1	3	2	16
11	30	1	2	1	12
12	27	1	2	1	4
13	32	1	2	1	3
14	41	2	5	2	9
15	162	4	8	1	65
16	950	6	21	2	14
17	1439	7	16	7	154
18	450	4	9	2	28
19	1209	12	32	5	130
20	1566	10	26	7	74

今回のシミュレーションは提案した手法の有効性を確認するために,『C 言語スタートブック改訂第 3 版』の課題問題と『アルゴリズム論』授業の課題問題と実用研究の大きなプログラムを用いて 20 個のプログラムに対して実験を行った．実験では以上の変換ケースで変換したプログラムとオリジナルプログラムとの比較を行った．

表 5.1,5.2 で示すように “行数” はサンプルプログラムの長さであり，それぞれ “関数名” とその関数のプログラムに現れる “箇所数” と “引数名” とその引数のプログラムに現れる箇所数 “変換箇所数” である． 我々の提案した NTO と DM と既存の SMMT 手法を用いてシミュレーションを実験した結果は表 5.3,5.4 に示す．

表 5.3: NTO と DM と SMMT 手法による計算された類似度 1-1

Case (i) 関数名のみ変換			Case (ii) 引数名のみ変換			Case (iii) 両方変換		
NTO	DM	SMMT	NTO	DM	SMMT	NTO	DM	SMMT
55.5%	74.1%	60.0%	81.4%	88.8%	80.0%	48.1%	66.6%	46.6%
48.3%	74.0%	70.2%	64.1%	71.6%	58.3%	30.2%	56.6%	38.8%
89.4%	95.3%	87.2%	60.9%	81.5%	58.2%	54.2%	79.4%	50.1%
90.6%	96.8%	95.1%	68.7%	87.5%	80.4%	57.8%	84.3%	75.6%
92.6%	94.2%	94.7%	72.1%	78.8%	85.5%	61.3%	71.9%	72.3%
88.8%	89.4%	92.3%	72.2%	73.6%	80.7%	66.6%	68.4%	76.9%
31.8%	84.0%	81.8%	36.5%	72.7%	72.7%	22.7%	61.3%	50.0%
68.5%	87.3%	55.4%	78.3%	88.2%	75.2%	46.3%	75.3%	50.4%
48.4%	70.9%	74.2%	83.4%	90.3%	74.2%	25.8%	61.2%	54.2%
62.3%	85.4%	58.2%	42.2%	67.5%	52.2%	36.6%	55.5%	50.6%
94.2%	96.0%	92.5%	80.0%	90.3%	77.7%	77.1%	82.8%	74.0%
55.4%	80.1%	84.6%	35.0%	90.0%	84.6%	40.3%	70.0%	45.2%
75.0%	89.5%	88.2%	81.2%	88.2%	82.3%	68.7%	75.0%	76.4%
40.7%	66.6%	79.1%	59.2%	85.5%	54.1%	29.2%	58.8%	58.3%
75.0%	94.1%	91.3%	66.1%	76.4%	65.5%	35.2%	70.5%	60.2%

実験結果からプログラム間の平均類似度は以下のように求められた:
(1) ケース (i) 関数名だけ変換した場合：72.4%, 87.9%，84.4%.
(2) ケース (ii) 引数名だけ変換した場合：69.4%, 84.5%，76.7%.
(3) ケース (iii) 関数名と引数名両方変換した場合：53.8%, 74.5%，65.9%.

ここでさらに，サンプルから大規模なプログラムを五つ抽出する．これらを以下の五つのケースでそれぞれ変換作業を行い，変換後のプログラムとオリジナルプログラム間の類似度を提案した NTO 手法と DM 手法と既存の SMMT 手法で実験をす

表 5.4: NTO と DM と SMMT 手法による計算された類似度 1-2

Case (i) 関数名のみ変換			Case (ii) 引数名のみ変換			Case (iii) 両方変換		
NTO	DM	SMMT	NTO	DM	SMMT	NTO	DM	SMMT
91.7%	96.2%	94.8%	94.0%	96.4%	94.4%	85.4%	92.3%	91.1%
86.1%	98.1%	97.9%	70.2%	89.6%	88.6%	64.3%	87.6%	85.1%
87.0%	97.2%	96.4%	72.2%	88.0%	82.2%	70.3%	87.1%	80.1%
85.7%	92.0%	96.6%	93.5%	96.7%	95.4%	82.0%	93.6%	92.0%
80.4%	97.2%	96.5%	76.1%	88.8%	92.7%	73.8%	92.2%	89.3%
Average Value								
72.4%	87.9%	84.4%	69.4%	84.5%	76.7%	53.8%	74.5%	65.9%

る．提案した NTO と DM 手法と既存の SMMT 手法を用いて計算した結果としてそれぞれの平均値を以下のように求められた：

ケース (i) 関数名だけ変更した場合：86.2%，96.1%，95.4.%.

ケース (ii) グローバル変数名が変更した場合：75.2%，90.6%，87.5%.

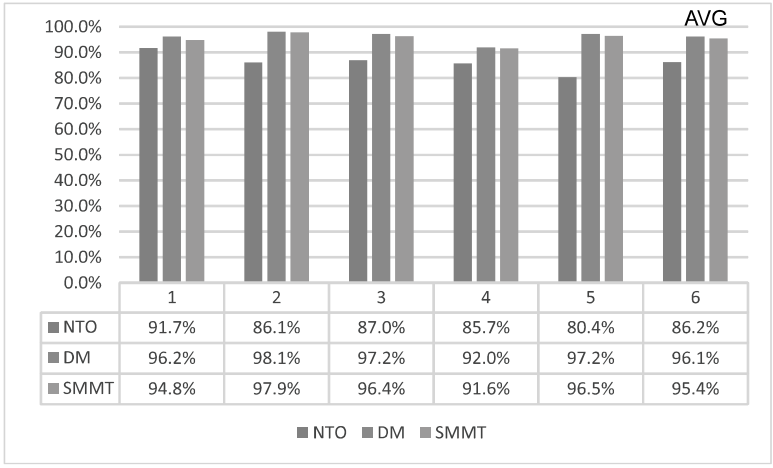
ケース (iii) 自作関数名とグローバル変数名が変更した場合：81.2%，91.9%，90.7%.

ケース (iv) ローカル変数名を変更した場合：81.0%，90.6%，87.4%.

ケース (v) ローカルとグローバル変数両方変更した場合：68.5%，84.7%，78.5%.

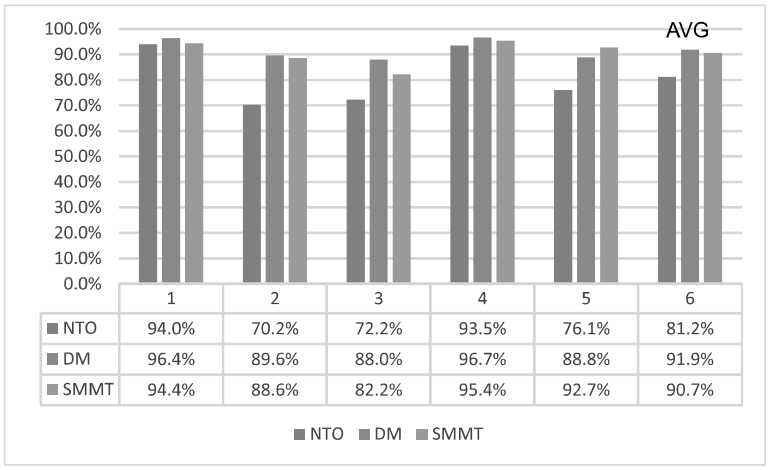
図 5.1～図 5.5 は五つのケースでシミュレーション実験を行った結果を示す．実験結果から DM 手法はもっとも高い類似度を得られたことがわかる．5 つのケースで変換されたプログラムに対して，DM 手法は高い類似度を得られた．ここで，SMMT 手法も DM 手法に近いよい結果がえられた．しかし，SMMT 手法はプログラムの行ごとを用いて比較しているが，プログラムの 1 行を複数回改行したり，関数内部の複数の行を 1 行に書き換えたりした場合，類似度が引くなる場合は類似度の誤判別の場合がよく存在する．つまり，正しくなくなる．NTO 手法で得られた結果も DM と SMMT 手法の結果に近似だった．

ここで，我々は提案した二つの NTO 手法と DM 手法と既存の SMMT 手法を用



	1	2	3	4	5	6
NTO	91.7%	86.1%	87.0%	85.7%	80.4%	86.2%
DM	96.2%	98.1%	97.2%	92.0%	97.2%	96.1%
SMMT	94.8%	97.9%	96.4%	91.6%	96.5%	95.4%

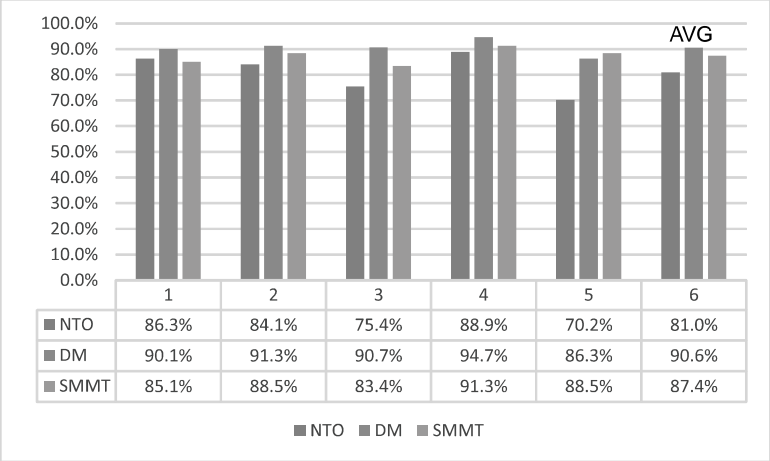
図 5.1: ケース (i) 関数名だけ変更した場合の類似度結果



	1	2	3	4	5	6
NTO	94.0%	70.2%	72.2%	93.5%	76.1%	81.2%
DM	96.4%	89.6%	88.0%	96.7%	88.8%	91.9%
SMMT	94.4%	88.6%	82.2%	95.4%	92.7%	90.7%

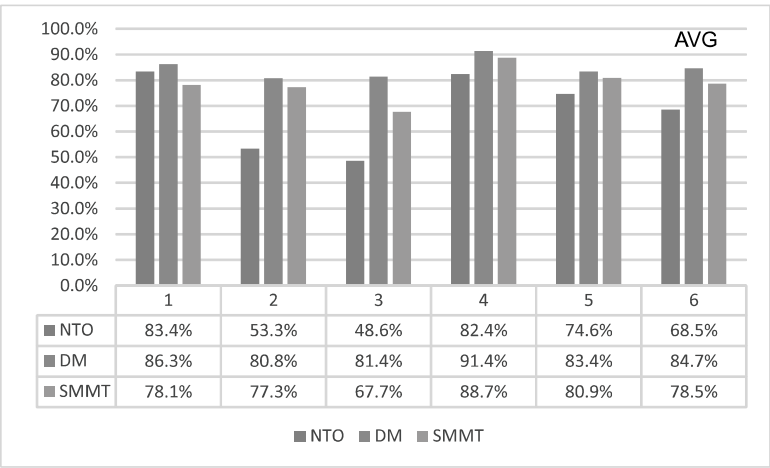
図 5.2: ケース (ii) グローバル変数が変更した場合の類似度結果

いて，山口大学情報系の C 言語受講生が提出したレポートに対して実験を行った．実験結果はぞれぞれ表 5.5〜5.10 に示す．結果からわかるように小規模な課題に対して，個人の処理手順（アルゴリズム）と書き方により学生のレポート・プログラム同士の類似度が低かった．実験で高い類似度のプログラム同士はやはり構造と記述文はほぼ同じである事を目視により確認できた．



	1	2	3	4	5	6
NTO	86.3%	84.1%	75.4%	88.9%	70.2%	81.0%
DM	90.1%	91.3%	90.7%	94.7%	86.3%	90.6%
SMMT	85.1%	88.5%	83.4%	91.3%	88.5%	87.4%

図 5.3: ケース (iii) 自作関数名とグローバル変数が変更した場合の類似度結果

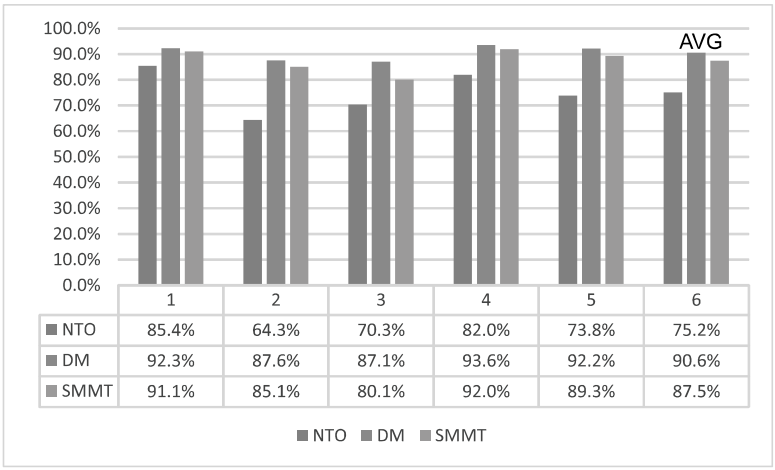


	1	2	3	4	5	6
NTO	83.4%	53.3%	48.6%	82.4%	74.6%	68.5%
DM	86.3%	80.8%	81.4%	91.4%	83.4%	84.7%
SMMT	78.1%	77.3%	67.7%	88.7%	80.9%	78.5%

図 5.4: ケース (iv) ローカル変数を変更した場合

5.2 考察

前節の実験では，提案した NTO 手法と DM 手法と既存の SMMT 手法を用いて多くの実験プログラムと実際の学生レポートデータを用いて実験を行った．それぞれのケースで実験した結果の比較では，三つの手法に対して高い順に DM,SMMT,NTO



	1	2	3	4	5	6
NTO	85.4%	64.3%	70.3%	82.0%	73.8%	75.2%
DM	92.3%	87.6%	87.1%	93.6%	92.2%	90.6%
SMMT	91.1%	85.1%	80.1%	92.0%	89.3%	87.5%

図 5.5: ケース (v) ローカルとグローバル変数両方変更した場合の類似度結果

表 5.5: NTO 手法による C 言語コース i のレポート間類似度結果

	No.001	No.002	No.003	No.004	No.005	No.006
No.001	100%	53.4%	32.8%	44.2%	54.4%	51.2%
No.002		100%	35.0%	58.6%	47.7%	69.9%
No.003			100%	23.9%	30.5%	30.0%
No.004				100%	63.4%	68.2%
No.005					100%	55.2%
No.006						100%

となる．既存の SMMT 手法は，Linux の diff を中心に使っており，プログラムの構造が一切反映されず，行ごとに一致部分のマッチング集合が求めらている．その中で，我々は可能な限り高い類似度を求めるため，diff の多数のオプションを使って計算実験をした．提案した NTO 手法では，与えられた二つの構文木に対して，最大共通部分木だけではなく，すべての共通部分木を求め，求められたすべての部分木集合のノードの数に基づいて類似度の計算をしている．しかし，最大部分木を求める際に，親子ペアの形を比較して計算を行うため，形もノード記述文（プログラムソースコード）も同じ場合のみマッチングパターンを取る．そのため，一つのノー

表 5.6: DM 手法による C 言語コース i のレポート間類似度結果

	No.001	No.002	No.003	No.004	No.005	No.006
No.001	100%	67.9%	58.9%	60.8%	65.6%	66.6%
No.002		100%	61.2%	70.2%	70.1%	80.4%
No.003			100%	48.5%	60.4%	59.3%
No.004				100%	73.3%	75.6%
No.005					100%	74.6%
No.006						100%

表 5.7: SMMT 手法による C 言語コース i のレポート間類似度結果

	No.001	No.002	No.003	No.004	No.005	No.006
No.001	100%	35.7%	40.9%	40.8%	50.0%	32.0%
No.002		100%	41.0%	48.1%	39.8%	70.5%
No.003			100%	30.1%	43.9%	33.9%
No.004				100%	64.4%	47.3%
No.005					100%	49.1%
No.006						100%

ド或いは形が異なっていれば全体の親子パターンのマッチングを取れない．このような厳しい条件での判定をしているため，NTO 手法では低い類似度が得られた．言い換えれば，NTO 手法で高い類似度を出した場合は二つのプログラムが同じものであるといえる．DM 手法では，構文木のレベルごとのノードを求め，同じレベルのノード間で二部グラフとして作成し，二部グラフの最大ノンクロスマッチングを取り，得られたノードの数に基づいて類似度の計算を行っている．ここで，同じレベルのノードに対して，条件なし最大ノンクロスマッチングを取るため，高い類似度の値を出している．

表 5.8: NTO 手法による C 言語コース ii のレポート間類似度結果

	No.001	No.002	No.003	No.004	No.005	No.006	No.007	No.008	No.009	No.0010
No.001	100%	31.6%	23.1%	30.2%	26.2%	29.4%	38.5%	30.1%	29.8%	26.2%
No.002		100%	36.5%	55.8%	45.8%	44.4%	46.1%	54.5%	45.9%	42.3%
No.003			100%	54.3%	50.0%	51.6%	34.7%	45.6%	51.9%	32.2%
No.004				100%	80.0%	76.2%	57.7%	80.8%	94.6%	63.1%
No.005					100%	74.6%	53.8%	72.8%	80.9%	53.1%
No.006						100%	46.1%	66.1%	70.2%	74.7%
No.007							100%	54.4%	56.5%	46.3%
No.008								100%	87.8%	59.8%
No.009									100%	65.8%
No.010										100%

表 5.9: DM 手法による C 言語コース ii のレポート間類似度結果

	No.001	No.002	No.003	No.004	No.005	No.006	No.007	No.008	No.009	No.0010
No.001	100%	50.3%	55.3%	58.9%	57.0%	58.8%	57.7%	58.0%	58.0%	49.6%
No.002		100%	50.0%	64.3%	64.6%	59.5%	58.5%	61.0%	64.1%	53.0%
No.003			100%	71.3%	66.1%	67.4%	57.7%	64.7%	68.7%	55.0%
No.004				100%	89.2%	84.9%	73.8%	88.2%	96.2%	75.2%
No.005					100%	83.3%	70.7%	83.0%	90.0%	69.1%
No.006						100%	66.1%	78.6%	83.2%	67.1%
No.007							100%	69.8%	70.9%	59.7%
No.008								100%	93.1%	75.1%
No.009									100%	76.5%
No.010										100%

表 5.10: SMMT 手法による C 言語コース ii のレポート間類似度結果

	No.001	No.002	No.003	No.004	No.005	No.006	No.007	No.008	No.009	No.0010
No.001	100%	39.6%	33.4%	40.8%	36.5%	29.4%	44.2%	41.1%	33.1%	29.8%
No.002		100%	41.2%	53.1%	49.7%	50.2%	48.2%	55.3%	51.2%	46.8%
No.003			100%	60.2%	52.2%	58.9%	34.7%	48.2%	60.9%	38.2%
No.004				100%	82.4%	79.3%	59.1%	83.3%	94.8%	65.2%
No.005					100%	76.3%	58.2%	78.6%	82.3%	50.6%
No.006						100%	50.3%	69.1%	75.3%	78.4%
No.007							100%	58.7%	59.6%	45.2%
No.008								100%	87.8%	60.1%
No.009									100%	69.2%
No.010										100%

第 6 章

おわりに

6.1 研究の成果

本論文では，プログラミング教育現場でC言語類似プログラム同士の類似度を求めるために，C言語プログラムを構文木化した構文木グラフのプログラムのコードと構造に基づいて評価を行う手法である，NTOとDM手法を提案した．

本研究では，我々の提案した(1)構文木化手法はC言語文法$Yacc/lex$による構文木解析ではなく，C言語プログラムを簡潔な木構造に，わかりやすく直感的な構文木に構築することができた．プログラムの関数の呼び出し関係を木グラフの形で表現し，関数内部をさらに構成部品を用いて木グラフを構成した．これにより，C言語プログラム全体を一つの構文木として表現することができた．

プログラミング教育を含む情報教育の現場においては，サンプルプログラムの提示による教育がよく行われているが，学習者にとっては，提示されたプログラムの構成や実行の流れを把握しづらいことがプログラミング学習のハードルを高くしている原因の1つと考えられる．そこで，3章の図3.8にプログラムを図的に表現して提示することで，学習者のサンプルプログラムに対する理解を助け，その教育効果を高めることにつながる．加えて，前述のように，教員にとっては学習者への適正な評価に繋がると考えられる．したがって，本研究でのC言語プログラムの類似性を評価するために提案した手法および開発したソフトウェアは，C言語プログラミングの教育現場やプログラムの構成・構造を把握する必要のある大規模なソフトウェアシステムの設計・改修にも寄与するものと思われる．

次に，変換された構文木を利用したC言語プログラムの類似度の計算結果を述べ

る．構文木に変換して比較することで，計算コストを下げるだけではなく，(2) 木の特性を生かした比較ができる．これにより，文字列比較ではできなかったプログラム構造の比較ができた．

5 章では我々の用意したそれぞれのサンプルプログラム（学生の宿題プログラムと研究用大規模プログラム）に対して，三つのケースとさらに五つのケースを抽出して提案した NTO 手法と DM 手法と既存の SMMT 手法で実験を行った．実験結果として，ケース (i) 72.4%, 87.9% and 84.4% ケース (ii) 69.4%, 84.5% and 76.7% ケース (iii) 53.8%, 74.5% and 65.9% となる．

本論文では C 言語プログラム同士の類似性判定手法として，書き換えたプログラムとオリジナルプログラム同士は類似度が高くなるべきであり，より高い類似度が求まる事が望ましい．DM 手法はよりよい妥当な手法であることを確認した．そして，NTO 手法で得られた結果は DM と SMMT 手法より低かったが，NTO 手法の中核である共通部分木を求めるアルゴリズムの中で，親子ペアの厳しい前提条件のところさらに改善する必要がある．

6.2 本研究の展望

近年,「ソフトウェアが様々な場所や用途で利用されている現在の社会において，ソフトウェアの担う役割は非常に大きくなっている．特に，社会基盤や大企業の基幹業務を担うような大規模ソフトウェアを高品質に開発，保守することは非常に重要になっており，そのようなソフトウェアが障害を起こした場合，国民や企業の経済的，身体的損失など社会的に大きな影響がでる．また，社会のニーズや制度の変化に対応するため，ソフトウェア開発および保守には高品質だけでなく，高生産性が求められるようになっている．そこで，ソフトウェア工学の分野でも，ソフトウェア開発と保守の品質と生産性を高めるための支援が重要となっている．加えて，ソフトウェアシステムは年々大規模化・複雑化しており，ソフトウェアの構築や保守にかかるコストが高まった事で，より多くの労力が必要となってきている．そのため，プログラム理解支援システムやソフト間のバージョン管理やコードの再利用などが求められている」[88].

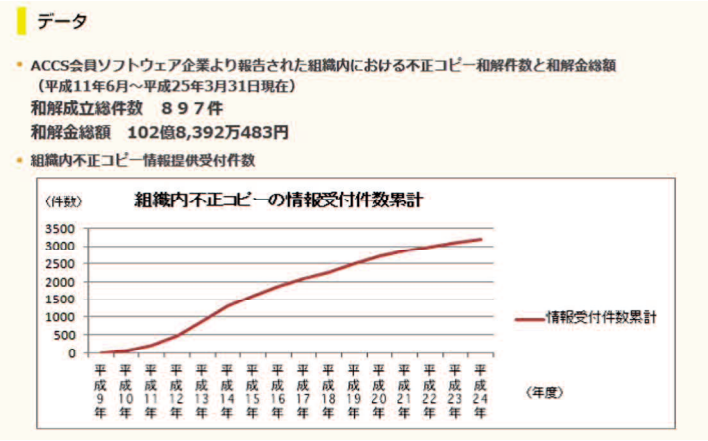


図 6.1: ACCS 会員ソフトウエア企業により報告 [89]

また，各種の教育支援システムとして，類似性に基づくレポート剽窃の検出ツールが開発されている．検出ツールは教師の立場から不正行為の発見だけでなく，学

生の不正抑止手段として利用されている．特に，大学をはじめ研究機構では科学・学術研究を行う際の倫理基準の遵守を徹底し，基準に反した不正行為「剽窃・盗用・捏造など」への厳正対応も必用不可欠であり，個人と企業に対して，知的財産にあたる独自のプログラミング法を保護しなければならない．

著作権侵害は，学校や職場で頻繁に起こる問題である (図 6.1)．学校では，プログラミング教育現場だけではなく文学や経済や生命科学分野などすべてに関わる事である．レポートや卒論などは大学での勉強の成果の証となる重要なものであり，学生が自ら作成すべきものである．そこで，無断コピーや剽窃を行った場合，不正行為と判断されて処罰の対象になる．更に, 大学院生と研究者が独自研究や論文の剽窃と盗用はより厳しい処罰の対象になる．また職場では，他社の中核技術が最も価値のある技術であり，その技術の盗用は法律違反になるとともに企業に対して甚大な経済損失になる．

The Software Alliance（BSA — ザ・ソフトウェア・アライアンス）はグローバル市場において世界のソフトウェア産業を牽引する業界団体である．この組織は世界各国の政府との意見交換，著作権をはじめとする知的財産権の保護ならびに教育啓発活動を通じて，BSA はデジタル社会の拡大とそれを推進する新たなテクノロジーへの信頼の構築に努めている [91]．ここで，BSA による世界ソフトウエア違法コピー調査 2011 による結果 (日本) 図 6.2 と (地域別) 図 6.3 を示す．

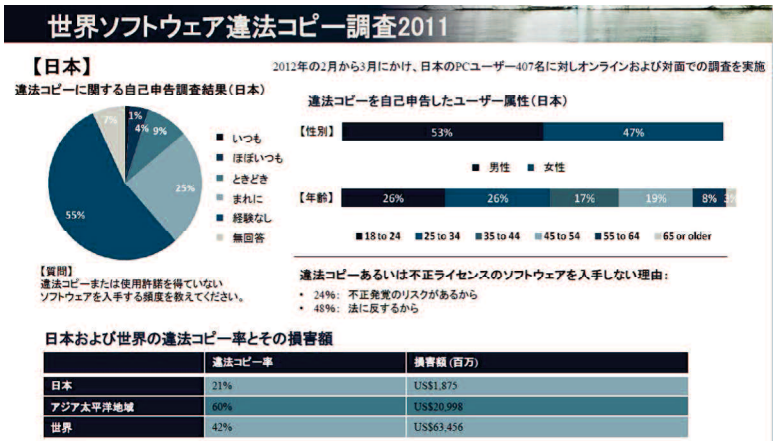


図 6.2: 世界ソフトウエア違法コピー調査 2011 による結果 (日本)[90]

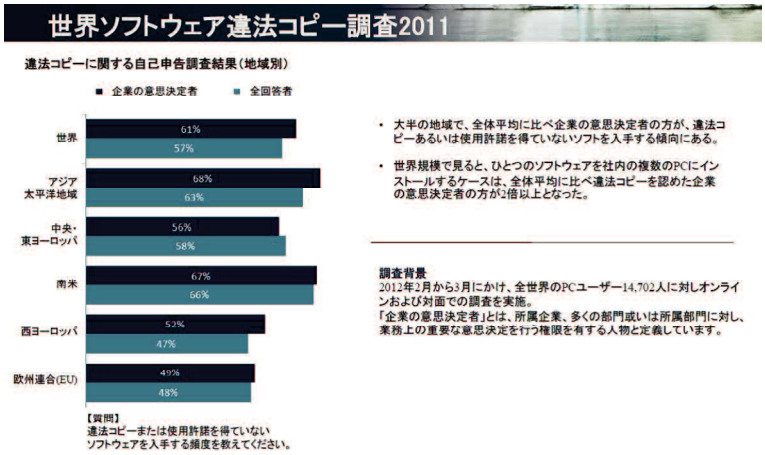


図 6.3: 世界ソフトウエア違法コピー調査 2011 による結果 (地域別)[90]

ソフトウエアシステム著者権侵害問題は，本論文で提案した類似度求める方法を利用して解決可能であると考えられる．それは，C 言語だけではなく，さまざまのプロジェクト向け言語で開発されたシステムに対して対応できる構成パターンと類似度求める手法を対策すれば解決できると思われる．したがって，本研究で得られる成果を踏まえた上，今後はソフトウエアシステムの違法コピー諸問題の解決に生かしていきたい．

6.3 今後の課題

本研究では，実用の盗用 C 言語プログラムを判別するために有効な NTO と DM 手法を提案しているが，今後は適用対象のプログラミング言語を広げ，Java，PHP，C ＃，C ＋＋などそれぞれのプログラミング言語の特性や構造特徴等について，具体的に分析して本研究の提案手法を理論的かつ実験的に評価していく必要がある．そして提案した NTO 手法に対して，最大共通部分木を求めるため，親子ペアの完全一致パターン即ち形もノード記述文（プログラムソースコード）も同じ場合のみマッチングパターンを取ることではなく，形が一致で子ノードの集合の一致部分もマッチングパターンを取る．ここで，急成長が続いているソフトウエアシステムに

おける，プログラム著者権侵害問題について，本研究成果を応用した解決法を検討し，本研究手法も改善をして行きたい．

参考文献

[1] http://www.mext.go.jp/b_menu/shingi/chukyo/chukyo3/028/siryo/06081106/002.htm

[2] http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

[3] 小田 悠介，上村 康輔, 若林 茂: “プログラム間の類似性の定量化手法”，教育システム情報学会 (51), pp.103-108 (2013.03).

[4] 山中 伸一: “コンピュータ・プログラムの著作権侵害は否定したが, 営業秘密の不正行為が成立する可能性があるとした事例”，日米法学会，アメリカ法, pp326-333 (1995.02).

[5] 村林 隆一 : “新判決例研究 (第 70 回) プログラムの著作権侵害事件において, 侵害者が正規品を購入することによって許諾料を支払った場合における著作権者の損害, その他 [1. 東京地裁平成 13.6.16 判決 2. 大阪地裁平成 15.10.23 判決 3. 名古屋地裁平成 18.6.8 判決]”，経済産業調査会知的財産情報センター，知財ぷりずむ 5 (58)，pp.109-119 (2007).

[6] 手嶋 豊: “アメリカ法におけるコンピュータプログラムの著作権侵害 (1)”，廣島法學 15(3), pp139-150 (1992.01).

[7] http://www.mext.go.jp/b_menu/hakusho/nc/t19920608001/t19920608001.html

[8] http://law.e-gov.go.jp/htmldata/S45/S45HO048.html

[9] 久保田 裕 : “コンピュータソフトウェアの法的保護”, 情報処理学会，Vol.36(4), pp.327-336 (1995.04).

[10] http://www.furutani.co.jp/kiso/tyosaku2.html

[11] 木村　勢一 : “ンピュータ・プログラムの著作権法と特許法とによる保護の変遷”, 特集「平成 18 年度著作権委員会」，パテント，Vol.60，No.6，(2007).

[12] Karl J. Ottenstein :“An Algorithmic Approach to the Detection and Prevention of Plagiarism”, CSD-TR200，Vol.103(2), pp.312-39 (1976).

[13] Wise, Michael J. : “Detection of similarities in student programs : YAP'ing may be preferable to Plague'ing”, SIGSCI Technical Symposium，Kansas City，USA，pp.268-271 (1992).

[14] G. Whale:“Identification of Program Similarity in Large Populations”，Science & Mathematics，Computer Journal，Vol.33, Issue 2，pp.140-146 (1988).

[15] Clough, Paul.:“Plagiarism in natural and programming languages:An overview of current tools and technologies”，Research Memoranda : CS-00-05，Department of Computer Science，University of Sheffield, pp.168-172 (1992).

[16] P.Zhang and S.Yang : “Research and Realization of Recognition Method on C Program Similar Code”，Dailian University of Technology，(2008).

[17] 福田 勇一 : “レポートプログラム処理システム”, 情報処理学会，全国大会講演論文集，第 53 回, pp.301-302 (1997).

[18] 福田 勇一 : “教育支援システム”, 情報処理学会，全国大会講演論文集，第 57 回，pp.318-319 (2000).

[19] 福田 勇一 : “C プログラム比較システム”, 情報処理学会，全国大会講演論文集，第 59 回，pp.231 (1999).

[20] 小田 悠介, 若林 茂：“プログラム間の類似性の定量化手法”, 神戸市立工業高等専門学校研究紀要 (51), pp.103-108 (2013.03).

[21] E.W.Myers：“An O(ND) difference algorithm and its variations”, Algorithmixa, 1, pp.251-266 (1986).

[22] 長橋 賢児：“類似度に基づくソフトウェア品質の評価”, 情報処理学会研究報告ソフトウェア工学（SE）, 2000-SE-126, pp.65-72 (2000).

[23] 山本, 松下, 神谷, 井上:“ソフトウェアシステムの類似度とその計測ツールSMMT”, 電子情報通信学会論文誌 D-I, Vol. J85-D-I, No.6, pp.503-511 (2002).

[24] 北川, 杉山：“自己組織化マップを用いた Java ソースコード間類似度測定ライブラリの試作”, 電子情報通信学会技術報告，SS2004-67, pp.19-24 (2004).

[25] Verco, Kristina L., and Michael J. Wise.：“Software for detecting suspected plagiarism comparing structure and attribute-counting systems”，Proceedings of the 1st Australian Conference on Computing Science Education，Sydney，Vol.102(3)，pp.3-5 (1996)．

[26] J.E．Grass：“Cdiff: A syntax directed differencer for C++ programs”, C++ Conference, pp.181-193, Portland, Oregon (1992).

[27] 会沢，練，飯田, 井上, 鳥居：“構文木の比較に基づいたプログラム差分の表示方法”, 情報処理学会第 48 回全国大会講演論文集, pp.111-112 (1994).

[28] I. Baxter, A. Yahin, L. Moura, M. Sant’Anna and L. Bier：“Clone Detection Using Abstract Syntax Trees”, IEEE，Published in the Proceedings of ICSM，pp.16-19 (1998)．

[29] M.Chilowicz, E.Duris and G.Roussel:“A Display Method of Program Difference Based on Parsing Tree Comparison”, Program Comprehension, ICPC ’09. IEEE 17th International Conference，pp.243-247 (2009)．

[30] L.Jiang, G.Misherghi, Z.Su and S.Glondu：“DECKARD: Scalable and Accurate Tree-Based Detection of Code Clones”, IEEE Computer Society Washington，DC, USA, pp.96-105 (2007)．

[31] http://www.ccfinder.net．

[32] P．N. Robinson, A．Wollstein, U．Bohme and B．Beattie：“Ontologizing gene-expression microarray data: characterizing clusters with Gene Ontology”, Oxford Journals，Science & Mathematics，Bioinformatics，Vol.20, Issue 6，pp.979-981 (2003).

[33] P．Stothard and D．S. Wishart：“Circular genome visualization and exploration using CGView”, Oxford Journals，Science & Mathematics，Bioinformatics，Vol.21, Issue 4，pp.537-539 (2004).

[34] Alok J. Saldanha：“Java Treeview-extensible visualization of microarray data”, Oxford Journals，Science & Mathematics，Bioinformatics，Vol.20, Issue 17，pp.3246-3248 (2004).

[35] 阿久津 達也, 深川 大路, 高須 淳宏：“木の編集距離の文字列の編集距離による近似”, 電子情報通信学会技術研究報告. COMP, コンピュテーション, Vol.106，No.63, pp.17-24 (2006).

[36] Tatsuya Akutsu：“A bisection algorithm for grammar-based compression of ordered trees”, Information Processing Letters, Vol.110，Issues 18-19, pp.815-820 (2010).

[37] 池田，小林，波多野，深川：“複数観点を用いた木類似度算出法によるソースコードの類似構造の抽出”, 第 3 回データ工学と情報マネジメントに関するフォーラム論文集，D3-3，(2011)．

[38] 齋藤 裕明, 古賀 久志，渡辺 俊典，横山 貴紀 : “木編集距離を利用した木データの構造と内容の類似性を反映する分類手法”, 電子情報通信学会技術研究報告. DE，Vol.106，No.97, pp.7-12 (2006).

[39] 小野里 卓也，古賀 久志，渡辺 俊典 : “木の DP マッチングを利用した DTD 類似度の考察”, 電子情報通信学会技術研究報告. NLC, 言語理解とコミュニケーション，Vol.104，No.669, pp.103-108 (2005).

[40] K.Zhang and Dennis Shasha : “Simple Fast Algorithms for the Editing Distance between Trees and Related Problems”, SIAM Journal on Computing, DOI:10.1137/0218082, Vol.18(6)，pp.1245-1262 (1989).

[41] Mateusz Pawlik and Nikolaus Augsten : “Tree edit distance: Robust and memory-efficient’, Information Systems，Vol.56，No.669, pp.157-173 (2016).

[42] Paolo Ciancarini, Angelo Di Iorio, Carlo Marchetti, Michele Schirinzi and Fabio Vitali : “Bridging the gap between tracking and detecting changes in XML”, Software: Practice and Experience. Vol.46, Issue.2, pp.227-250 (2016).

[43] Andrea Torsello and Edwin R. Hancock : “Computing approximate tree edit distance using relaxation labeling”, Software: Practice and Experience. Vol.24, Issue.8, pp.1089-1097 (2003).

[44] 小堀 一雄, 山本 哲男, 松下 誠, 井上 克郎 : “類似度メトリクスを用いた Java ソースコード間類似度測定ツールの試作”, 電子情報通信学会技術研究報告. SS, ソフトウェアサイエンス，Vol.103(102) (2003).

[45] Weyuker, Elaine J. : “Testing componentbased software: A cautionary tale”, IEEE Computer Society, Software15.5，pp.54-59 (1998).

[46] A. W. Brown and K. C. Wallnau : “International workshop on component-based software engineering”, IEEE Computer Society, Print ISBN:1-58113-074-0 , Los Angeles, CA, USA，pp.714 (1999).

[47] Ren, Xiaoxia, et al. : “Chianti: A Tool for Change Impact Analysis of Java Programs”, OOPSLA ’04 Proceedings of the 19th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications，ACM New York, NY, USA, Vol.39, Issue.10, pp.432-448 (2004).

[48] J.Han : “A comprehensive interface definition framework for software components”, Software Engineering Conference，IEEE Computer Society，ISBN:0-8186-9183-2，Taipei, pp.110-117 (2014).

[49] K.Koboro, M.Matsushita and K.Inous : “Using Java applets and CORBA for multi-user distributed applications”，IEEE Computer Society，Vol.1, Issue.3,ISSN :1089-7801, pp.43-55 (1997).

[50] ウィルソン, R. J. 「グラフ理論」原書第 4 版，西関隆夫・西関裕子，株式会社近代科学社，(2007).

[51] R．ディーステル. 「グラフ理論」根上 生也 訳，丸善出版，ISBN : 978-4-621-06185-5，(2012).

[52] 山本恭之，有福洋史，平田耕 : “根付き無順序木の編集距離とその変種の比較”，情報処理学会研究報告，(2011.03).

[53] 大倉 暢仁，平田 耕一，久保山 哲二，原尾 正輝 : “ラベル付き根付き無順序木のフィルタリング”，電気関係学会九州支部連合大会講演論文集, pp.506-506 (2006).

[54] http://home.a00.itscom.net/hatada/c-tips/ast/ast01.html

[55] 小田 悠介，札場 寛之，ニュービッグ グラム : “ソースコード構文木からの統計的自動コメント生成”，研究報告自然言語処理（NL）2014-NL-219(12), pp.1-9 (2014.12).

[56] 木津 栄二郎，大森 隆行，丸山 勝久 : “コードの編集履歴を用いたプログラム変更の検出”，情報処理学会論文誌, Vol.56(2), pp.611-626 (2015.02).

[57] Stanley M. Selkow：“The tree-to-tree editing problem”，Information Processing Letters, Vol.6, Issue 6, pp.184-186 (1977.12).

[58] Kaizhong Zhang, Dennis Shasha：“Simple Fast Algorithms for the Editing Distance between Trees and Related Problems”，SIAM J. Comput, Vol.18, Issue.6, pp.1245-1262 (1977.12).

[59] 齋藤 裕明，古賀 久志，渡辺 俊典，横山 貴紀：“木編集距離を利用した木データの構造と内容の類似性を反映する分類手法”, 電子情報通信学会技術研究報告，DE, データ工学 Vol.106(97), pp.7-12 (2006).

[60] T.Akutsu and M.M. Halldorsson：“On the approximation of largest common point sets and largest common subtrees”, in: Proc. 5th Ann. Internat. Symp. on Algorithms and Computation, Lecture Notes in Computer Science (LNCS), Vol.834, Springer, Berlin, pp.405-413 (1994).

[61] Fernandez, M. L., and Valiente, G.：“A graph distance metric combining maximum common subgraph and minimum common supergraph”，Pattern Recognition Letters，Vol.22，pp.753-758 (2001).

[62] H.Zha，X.He，Chris Ding，Horst Simon and M.Gu：“Bipartite graph partitioning and data clustering”，ACM New York, NY, USA, ISBN:1-58113-436-3, pp.25-32 (2001).

[63] A.V. Aho, J.D. Ullman, D.S. Aho and J.D：“Hirschberg Bounds on the complexity of the longest common subsequence problem”，J. ACM, Vol.1(23), pp.1-12 (1976).

[64] Tatsuya Akutsu：“An RNC Algorithm for Finding a Largest Common Subtree of Two Trees”，IEICE TRANSACTIONS on Information and Systems，Vol.E75-D，No.1，pp.95-101 (1992).

[65] 増田 澄男，森 一郎，田中 栄一：“二つの木の最大共通部分グラフを求めるアルゴリズム”，電子情報通信学会論文誌, A, 基礎・境界 J77-A(3), pp.460-470 (1994).

[66] 垣村 尚徳：“対称二部グラフのマッチング構造”，情報処理学会研究報告アルゴリズム，Vol.84, pp.33-40 (2008).

[67] Satoshi, T., & Shuichi, U. ：“Stable Matchings in Trees”，Institute of Electronics, Information and Communication Engineers, MSS, Vol.113, No.279, pp.61-66 (2013.10).

[68] Peng, H., & Fujii, T. : “User Mobility Aware Resource Allocation for Cognitive Radio Networks”，Institute of Electronics, Information and Communication Engineers, SR, Vol.113, No.57, pp.33-39 (2013.05).

[69] A.V. Aho, D.S. Hirschberg and J.D. Ullman：“Bounds on the complexity of the longest common subsequence problem”，J. Assoc: Comput，No.1, pp.1-12 (1976).

[70] Masek, W. J., & Paterson, M. S. : “A faster algorithm computing string edit distances”，Journal of Computer and System Sciences，Vol.20, Issue 1, pp.18-31 (1980).

[71] http://sacsis.hpcc.jp/2009/gpu/problem.html

[72] https://ja.wikipedia.org/wiki/

[73] Philip Bille：“A survey on tree edit distance and related problems”，Theoretical Computer Science，Vol.337, Issue 1-9, pp.217-239 (2005).

[74] W.Chen：“New Algorithm for Ordered Tree-to-Tree Correction Problem”，Journal of Algorithms，Vol.40, Issue 2, pp.135-158 (2001).

[75] 山本哲男, 松下誠, 神谷年洋, 井上克郎：“クローン検出ツールを用いたソフトウェアシステムの類似度調査”，電子情報通信学会技術研究報告，SS2001-15, Vol.101, No.240, pp.25-32 (2001)．

[76] 包 胡日査, 中田 充，葛 崎偉：“C 言語プログラムの構文木表現”，コンピュータ＆エデュケーション, Vol.36, pp.56-61 (2014).

[77] 林 晴比古.「新・C 言語入門 シニア編」ソフトバンククリエイティブ-新訂版，ISBN-10:4797325623，(2004.02).

[78] 近藤 嘉雪.「C プログラマのためのアルゴリズムとデータ構造」ソフトバンククリエイティブ，ISBN-10:4797304952，(1998.03).

[79] http://detail.chiebukuro.yahoo.co.jp/qa/question_detail/q14104701254

[80] https://ja.wikipedia.org/wiki/

[81] http://www.atmarkit.co.jp/fxml/dictionary/indexpage/xmlindex.html

[82] http://coins-compiler.osdn.jp/COINSdoc/visual/visual-frame.html

[83] http://www.graphviz.org/

[84] 市川, 橋本, 徳永, 田中: “テキスト構文構造類似度を用いた類似度を用いた類似文検索”, 情報処理学会研究報告, 2005-FI-79(6),pp.39-46 (2005).

[85] 包 胡日査, 森田 廉，葛 崎偉，中田 充：“構文木の類似度を用いた C 言語プログラムの類似性検証”，電子情報通信学会技術研究報告. CST, コンカレント工学, Vol.110, No.370, pp.83-86 (2011.01).

[86] 包 胡日査, 中田 充，葛 崎偉:“順序木の類似度を評価する手法の提案”，電子情報通信学会技術研究報告. MSS, システム数理と応用, Vol.112, No.383, MSS2012-56, pp.61-64 (2013.01).

[87] H.Bao, M.Nakata, R.Wu and Q.W. Ge: “Similarity Evaluation of Ordered Trees and Its Application to Similarity Verification of C Language Programs”, Proc. ITC-CSCC2013, pp.64-67 (2013.07).

[88] 小 堀 一 雄: “ソースコードの静的解析によるソフトウェア保守支援に関する研究”，大阪大学大学院情報科学研究科, (2013.07).

[89] http://www2.accsjp.or.jp/piracy/

[90] http://globalstudy.bsa.org/

[91] http://bsa.or.jp/about-bsa/

[92] 西垣 正勝，本部 栄成，米山 裕太，高橋 健太：“すれちがい通信を用いた分散型不正コピー検知の提案”，情報処理学会論文誌，Vol.54，No.9，pp.2188-2196 (2013).

[93] 山本賢，岡山聖彦，山井成良：“ソフトウェア不正コピー対策のための LAN アクセス制御システム”，第 7 回情報科学技術フォーラム (FIT2008)，科学技術レターズ，Vol.4，pp.23-26 (2008).

[94] 中村 直己，西垣 正勝，曽我 正和，田窪 昭夫：“サーバ・クライアント間の同期型情報管理によるソフトウェア保護”，情報処理学会研究報告マルチメディア通信と分散処理（DPS）2002(12(2001-DPS-106)), pp.265-270 (2002).

付録 I: JAVA，C#で作成られたスクリプトファイル (実用プログラム)

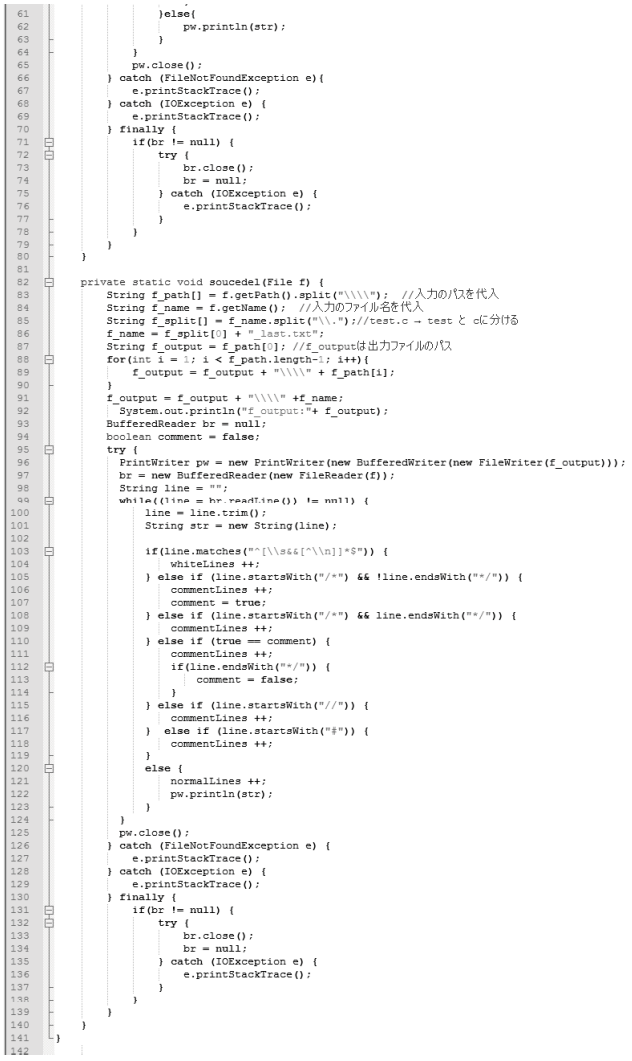
ここでは，まずEclipse環境でC言語ソースプログラムの前処理作業の *abstract.java* ソースプログラムを示す．このプログラムでは3章で説明したようにプログラムの七つの作業である．

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;

public class CcouceP {
    static long normalLines = 0;
    static long commentLines = 0;
    static long whiteLines = 0;

    public static void main(String[] args) {
        File f = new File("..\\JavaData");
        File[] codeFiles = f.listFiles();
        for(File child : codeFiles){
            if(child.getName().matches(".*\\.c$")) {
                parse(child);
            }
            else if (child.getName().matches(".*\\.txt$")){
                soucedel(child);
            }
        }
    }
    private static void parse(File f) {
    ////出力ファイルのパスを求める
        String f_path[] = f.getPath().split("\\\\");  //入力のパスを代入
        String f_name = f.getName();  //入力のファイル名を代入
        String f_split[] = f_name.split("\\.");//test.c → test と cに分ける
        f_name = f_split[0] + "_output.txt";
        String f_output = f_path[0]; //f_outputは出力ファイルのパス
        for(int i = 1; i < f_path.length-1; i++){
            f_output = f_output + "\\\\" + f_path[i];
        }
        f_output = f_output + "\\\\" +f_name;
        System.out.println("f_output:"+ f_output);
        BufferedReader br = null;
        boolean comment = false;
        try {
            PrintWriter pw = new PrintWriter(new BufferedWriter(new FileWriter(f_output)));
            br = new BufferedReader(new FileReader(f));
            String line = "";
            while((line = br.readLine()) != null) {
                line = line.trim();
                String str = new String(line);
                String[] strAry1 = str.split("/\\*");
                String[] strAry2 = str.split("//");
                if(strAry1.length!=1){
                    strAry1[1]="/*"+strAry1[1];
                    for (int i=0; i<strAry1.length; i++) {
                        pw.println(strAry1[i]);
                    }
                }
                else if(strAry2.length!=1){
                    strAry2[1]="//"+strAry2[1]; //分割したあと分割の後ろの部分の前に//を追加して置く！
                    for(int i=0; i<strAry2.length; i++){
                        pw.println(strAry2[i]);
                    }
```

図 6.4: abstract.java（1）



```
                }else{
                    pw.println(str);
                }
            }
            pw.close();
        } catch (FileNotFoundException e){
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if(br != null) {
                try {
                    br.close();
                    br = null;
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    }

    private static void soucedel(File f) {
        String f_path[] = f.getPath().split("\\\\");  //入力のパスを代入
        String f_name = f.getName();  //入力のファイル名を代入
        String f_split[] = f_name.split("\\.");//test.c → test と cに分ける
        f_name = f_split[0] + "_last.txt";
        String f_output = f_path[0]; //f_outputは出力ファイルのパス
        for(int i = 1; i < f_path.length-1; i++){
            f_output = f_output + "\\\\" + f_path[i];
        }
        f_output = f_output + "\\\\" +f_name;
        System.out.println("f_output:"+ f_output);
        BufferedReader br = null;
        boolean comment = false;
        try {
          PrintWriter pw = new PrintWriter(new BufferedWriter(new FileWriter(f_output)));
          br = new BufferedReader(new FileReader(f));
          String line = "";
          while((line = br.readLine()) != null) {
              line = line.trim();
              String str = new String(line);

              if(line.matches("^[\\s&&[^\\n]]*$")) {
                  whiteLines ++;
              } else if (line.startsWith("/*") && !line.endsWith("*/")) {
                  commentLines ++;
                  comment = true;
              } else if (line.startsWith("/*") && line.endsWith("*/")) {
                  commentLines ++;
              } else if (true == comment) {
                  commentLines ++;
                  if(line.endsWith("*/")) {
                      comment = false;
                  }
              } else if (line.startsWith("//")) {
                  commentLines ++;
              }  else if (line.startsWith("#")) {
                  commentLines ++;
              }
              else {
                  normalLines ++;
                  pw.println(str);
              }
          }
          pw.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if(br != null) {
                try {
                    br.close();
                    br = null;
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

図 6.5: abstract.java（2）

次に，*FunsSplit.java* では Full プログラムのすべての関数を関数ごとに分けて，関数名はファイル名として一つずつ出力される．最後にすべての関数をまとめてひとつのファイルに保存されて出力作業である．

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.Stack;


public class FunsSplit {
    static long  brace = 0;     //関数名を確定するための変数，大括弧が初めに会った場合その行或いは前の行から関数名読み取る．
    static long N = 1;
    static int Fun_number = 0;    //関数の数を求める．
    static long  line_number = 0;    //行の数を求める．
    static String Fun_name = null;      //関数名を初期化する．
    static String Fun_name_last = null;


    public static void main(String[] args) throws IOException {
        //File f = new File("C:\\Users\\bao\\workspace\\CodeProcess\\JavaData");
        File f = new File("..\\JavaData");
        File[] codeFiles = f.listFiles();
        for(File child : codeFiles){
            if(child.getName().matches(".*"+"last"+ "\\.txt$")) {
                FunSplit(child);
            }
        }

    }


    static void FunSplit(File inputFile) throws IOException {
        ArrayList<String> strOut = new ArrayList();     //
        BufferedReader br = null;
        br = new BufferedReader(new FileReader(inputFile));
        String line = "";
        String line_before = null;
        Stack stk = new Stack();
        while((line = br.readLine()) != null) {

            line = line.trim();
            line_number ++;
            String str = new String(line);

            if (line.indexOf("{") > -1){
                stk.push("{");
                brace++ ;

                if (brace == 1){

                    if(line.startsWith("{")){
```

図 6.6: FunSplit.java（3）

```
55
56                  int index = line_before.indexOf("(");
57                  int index1 = line_before.substring(0,index).indexOf(" ");
58                  Fun_name = line_before.substring(0,index).substring(index1+1);
59                  //System.out.println(line_before.substring(0,index).substring(index1+1));
60                  //System.out.println(Fun_name);
61                  if(Fun_name.startsWith(" ")){
62                      //int index11 = Fun_name.indexOf(" ");
63                      //Fun_name_last = Fun_name.substring(0,index11);
64                      String[] strAry = Fun_name.split(" ");
65                      Fun_name_last = strAry[0];
66                  } else {
67                          Fun_name_last = Fun_name;
68                  }
69                  System.out.println("Fun_name_last" + Fun_name_last);
70              } else{
71
72                  int index = line.indexOf("(");
73                  int index1 = line.substring(0,index).indexOf(" ");
74                  Fun_name = line.substring(0,index).substring(index1+1);
75                  //System.out.println(line.substring(0,index).substring(index1+1));
76                  //System.out.println(Fun_name);
77                  if(Fun_name.startsWith(" ")){
78                      int index11 = Fun_name.indexOf(" ");
79                      Fun_name_last = Fun_name.substring(0,index11);
80                  } else {
81                          Fun_name_last = Fun_name;
82                  }
83                  System.out.println("Fun_name_last" + Fun_name_last);
84              }
85          }
86
87          System.out.println(str);
88          strOut.add(str);
89
90      }  else if (line.indexOf("}") > -1){
91
92          stk.pop();
93          //pw.println(str);
94          System.out.println(str);
95          strOut.add(str);
96
97          if (stk.empty()){
98              Fun_number ++;
99              System.out.println("line_number→" +line_number);
100
101             //File file = new File("C:\\Users\\bao\\workspace\\CodeProcess\\JavaData\\functions\\" + Fun_name +"_function.txt");
102             String pathName = "..\\JavaData\\functions\\";
103             String outFileName = pathName + Fun_name_last +"_function.txt";
104
105             FileWrite outFileHand = new FileWrite();
106             outFileHand.fileName = outFileName;
107             outFileHand.writeDataCSV(strOut,false);
108             brace = 0;
109             strOut.clear();     //一回読み 出した 記録が削除する.
```

図 6.7: FunSplit.java（4）

```
110                 System.out.println("Fun_number→" + Fun_number);
111
112             }
113         }
114         else {
115
116             System.out.println(str);
117             strOut.add(str);
118         }
119
120         line_before = line;
121
122     }
123
124
125 }
126 }
```

図 6.8: FunSplit.java（5）

図 6.9: FunSplit.java（6）

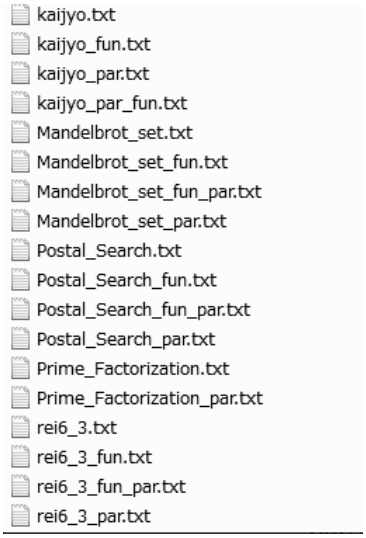


図 6.10: FunSplit.java（7）

前処理出力されたファイルが図 6.9,6.10 のように示す．(ここでは一部のファイルが示す)

これらかの処理は前処理後出力されたファイルを構文木 (XML) グラフ化する操作であり，ソースプログラムを図 $CProgramsXML.cs$ のように示す．このプログラムは 3 章で提案した構文木部品を用いて関数の呼び出し順で処理される作業である．出力された構文木 XMLformat は四つの属性情報が持つ，各節点のレベル情報，深さ優先順での ID 属性，子ノードの数属性およびソースコードを持つノード属性が存在するような作業である．

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Text.RegularExpressions;
using System.Xml;
using System.Collections;
using System.Diagnostics;
using System.Net;

namespace FangAn
{
    public partial class Form1 : Form
    {
        public string TxtName = "";
        List<string> l1;
        List<string[]> l2;
        List<string> List_content;
        List<string> list_history;
        public int label;
        public Form1()
        {
            InitializeComponent();
        }

        public string getPath()
        {
            StringBuilder sb = new StringBuilder();
            return sb.AppendFormat("{0}//XMLFiles//{1}.xml", Application.StartupPath, TxtName).ToString();
        }
        public void XML(XmlDocument xmldoc)
        {
            //まずfileがあるかどうか確認，なければ新規作成する。
            if (!Directory.Exists(Application.StartupPath + "//XMLFiles"))
            {
                Directory.CreateDirectory(Application.StartupPath + "//XMLFiles");
            }
            BianLiJieDian(xmldoc.ChildNodes[1]);
            xmldoc.Save(@"" + getPath() + "");//保存するパス
        }
        private void button1_Click(object sender, EventArgs e)
        {
                #region
                TxtName = "";
                l1 = new List<string>();
                l2 = new List<string[]>();
                List_content = new List<string>();
                list_history = new List<string>();
                label = 1;
                list_history.Add("printf");
```

図 6.11: CProgramsXML.cs（1）

```
list_history.Add("print");

Stream myStream = null;
OpenFileDialog openFileDialog1 = new OpenFileDialog();
openFileDialog1.InitialDirectory = "E:\\Txt转换XML2\\XML转换\\txt";
openFileDialog1.Filter = "txt files (*.txt)|*.txt";
openFileDialog1.FilterIndex = 2;
openFileDialog1.RestoreDirectory = true;
if (openFileDialog1.ShowDialog() == DialogResult.OK)
{
    try
    {
        if ((myStream = openFileDialog1.OpenFile()) != null)
        {
            using (myStream)
            {
                using (StreamReader sr = File.OpenText(openFileDialog1.FileName))
                {
                    try
                    {
                        TxtName = System.IO.Path.GetFileNameWithoutExtension(openFileDialog1.FileName);
                        String resources = sr.ReadToEnd();//ttxtからデータを取る
                        #region 解析Method主体
                        //Regex r_main = new Regex(@"(void|int|string|short|long|float|double|char|struct|union|enum)\s+\w+\((void|int|string|short
                        Regex r_main = new Regex(@"(void|int|string|short|long|float|double|char|struct|union|enum)\s+\w+\(.*?\)");//解析Method主体
                        MatchCollection ms_main = r_main.Matches(resources);
                        foreach (Match m in ms_main)
                        {
                            l1.Add(m.Value);
                        }
                        if (l1.Count > 0)
                        {
                            Regex r_main2 = new Regex(@"^(void|int|string|short|long|float|double|char|struct|union|enum)\s+(.+?)\((.*?)\)");
                            for (int j = 0; j < l1.Count; j++)
                            {
                                MatchCollection ms_main2 = r_main2.Matches(l1[j]);
                                if (ms_main2.Count > 0)
                                {
                                    string s = "";
                                    foreach (Match m in ms_main2)
                                    {
                                        for (int i = 1; i < m.Groups.Count; i++)
                                        {
                                            if (!string.IsNullOrEmpty(m.Groups[i].Value.Trim()))
                                            {
                                                s = s + m.Groups[i].Value + '+';
                                            }
                                        }
                                    }
                                    l2.Add(s.Split(new char[] { '+' }, StringSplitOptions.RemoveEmptyEntries));
                                }

                            }

```

図 6.12: CProgramsXML.cs（2）

```
110     }
111     #endregion
112
113     #region 解析主体
114     XmlDocument xmldoc = new XmlDocument();            //節を宣言
115     XmlDeclaration dec = xmldoc.CreateXmlDeclaration("1.0", "utf-8", null);
116     xmldoc.AppendChild(dec);            //ひとつの節追加
117     //XmlElement xmlelem = xmldoc.CreateElement("cToxmlRoot");
118     //xmldoc.AppendChild(xmlelem);
119     XmlElement xmlelem1 = xmldoc.CreateElement("funcType");
120     xmlelem1.SetAttribute("level", "0");
121     xmlelem1.SetAttribute("label", label.ToString());
122     xmlelem1.SetAttribute("text", l2[0][0]);
123     xmlelem1.InnerText = l2[0][0];
124     xmldoc.AppendChild(xmlelem1);
125     XmlElement xmlelem_main = xmlelem1;
126     label++;
127
128     int index_main = 0;
129     if (l2[0].Length == 2)
130     {
131
132         XmlElement xmlelem2 = xmldoc.CreateElement("funcName");
133         xmlelem2.SetAttribute("level", "1");
134         xmlelem2.SetAttribute("label", label.ToString());
135         xmlelem2.SetAttribute("text", l2[0][1]);
136         xmlelem2.InnerText = l2[0][1];
137         xmlelem_main.AppendChild(xmlelem2);
138         xmlelem_main = xmlelem2;
139         label++;
140         index_main = 2;
141     }
142     if (l2[0].Length == 3)
143     {
144         //XmlElement xmlelem1 = xmldoc.CreateElement("funcType");
145         //xmlelem1.SetAttribute("level", "0");
146         //xmlelem1.SetAttribute("label", label.ToString());
147         //xmlelem1.InnerText = l2[0][0];
148         //xmlelem_main.AppendChild(xmlelem1);
149         //label++;
150         //xmlelem_main = xmlelem1;
151
152         XmlElement xmlelem2 = xmldoc.CreateElement("funcName");
153         xmlelem2.SetAttribute("level", "1");
154         xmlelem2.SetAttribute("label", label.ToString());
155         xmlelem2.SetAttribute("text", l2[0][1]);
156         xmlelem2.InnerText = l2[0][1];
157         xmlelem_main.AppendChild(xmlelem2);
158         label++;
159         xmlelem_main = xmlelem2;
160
161         XmlElement xmlelem3 = xmldoc.CreateElement("funcArgType");
162         xmlelem3.SetAttribute("level", "2");
163         xmlelem3.SetAttribute("label", label.ToString());
```

図 6.13: CProgramsXML.cs（3）

```
        xmlelem3.SetAttribute("text", l2[0][2]);
        xmlelem3.InnerText = l2[0][2];
        xmlelem_main.AppendChild(xmlelem3);
        xmlelem_main = xmlelem3;
        label++;
        index_main = 3;
    }
    #endregion

    #region 解析内容
    resources = Regex.Replace(resources, @"(void|int|string|short|long|float|double|char|struct|union|enum)\s+\w+\(.*?\)", "&&&") + "&&&";
    List<string> l3 = new List<string>();
    string[] s_conlist = resources.Split(new string[] { "&&&" }, StringSplitOptions.RemoveEmptyEntries);
    foreach (string s in s_conlist)
    {
        List_content.Add(s);
    }
    Regex r_content = new Regex(@"\({[\s\S]*}\)");
    MatchCollection ms_content = r_content.Matches(s_conlist[0]);
    if (ms_content.Count > 0)
    {
        string[] slist = ms_content[0].Groups[1].Value.Split(new string[] { "\r\n" }, StringSplitOptions.RemoveEmptyEntries);
        #region
        int s_dex = 0;
        bool IsMatch = true;
        bool b_if = false;
        bool a_if = true;
        bool d_if = false;
        bool b_while = false;
        bool d_while = false;
        bool a_while = true;
        bool b_switch = false;
        bool d_switch = false;
        bool a_switch = true;
        bool b_for = false;
        bool d_for = false;
        bool a_for = true;
        bool b_do = false;
        bool d_do = false;
        bool a_do = true;
        int s_dex4 = 0;
        string s_yuju = "";

        foreach (string s in slist)
        {

            #region if
            if (a_if && IsMatch && a_do && a_for && a_if && a_switch && a_while)
            {
                Regex r2 = new Regex(@"if\s*\(.+\)");//if
                Regex r3 = new Regex(@"else if\(.+?\)");//if
                if (!r3.IsMatch(s))
                {
                    MatchCollection ms2 = r2.Matches(s);
```

図 6.14: CProgramsXML.cs（4）

```
            if (ms2.Count > 0)
            {
                b_if = true;
                a_if = false;
                s_yuju = "";
                IsMatch = false;
            }
        }
    }
    if (b_if)
    {
        s_yuju += "\r\n" + s;
        IsMatch = false;
        if (s == "{")
        {
            d_if = true;
            s_dex4++;
        }
        if (s == "}")
        {
            s_dex4--;
        }
        if (s_dex4 == 0 && d_if == true)
        {
            if (s_dex < slist.Length - 1)
            {
                if (slist[s_dex + 1].IndexOf("else") > -1 || slist[s_dex + 1].IndexOf("else if") > -1)
                {
                    d_if = false;
                }
                else
                {
                    b_if = false;
                    d_if = false;
                    H_If(s_yuju, xmldoc, xmlelem_main, index_main);
                    a_if = true;



                }
            }
            else
            {
                b_if = false;
                d_if = false;
                H_If(s_yuju, xmldoc, xmlelem_main, index_main);
                a_if = true;

            }
        }
    }
    #endregion

    #region whileループ
    if (a_if && IsMatch && a_do && a_for && a_if && a_switch && a_while)
```

図 6.15: CProgramsXML.cs（5）

```
272     {
273         Regex r_while = new Regex(@"while\s*\(.+\)");//函数定义
274         MatchCollection ms_while = r_while.Matches(s);
275         if (ms_while.Count > 0)
276         {
277             b_while = true;
278             s_yuju = "";
279             s_dex4 = 0;
280             a_while = false;
281             IsMatch = false;
282         }
283     }
284     if (b_while)
285     {
286         IsMatch = false;
287         s_yuju += "\r\n" + s;
288         if (s == "{")
289         {
290             d_while = true;
291             s_dex4++;
292         }
293         if (s == "}")
294         {
295             s_dex4--;
296         }
297         if (s_dex4 == 0 && d_while == true)
298         {
299             b_while = false;
300             d_while = false;
301             a_while = true;
302             H_While(s_yuju, xmldoc, xmlelem_main, index_main);
303 
304         }
305     }
306     #endregion
307 
308     #region forループ
309     if (a_if && IsMatch && a_do && a_for && a_if && a_switch && a_while)
310     {
311         Regex r_for = new Regex(@"(for)\s*\((.+);(.+);(.+)\)");//函数定义
312         MatchCollection ms_for = r_for.Matches(s);
313         if (ms_for.Count > 0)
314         {
315             b_for = true;
316             s_yuju = "";
317             s_dex4 = 0;
318             a_for = false;
319             IsMatch = false;
320         }
321     }
322     if (b_for)
323     {
324         IsMatch = false;
325         s_yuju += "\r\n" + s;
```

図 6.16: CProgramsXML.cs（6）

```
326         if (s == "{")
327         {
328             d_for = true;
329             s_dex4++;
330         }
331         if (s == "}")
332         {
333             s_dex4--;
334         }
335         if (s_dex4 == 0 && d_for == true)
336         {
337             b_for = false;
338             d_for = false;
339             a_for = true;
340             H_For(s_yuju, xmldoc, xmlelem_main, index_main);
341 
342         }
343     }
344     #endregion
345 
346     #region do whileループ
347     if (a_if && IsMatch && a_do && a_for && a_if && a_switch && a_while)
348     {
349         if (s == "do")
350         {
351             b_do = true;
352             a_do = false;
353             IsMatch = false;
354         }
355 
356     }
357     if (b_do)
358     {
359         IsMatch = false;
360         s_yuju += "\r\n" + s;
361         if (s == "{")
362         {
363             d_do = true;
364             s_dex4++;
365         }
366         if (s == "}")
367         {
368             s_dex4--;
369         }
370         if (s_dex4 == 0 && d_do == true)
371         {
372             if (s_dex < slist.Length - 1)
373             {
374                 if (slist[s_dex + 1].IndexOf("while") > -1)
375                 {
376 
377                 }
378                 else
379                 {
```

図 6.17: CProgramsXML.cs（7）

```
                    b_do = false;
                    d_do = false;
                    a_do = true;
                    IsMatch = true;
                    H_Do(s_yuju, xmldoc, xmlelem_main, index_main);

                }
            }
            else
            {
                b_do = false;
                d_do = false;
                a_do = true;
                IsMatch = true;
                H_Do(s_yuju, xmldoc, xmlelem_main, index_main);

            }

        }
    }
#endregion

#region switch ループ
if (a_if && IsMatch && a_do && a_for && a_if && a_switch && a_while)
{
    Regex r_switch = new Regex(@"switch\s*\(.+\)");//函数定義
    MatchCollection ms_switch = r_switch.Matches(s);
    if (ms_switch.Count > 0)
    {
        b_switch = true;
        s_yuju = "";
        s_dex4 = 0;
        a_switch = false;
        IsMatch = false;
    }
}
if (b_switch)
{
    IsMatch = false;
    s_yuju += "\r\n" + s;
    if (s == "{")
    {
        d_switch = true;
        s_dex4++;
    }
    if (s == "}")
    {
        s_dex4--;
    }
    if (s_dex4 == 0 && d_switch == true)
    {
        b_switch = false;
        d_switch = false;
        a_switch = true;
```

図 6.18: CProgramsXML.cs（8）

```
                                H_Switch(s_yuju, xmldoc, xmlelem_main, index_main);

                            }
                        }
                        #endregion

                        #region パラメータ定義
                        if (s_yuju == "" && a_if && IsMatch && a_do && a_for && a_if && a_switch && a_while)
                        {
                            Regex r1 = new Regex(@"^(void|int|string|short|long|float|double|char|struct|union|enum)\s+.
                            MatchCollection ms1 = r1.Matches(s);
                            if (ms1.Count > 0)
                            {
                                IsMatch = false;
                                H_DingYi(s, xmldoc, xmlelem_main, index_main);

                            }
                        }
                        #endregion

                        #region
                        if (a_if && IsMatch && a_do && a_for && a_if && a_switch && a_while)
                        {
                            H_FuZhi(s, xmldoc, xmlelem_main, index_main);
                            s_yuju = "";
                        }
                        #endregion

                        s_dex++;
                        IsMatch = true;
                    }
                    #endregion
                }

                #endregion

                XML(xmldoc);
                MessageBox.Show("Successful Conversion！");
            }
            catch (Exception ex2)
            {
                MessageBox.Show("Conversion Failed: " + ex2.Message);
            }


        }

    }
}
catch (Exception ex)
{
    MessageBox.Show("Conversion Failed: " + ex.Message);
}
```

図 6.19: CProgramsXML.cs（9）

```
488         }
489         #endregion
490     }
491
492     public void BianLiJieDian(XmlNode oNode)
493     {
494         //XmlNodeList oList = oNode.ChildNodes;             //
495
496         int j = 0;
497         for (int i = 0; i < oNode.ChildNodes.Count; i++)
498         {
499             if (oNode.ChildNodes[i] is XmlElement)
500             {
501                 j += 1;
502                 BianLiJieDian(oNode.ChildNodes[i]);
503             }
504         }
505
506         ((XmlElement)oNode).SetAttribute("numofchildren", j.ToString());
507     }
508     public void CheckYuJu(string resources, XmlDocument xmldoc, XmlElement xmlelem_main, int index_main)
509     {
510         try
511         {
512             string[] slist = resources.Split(new string[] { "\r\n" }, StringSplitOptions.RemoveEmptyEntries);
513             #region
514             int s_dex = 0;
515             bool IsMatch = true;
516             bool b_if = false;
517             bool a_if = true;
518             bool d_if = false;
519             bool b_while = false;
520             bool d_while = false;
521             bool a_while = true;
522             bool b_switch = false;
523             bool d_switch = false;
524             bool a_switch = true;
525             bool b_for = false;
526             bool d_for = false;
527             bool a_for = true;
528             bool b_do = false;
529             bool d_do = false;
530             bool a_do = true;
531             int s_dex4 = 0;
532             string s_yuju = "";
533
534             foreach (string s in slist)
535             {
536
537                 #region if
538                 if (a_if && IsMatch && a_do && a_for && a_if && a_switch && a_while)
539                 {
540                     Regex r2 = new Regex(@"if\s*\(.+\)");//if
541                     Regex r3 = new Regex(@"else if\(.+?\)");//if
```

図 6.20: CProgramsXML.cs（10）

```
460                                                 }
461                                                 #endregion
462
463                                                 s_dex++;
464                                                 IsMatch = true;
465                                             }
466                                             #endregion
467                                         }
468
469                                         #endregion
470
471                                         XML(xmldoc);
472                                         MessageBox.Show("Successful Conversion！");
473                                     }
474                                     catch (Exception ex2)
475                                     {
476                                         MessageBox.Show("Conversion Failed: " + ex2.Message);
477                                     }
478
479
480                                 }
481
482                             }
483                         }
484                     }
485                     catch (Exception ex)
486                     {
487                         MessageBox.Show("Conversion Failed: " + ex.Message);
488                     }
489                 }
490                 #endregion
491         }
492
493         public void BianLiJieDian(XmlNode oNode)
494         {
495             //XmlNodeList oList = oNode.ChildNodes;             //
496
497             int j = 0;
498             for (int i = 0; i < oNode.ChildNodes.Count; i++)
499             {
500                 if (oNode.ChildNodes[i] is XmlElement)
501                 {
502                     j += 1;
503                     BianLiJieDian(oNode.ChildNodes[i]);
504                 }
505             }
506
507             ((XmlElement)oNode).SetAttribute("numofchildren", j.ToString());
508         }
509         public void CheckYuJu(string resources, XmlDocument xmldoc, XmlElement xmlelem_main, int index_main)
510         {
```

図 6.21: CProgramsXML.cs（11）

```
651             IsMatch = false;
652             s_yuju += "\r\n" + s;
653             if (s == "{")
654             {
655                 d_for = true;
656                 s_dex4++;
657             }
658             if (s == "}")
659             {
660                 s_dex4--;
661             }
662             if (s_dex4 == 0 && d_for == true)
663             {
664                 b_for = false;
665                 d_for = false;
666                 a_for = true;
667                 H_For(s_yuju, xmldoc, xmlelem_main, index_main);
668 
669             }
670         }
671         #endregion
672 
673         #region do while
674         if (a_if && IsMatch && a_do && a_for && a_if && a_switch && a_while)
675         {
676             if (s == "do")
677             {
678                 b_do = true;
679                 a_do = false;
680                 IsMatch = false;
681             }
682 
683         }
684         if (b_do)
685         {
686             IsMatch = false;
687             s_yuju += "\r\n" + s;
688             if (s == "{")
689             {
690                 d_do = true;
691                 s_dex4++;
692             }
693             if (s == "}")
694             {
695                 s_dex4--;
696             }
697             if (s_dex4 == 0 && d_do == true)
698             {
699                 if (s_dex < slist.Length - 1)
700                 {
701                     if (slist[s_dex + 1].IndexOf("while") > -1)
702                     {
703 
704                     }
```

図 6.22: CProgramsXML.cs（12）

```
705                 else
706                 {
707                     b_do = false;
708                     d_do = false;
709                     a_do = true;
710                     IsMatch = true;
711                     H_Do(s_yuju, xmldoc, xmlelem_main, index_main);
712 
713                 }
714             }
715             else
716             {
717                 b_do = false;
718                 d_do = false;
719                 a_do = true;
720                 IsMatch = true;
721                 H_Do(s_yuju, xmldoc, xmlelem_main, index_main);
722 
723             }
724 
725         }
726     }
727     #endregion
728 
729     #region switch
730     if (a_if && IsMatch && a_do && a_for && a_if && a_switch && a_while)
731     {
732         Regex r_switch = new Regex(@"switch\s*\(.+\)");//
733         MatchCollection ms_switch = r_switch.Matches(s);
734         if (ms_switch.Count > 0)
735         {
736             b_switch = true;
737             s_yuju = "";
738             s_dex4 = 0;
739             a_switch = false;
740             IsMatch = false;
741         }
742     }
743     if (b_switch)
744     {
745         IsMatch = false;
746         s_yuju += "\r\n" + s;
747         if (s == "{")
748         {
749             d_switch = true;
750             s_dex4++;
751         }
752         if (s == "}")
753         {
754             s_dex4--;
755         }
756         if (s_dex4 == 0 && d_switch == true)
757         {
758             b_switch = false;
```

図 6.23: CProgramsXML.cs（13）

```
759                     d_switch = false;
760                     a_switch = true;
761                     H_Switch(s_yuju, xmldoc, xmlelem_main, index_main);
762 
763                 }
764             }
765             #endregion
766 
767             #region パラメータ定義
768             if (s_yuju == "" && a_if && IsMatch && a_do && a_for && a_if && a_switch && a_while)
769             {
770                 Regex r1 = new Regex(@"^(void|int|string|short|long|float|double|char|struct|union|enum)\s+.+");
771                 MatchCollection ms1 = r1.Matches(s);
772                 if (ms1.Count > 0)
773                 {
774                     IsMatch = false;
775                     H_DingYi(s, xmldoc, xmlelem_main, index_main);
776 
777                 }
778             }
779             #endregion
780 
781             #region
782             if (a_if && IsMatch && a_do && a_for && a_if && a_switch && a_while)
783             {
784                 H_FuZhi(s, xmldoc, xmlelem_main, index_main);
785                 s_yuju = "";
786             }
787             #endregion
788 
789             s_dex++;
790             IsMatch = true;
791         }
792         #endregion
793     }
794     catch (Exception ex)
795     {
796         MessageBox.Show("Conversion Failed:, " + ex.Message);
797         return;
798     }
799 }
800 
801 /// <summary>
802 /// (たとえa==3;)
803 /// </summary>
804 /// <param name="resources"></param>
805 /// <param name="xmldoc"></param>
806 /// <param name="XmlElen"></param>
807 /// <param name="L_index"></param>
808 public void H_FuZhi(string resources, XmlDocument xmldoc, XmlElement XmlElen, int L_index)
809 {
810     try
811     {
812         //string pattern = @"(?<na>[A-Za-z0-9/_]+)(?<na2>\(.*?\))";
```

図 6.24: CProgramsXML.cs（14）

```
813             //string substitution = "λ${na2}";
814             //string d3 = Regex.Replace(resources, pattern, substitution);
815 
816             XmlElement X1 = xmldoc.CreateElement("funcContents");
817             X1.SetAttribute("level", L_index.ToString());
818             X1.SetAttribute("label", label.ToString());
819             X1.SetAttribute("text", resources);
820             X1.InnerText = resources;
821             XmlElen.AppendChild(X1);
822             label++;
823 
824             #region
825             Regex r1 = new Regex(@"([A-Za-z0-9/_]+)\(.*?\)");
826             MatchCollection ms1 = r1.Matches(resources);
827             if (ms1.Count > 0)
828             {
829                 foreach (Match m in ms1)
830                 {
831                     //for (int i = 1; i < m.Groups.Count; i++)
832 
833                     H_HanShu(m.Groups[1].Value, xmldoc, X1, L_index );
834 
835                 }
836 
837             }
838             #endregion
839         }
840         catch (Exception ex)
841         {
842             MessageBox.Show("Conversion Failed:, " + ex.Message);
843             return;
844         }
845     }
846     /// <summary>
847     /// if
848     /// </summary>
849     /// <param name="XmlElen"></param>
850     public void H_If(string resources, XmlDocument xmldoc, XmlElement XmlElen, int L_index)
851     {
852         try
853         {
854             string[] slist = resources.Split(new string[] { "\r\n" }, StringSplitOptions.RemoveEmptyEntries);
855             bool b_if = false;
856             bool d_if = false;
857             bool b_else = false;
858             bool d_else = false;
859             bool b_elseif = false;
860             bool d_elseif = false;
861             int s_if = 0;
862             int s_else = 0;
863             int s_elseif = 0;
864             int L_index2 = L_index;
865             string s_yuju = "";
866             XmlElement x_old;
```

図 6.25: CProgramsXML.cs（15）

```
867     x_old = XmlElen;
868     foreach (string s in slist)
869     {
870         #region if判別
871         if (s_if == 0 && s_else == 0 && s_elseif == 0)
872         {
873             List<string> list_if = new List<string>();
874             Regex r_checkelseif = new Regex(@"else if\(.+?\)");//if
875             if (!r_checkelseif.IsMatch(s))
876             {
877                 Regex r_if = new Regex(@"(if)\s*\((.+)\)");
878                 MatchCollection ms_if = r_if.Matches(s);
879                 if (ms_if.Count > 0)
880                 {
881                     b_if = true;
882                     s_yuju = "";
883                     foreach (Match m in ms_if)
884                     {
885                         for (int i = 1; i < m.Groups.Count; i++)
886                         {
887                             list_if.Add(m.Groups[i].Value);
888                         }
889                     }
890                     L_index2++;
891                     XmlElement X1 = xmldoc.CreateElement("funcContents");
892                     X1.SetAttribute("level", L_index2.ToString());
893                     X1.SetAttribute("label", label.ToString());
894                     X1.SetAttribute("text", list_if[0]);
895                     X1.InnerText = list_if[0];
896                     x_old.AppendChild(X1);
897                     x_old = X1;
898                     L_index2++;
899                     label++;
900
901                     //string pattern = @"(?<na>[A-Za-z0-9/_]+)(?<na2>\(.*?\))";
902                     //string substitution = "A${na2}";
903                     //string d3 = Regex.Replace(list_if[1], pattern, substitution);
904
905                     XmlElement X2 = xmldoc.CreateElement("funcContents");
906                     X2.SetAttribute("level", L_index2.ToString());
907                     X2.SetAttribute("label", label.ToString());
908                     X2.SetAttribute("text", list_if[1]);
909                     X2.InnerText = list_if[1];
910                     x_old.AppendChild(X2);
911                     x_old = X2;
912                     L_index2++;
913                     label++;
914                     //if(自作定義方法)-------------
915                     #region 自
916                     Regex r1 = new Regex(@"([A-Za-z0-9/_]+)\(.*?\)");
917                     MatchCollection ms1 = r1.Matches(list_if[1]);
918                     if (ms1.Count > 0)
919                     {
920                         foreach (Match m in ms1)
```

図 6.26: CProgramsXML.cs（16）

```
921                 {
922                     //H_HanShu(m.Groups[1].Value, xmldoc, x_old, L_index2);
923                     YuJu_HanShu(list_if[1], xmldoc, x_old, L_index2);
924                 }
925 
926             }
927             #endregion
928         }
929     }
930 }
931 if (b_if)
932 {
933     if (s == "{")
934     {
935         d_if = true;
936         s_if++;
937     }
938     if (s == "}")
939     {
940         s_if--;
941     }
942     if (d_if == true)
943     {
944         s_yuju += "\r\n" + s;
945     }
946     if (s_if == 0 && d_if == true)
947     {
948         b_if = false;
949         s_yuju = s_yuju.Substring(3, s_yuju.Length - 4);
950         XmlElement X3 = xmldoc.CreateElement("funcContents");
951         X3.SetAttribute("level", L_index2.ToString());
952         X3.SetAttribute("label", label.ToString());
953         X3.SetAttribute("text", "Yes");
954         X3.InnerText = "Yes";
955         x_old.AppendChild(X3);
956         label++;
957 
958         CheckYuJu(s_yuju, xmldoc, X3, L_index2 + 1);
959     }
960 }
961 #endregion
962 
963 #region else判別
964 if (s_if == 0 && s_else == 0 && s_elseif == 0)
965 {
966     if (s.Trim() == "else")
967     {
968         b_else = true;
969         s_yuju = "";
970     }
971 }
972 if (b_else)
973 {
974     if (s == "{")
```

図 6.27: CProgramsXML.cs（17）

```
975         {
976             d_else = true;
977             s_else++;
978         }
979         if (s == "}")
980         {
981             s_else--;
982         }
983         if (d_else == true)
984         {
985             s_yuju += "\r\n" + s;
986         }
987         if (s_else == 0 && d_else == true)
988         {
989             b_else = false;
990             s_yuju = s_yuju.Substring(3, s_yuju.Length - 4);
991             XmlElement X3 = xmldoc.CreateElement("funcContents");
992             X3.SetAttribute("level", L_index2.ToString());
993             X3.SetAttribute("label", label.ToString());
994             X3.SetAttribute("text", "NO");
995             X3.InnerText = "NO";
996             x_old.AppendChild(X3);
997             label++;
998             CheckYuJu(s_yuju, xmldoc, X3, L_index2 + 1);
999         }
000     }
001     #endregion
002
003     #region else if判別
004     if (s_if == 0 && s_else == 0 && s_elseif == 0)
005     {
006         List<string> list_elseif = new List<string>();
007         Regex r_elseif = new Regex(@"(else if)\((.+?)\)");
008         MatchCollection ms_elseif = r_elseif.Matches(s);
009         if (ms_elseif.Count > 0)
010         {
011             b_elseif = true;
012             s_yuju = "";
013             foreach (Match m in ms_elseif)
014             {
015                 for (int i = 1; i < m.Groups.Count; i++)
016                 {
017                     list_elseif.Add(m.Groups[i].Value);
018                 }
019             }
020             XmlElement X11 = xmldoc.CreateElement("funcContents");
021             X11.SetAttribute("level", L_index2.ToString());
022             X11.SetAttribute("label", label.ToString());
023             X11.SetAttribute("text", "else if");
024             X11.InnerText = "else if";
025             x_old.AppendChild(X11);
026             L_index2++;
027             label++;
028
```

図 6.28: CProgramsXML.cs（18）

```
1029            //string pattern = @"(?<na>[A-Za-z0-9/_]+)(?<na2>\(.*?\))";
1030            //string substitution = "λ${na2}";
1031            //string d3 = Regex.Replace(list_elseif[1], pattern, substitution);
1032
1033            XmlElement X12 = xmldoc.CreateElement("funcContents");
1034            X12.SetAttribute("level", L_index2.ToString());
1035            X12.SetAttribute("label", label.ToString());
1036            X12.SetAttribute("text", list_elseif[1]);
1037            X12.InnerText = list_elseif[1];
1038            X11.AppendChild(X12);
1039            x_old = X12;
1040            L_index2++;
1041            label++;
1042            //if()-------------2013-12-18
1043            #region
1044            Regex r1 = new Regex(@"([A-Za-z0-9/_]+)\(.*?\)");
1045            MatchCollection ms1 = r1.Matches(list_elseif[1]);
1046            if (ms1.Count > 0)
1047            {
1048                foreach (Match m in ms1)
1049                {
1050                    //H_HanShu(m.Groups[1].Value, xmldoc, x_old, L_index2);
1051                    YuJu_HanShu(list_elseif[1], xmldoc, x_old, L_index2);
1052                }
1053
1054            }
1055            #endregion
1056
1057
1058        }
1059    }
1060    if (b_elseif)
1061    {
1062        if (s == "{")
1063        {
1064            d_elseif = true;
1065            s_elseif++;
1066        }
1067        if (s == "}")
1068        {
1069            s_elseif--;
1070        }
1071        if (d_elseif == true)
1072        {
1073            s_yuju += "\r\n" + s;
1074        }
1075        if (s_elseif == 0 && d_elseif == true)
1076        {
1077            b_elseif = false;
1078            s_yuju = s_yuju.Substring(3, s_yuju.Length - 4);
1079            XmlElement X3 = xmldoc.CreateElement("funcContents");
1080            X3.SetAttribute("level", L_index2.ToString());
1081            X3.SetAttribute("label", label.ToString());
1082            X3.SetAttribute("text", "Yes");
```

図 6.29: CProgramsXML.cs（19）

```
                    if (ms1.Count > 0)
                    {
                        foreach (Match m in ms1)
                        {
                            //H_HanShu(m.Groups[1].Value, xmldoc, x_old, L_index2);
                            YuJu_HanShu(list_if[1], xmldoc, x_old, L_index2);
                        }

                    }
                    #endregion
                }
            }
        }
        if (b_if)
        {
            if (s == "{")
            {
                d_if = true;
                s_if++;
            }
            if (s == "}")
            {
                s_if--;
            }
            if (d_if == true)
            {
                s_yuju += "\r\n" + s;
            }
            if (s_if == 0 && d_if == true)
            {
                b_if = false;
                s_yuju = s_yuju.Substring(3, s_yuju.Length - 4);
                XmlElement X3 = xmldoc.CreateElement("funcContents");
                X3.SetAttribute("level", L_index2.ToString());
                X3.SetAttribute("label", label.ToString());
                X3.SetAttribute("text", "Yes");
                X3.InnerText = "Yes";
                x_old.AppendChild(X3);
                label++;

                CheckYuJu(s_yuju, xmldoc, X3, L_index2 + 1);
            }
        }
        #endregion

        #region else判別
        if (s_if == 0 && s_else == 0 && s_elseif == 0)
        {
            if (s.Trim() == "else")
            {
                b_else = true;
```

図 6.30: CProgramsXML.cs（20）

```
970                         s_yuju = "";
971                     }
972                 }
973                 if (b_else)
974                 {
975                     if (s == "{")
976                     {
977                         d_else = true;
978                         s_else++;
979                     }
980                     if (s == "}")
981                     {
982                         s_else--;
983                     }
984                     if (d_else == true)
985                     {
986                         s_yuju += "\r\n" + s;
987                     }
988                     if (s_else == 0 && d_else == true)
989                     {
990                         b_else = false;
991                         s_yuju = s_yuju.Substring(3, s_yuju.Length - 4);
992                         XmlElement X3 = xmldoc.CreateElement("funcContents");
993                         X3.SetAttribute("level", L_index2.ToString());
994                         X3.SetAttribute("label", label.ToString());
995                         X3.SetAttribute("text", "NO");
996                         X3.InnerText = "NO";
997                         x_old.AppendChild(X3);
998                         label++;
999                         CheckYuJu(s_yuju, xmldoc, X3, L_index2 + 1);
1000                    }
1001                }
1002                #endregion
1003
1004                #region else if判別
1005                if (s_if == 0 && s_else == 0 && s_elseif == 0)
1006                {
1007                    List<string> list_elseif = new List<string>();
1008                    Regex r_elseif = new Regex(@"(else if)\((.+?)\)");
1009                    MatchCollection ms_elseif = r_elseif.Matches(s);
1010                    if (ms_elseif.Count > 0)
1011                    {
1012                        b_elseif = true;
1013                        s_yuju = "";
1014                        foreach (Match m in ms_elseif)
1015                        {
1016                            for (int i = 1; i < m.Groups.Count; i++)
1017                            {
1018                                list_elseif.Add(m.Groups[i].Value);
1019                            }
1020                        }
```

図 6.31: CProgramsXML.cs（21）

```
                    X3.InnerText = "Yes";
                    x_old.AppendChild(X3);
                    label++;
                    CheckYuJu(s_yuju, xmldoc, X3, L_index2 + 1);
                }
            }
            #endregion
        }


    }
    catch (Exception ex)
    {
        MessageBox.Show("Conversion Failed:, " + ex.Message);
        return;
    }
}

public void H_While(string resources, XmlDocument xmldoc, XmlElement XmlElen, int L_index)
{
    try
    {
        string[] slist = resources.Split(new string[] { "\r\n" }, StringSplitOptions.RemoveEmptyEntries);
        bool b_while = false;
        bool d_while = false;
        bool a_while = true;
        int s_while = 0;
        string s_yuju = "";
        List<string> list_while = new List<string>();
        XmlElement x_old = XmlElen;
        foreach (string s in slist)
        {
            if (a_while)
            {
                Regex r_while = new Regex(@"(while)\s*\((.+)\)");
                MatchCollection ms_while = r_while.Matches(s);
                if (ms_while.Count > 0)
                {
                    b_while = true;
                    s_yuju = "";
                    a_while = false;
                    foreach (Match m in ms_while)
                    {
                        for (int i = 1; i < m.Groups.Count; i++)
                        {
                            list_while.Add(m.Groups[i].Value);
                        }
                    }
                    XmlElement X1 = xmldoc.CreateElement("funcContents");
                    X1.SetAttribute("level", L_index.ToString());
                    X1.SetAttribute("label", label.ToString());
                    X1.SetAttribute("text", list_while[0]);
                    X1.InnerText = list_while[0];
                    x_old.AppendChild(X1);
```

図 6.32: CProgramsXML.cs（22）

```
1137                        x_old = X1;
1138                        label++;
1139
1140                        XmlElement X2 = xmldoc.CreateElement("funcContents");
1141                        X2.SetAttribute("level", (L_index + 1).ToString());
1142                        X2.SetAttribute("label", label.ToString());
1143                        X2.SetAttribute("text", list_while[1]);
1144                        X2.InnerText = list_while[1];
1145                        x_old.AppendChild(X2);
1146                        x_old = X2;
1147                        label++;
1148                    }
1149                }
1150                if (b_while)
1151                {
1152                    if (s == "{")
1153                    {
1154                        d_while = true;
1155                        s_while++;
1156                    }
1157                    if (s == "}")
1158                    {
1159                        s_while--;
1160                    }
1161                    if (d_while == true)
1162                    {
1163                        s_yuju += "\r\n" + s;
1164                    }
1165                    if (s_while == 0 && d_while == true)
1166                    {
1167                        b_while = false;
1168                        d_while = false;
1169                        s_yuju = s_yuju.Substring(3, s_yuju.Length - 4);
1170                        CheckYuJu(s_yuju, xmldoc, x_old, L_index + 2);
1171                    }
1172                }
1173            }
1174        }
1175        catch (Exception ex)
1176        {
1177            MessageBox.Show("Conversion Failed:, " + ex.Message);
1178            return;
1179
1180        }
1181    }
1182
1183    public void H_For(string resources, XmlDocument xmldoc, XmlElement XmlElen, int L_index)
1184    {
1185        try
1186        {
1187            string[] slist = resources.Split(new string[] { "\r\n" }, StringSplitOptions.RemoveEmptyEntries);
1188            bool b_for = false;
1189            bool d_for = false;
1190            bool a_for = true;
```

図 6.33: CProgramsXML.cs（23）

```
1191    int s_for = 0;
1192    string s_yuju = "";
1193    List<string> list_for = new List<string>();
1194    XmlElement x_old = XmlElen;
1195    foreach (string s in slist)
1196    {
1197        if (a_for)
1198        {
1199            Regex r_while = new Regex(@"(for)\s*\((.+)\)");
1200            MatchCollection ms_while = r_while.Matches(s);
1201            if (ms_while.Count > 0)
1202            {
1203                b_for = true;
1204                s_yuju = "";
1205                a_for = false;
1206                foreach (Match m in ms_while)
1207                {
1208                    for (int i = 1; i < m.Groups.Count; i++)
1209                    {
1210                        list_for.Add(m.Groups[i].Value);
1211                    }
1212                }
1213                XmlElement X1 = xmldoc.CreateElement("funcContents");
1214                X1.SetAttribute("level", L_index.ToString());
1215                X1.SetAttribute("label", label.ToString());
1216                X1.SetAttribute("text", list_for[0]);
1217                X1.InnerText = list_for[0];
1218                x_old.AppendChild(X1);
1219                x_old = X1;
1220                label++;
1221
1222                XmlElement X2 = xmldoc.CreateElement("funcContents");
1223                X2.SetAttribute("level", (L_index + 1).ToString());
1224                X2.SetAttribute("label", label.ToString());
1225                X2.SetAttribute("text", list_for[1]);
1226                X2.InnerText = list_for[1];
1227                x_old.AppendChild(X2);
1228                x_old = X2;
1229                label++;
1230            }
1231        }
1232        if (b_for)
1233        {
1234            if (s == "{")
1235            {
1236                d_for = true;
1237                s_for++;
1238            }
1239            if (s == "}")
1240            {
1241                s_for--;
1242            }
1243            if (d_for == true)
1244            {
```

図 6.34: CProgramsXML.cs（24）

```
1299                }
1300                XmlElement X1 = xmldoc.CreateElement("funcContents");
1301                X1.SetAttribute("level", (L_index++).ToString());
1302                X1.SetAttribute("label", label.ToString());
1303                X1.SetAttribute("text", list_do[0]);
1304                X1.InnerText = list_do[0];
1305                x_old.AppendChild(X1);
1306                x_old = X1;
1307                label++;
1308
1309                XmlElement X2 = xmldoc.CreateElement("funcContents");
1310                X2.SetAttribute("level", (L_index++).ToString());
1311                X2.SetAttribute("label", label.ToString());
1312                X2.SetAttribute("text", list_do[1]);
1313                X2.InnerText = list_do[1];
1314                x_old.AppendChild(X2);
1315                x_old = X2;
1316                label++;
1317            }
1318        }
1319        for (int c = 1; c < slist.Length - 1; c++)
1320        {
1321            string s = slist[c];
1322            if (b_do)
1323            {
1324                if (s == "{")
1325                {
1326                    d_do = true;
1327                    s_do++;
1328                }
1329                if (s == "}")
1330                {
1331                    s_do--;
1332                }
1333                if (d_do == true)
1334                {
1335                    s_yuju += "\r\n" + s;
1336                }
1337                if (s_do == 0 && d_do == true)
1338                {
1339                    b_do = false;
1340                    d_do = false;
1341                    s_yuju = s_yuju.Substring(3, s_yuju.Length - 4);
1342                    CheckYuJu(s_yuju, xmldoc, x_old, L_index);
1343                }
1344            }
1345        }
1346    }
1347    catch (Exception ex)
1348    {
1349        MessageBox.Show("Conversion Failed:, " + ex.Message);
1350        return;
1351    }
1352 }
```

図 6.35: CProgramsXML.cs（25）

```
1353
1354    public void H_Switch(string resources, XmlDocument xmldoc, XmlElement XmlElen, int L_index)
1355    {
1356        try
1357        {
1358            string[] slist = resources.Split(new string[] { "\r\n" }, StringSplitOptions.RemoveEmptyEntries);
1359            bool b_while = false;
1360            bool d_while = false;
1361            bool a_while = true;
1362            int s_while = 0;
1363            string s_yuju = "";
1364            List<string> list_while = new List<string>();
1365            XmlElement x_old = XmlElen;
1366            XmlElement x_case = XmlElen;
1367            foreach (string s in slist)
1368            {
1369                if (a_while)
1370                {
1371                    Regex r_while = new Regex(@"(switch)\s*\((.+)\)");
1372                    MatchCollection ms_while = r_while.Matches(s);
1373                    if (ms_while.Count > 0)
1374                    {
1375                        b_while = true;
1376                        s_yuju = "";
1377                        a_while = false;
1378                        foreach (Match m in ms_while)
1379                        {
1380                            for (int i = 1; i < m.Groups.Count; i++)
1381                            {
1382                                list_while.Add(m.Groups[i].Value);
1383                            }
1384                        }
1385                        XmlElement X1 = xmldoc.CreateElement("funcContents");
1386                        X1.SetAttribute("level", (L_index++).ToString());
1387                        X1.SetAttribute("label", label.ToString());
1388                        X1.SetAttribute("text", list_while[0]);
1389                        X1.InnerText = list_while[0];
1390                        x_old.AppendChild(X1);
1391                        x_old = X1;
1392                        label++;
1393
1394                        XmlElement X2 = xmldoc.CreateElement("funcContents");
1395                        X2.SetAttribute("level", (L_index++).ToString());
1396                        X2.SetAttribute("label", label.ToString());
1397                        X2.SetAttribute("text", list_while[1]);
1398                        X2.InnerText = list_while[1];
1399                        x_old.AppendChild(X2);
1400                        x_old = X2;
1401                        label++;
1402                    }
1403                }
1404                if (b_while)
1405                {
1406                    Regex r = new Regex(@"(case)\s(.+)");
```

図 6.36: CProgramsXML.cs（26）

```
1407    MatchCollection ms = r.Matches(s);
1408    List<string> list_case = new List<string>();
1409    if (ms.Count > 0)
1410    {
1411        d_while = true;
1412        s_yuju = "";
1413        foreach (Match m in ms)
1414        {
1415            for (int i = 1; i < m.Groups.Count; i++)
1416            {
1417                list_case.Add(m.Groups[i].Value);
1418            }
1419        }
1420        XmlElement X1 = xmldoc.CreateElement("funcContents");
1421        X1.SetAttribute("level", L_index.ToString());
1422        X1.SetAttribute("label", label.ToString());
1423        X1.SetAttribute("text", list_case[1]);
1424        X1.InnerText = list_case[1];
1425        x_old.AppendChild(X1);
1426        x_case = X1;
1427        label++;
1428
1429    }
1430    Regex r_default = new Regex(@"(default)");
1431    MatchCollection ms_default = r_default.Matches(s);
1432    if (ms_default.Count > 0)
1433    {
1434        d_while = true;
1435        s_yuju = "";
1436        XmlElement X1 = xmldoc.CreateElement("funcContents");
1437        X1.SetAttribute("level", L_index.ToString());
1438        X1.SetAttribute("label", label.ToString());
1439        X1.SetAttribute("text", "default");
1440        X1.InnerText = "default";
1441        x_old.AppendChild(X1);
1442        x_case = X1;
1443        label++;
1444
1445    }
1446    if (d_while)
1447    {
1448        if (s.IndexOf("case ") <= -1 && s.IndexOf("default ") <= -1 && s_while < slist.Length - 1)
1449        {
1450            s_yuju += "\r\n" + s;
1451        }
1452        if (s_while < slist.Length - 1)
1453        {
1454            if (slist[s_while + 1].IndexOf("case ") > -1 || slist[s_while + 1].IndexOf("default ") > -1)
1455            {
1456
1457                d_while = false;
1458                CheckYuJu(s_yuju, xmldoc, x_case, L_index + 1);
1459            }
1460        }
```

図 6.37: CProgramsXML.cs（27）

```
                    else
                    {

                        d_while = false;
                        CheckYuJu(s_yuju, xmldoc, x_case, L_index + 1);
                    }

                }
            }
            s_while++;
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Conversion Failed:, " + ex.Message);
        return;
    }
}

public void H_DingYi(string resources, XmlDocument xmldoc, XmlElement XmlElen, int L_index)
{
    try
    {
        List<string> l1 = new List<string>();
        Regex r2 = new Regex(@"^(void|int|string|short|long|float|double|char|struct|union|enum)\s+(.+)");
        MatchCollection ms2 = r2.Matches(resources);
        if (ms2.Count > 0)
        {
            foreach (Match m in ms2)
            {
                for (int i = 1; i < m.Groups.Count; i++)
                {
                    l1.Add(m.Groups[i].Value);
                }
            }
            XmlElement Xold = XmlElen;
            foreach (string s in l1)
            {
                XmlElement X1 = xmldoc.CreateElement("funcContents");
                X1.SetAttribute("level", L_index.ToString());
                X1.SetAttribute("label", label.ToString());
                X1.SetAttribute("text", s);
                X1.InnerText = s;
                Xold.AppendChild(X1);
                Xold = X1;
                L_index++;
                label++;
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Conversion Failed:, " + ex.Message);
        return;
```

図 6.38: CProgramsXML.cs（28）

```
1515        }
1516    }
1517
1518    public void H_HanShu(string resources, XmlDocument xmldoc, XmlElement XmlElen, int L_index)
1519    {
1520        try
1521        {
1522            int i = 0;
1523            XmlElement X1 = XmlElen;
1524            if (!list_history.Contains(resources))
1525            {
1526                foreach (string s in l1)
1527                {
1528                    if (s.IndexOf(resources)> -1)
1529                    {
1530                        if (l2[i][1] == resources)
1531                        {
1532                            list_history.Add(resources);
1533                            int c = 0;
1534                            foreach (string s1 in l2[i])
1535                            {
1536                                L_index++;
1537                                XmlElement xmlelem = xmldoc.CreateElement("funcContents");
1538                                xmlelem.SetAttribute("level", L_index.ToString());
1539                                xmlelem.SetAttribute("label", label.ToString());
1540                                //if (c == 1)
1541                                //{
1542                                //    xmlelem.SetAttribute("text", "λ");
1543                                //    xmlelem.InnerText = "λ";
1544                                //}
1545                                //else
1546                                //{
1547                                    xmlelem.SetAttribute("text", s1);
1548                                    xmlelem.InnerText = s1;
1549                                //}
1550                                X1.AppendChild(xmlelem);
1551                                X1 = xmlelem;
1552                                label++;
1553                                c++;
1554                            }
1555                            string content = List_content[i];
1556                            Regex r_content = new Regex(@"\{([\s\S]*)\}");
1557                            MatchCollection ms_content = r_content.Matches(content);
1558                            if (ms_content.Count > 0)
1559                            {
1560                                CheckYuJu(ms_content[0].Groups[1].Value, xmldoc, X1, L_index++);
1561                            }
1562                        }
1563                    }
1564                    i++;
1565                }
1566            }
1567        }
1568        catch (Exception ex)
```

図 6.39: CProgramsXML.cs（29）

```
1569        {
1570            MessageBox.Show("Conversion Failed:, " + ex.Message);
1571            return;
1572        }
1573    }
1574    public void YuJu_HanShu(string resources, XmlDocument xmldoc, XmlElement XmlElen, int L_index)
1575    {
1576        string H_name = "";
1577        try
1578        {
1579            Regex r1 = new Regex(@"([A-Za-z0-9/_]+)\(.*?\)");
1580            MatchCollection ms1 = r1.Matches(resources);
1581            if (ms1.Count > 0)
1582            {
1583                foreach (Match m in ms1)
1584                {
1585                    H_name = m.Groups[1].Value;
1586                }
1587
1588            }
1589            int i = 0;
1590            XmlElement X1 = XmlElen;
1591            if (!list_history.Contains(H_name))
1592            {
1593                foreach (string s in l1)
1594                {
1595                    if (s.IndexOf(H_name) > -1)
1596                    {
1597                        if (l2[i][1] == H_name)
1598                        {
1599                            //string pattern = @"(?<na>[A-Za-z0-9/_]+)(?<na2>\(.*?\))";
1600                            //string substitution = "λ${na2}";
1601                            //string d3 = Regex.Replace(resources, pattern, substitution);
1602
1603                            XmlElement X2 = xmldoc.CreateElement("funcContents");
1604                            X2.SetAttribute("level", L_index.ToString());
1605                            X2.SetAttribute("label", label.ToString());
1606                            X2.SetAttribute("text", resources);
1607                            X2.InnerText = resources;
1608                            X1.AppendChild(X2);
1609                            X1 = X2;
1610                            label++;
1611
1612                            list_history.Add(H_name);
1613                            int c = 0;
1614                            foreach (string s1 in l2[i])
1615                            {
1616                                L_index = L_index + 1;
1617                                XmlElement xmlelem = xmldoc.CreateElement("funcContents");
1618                                xmlelem.SetAttribute("level", L_index.ToString());
1619                                xmlelem.SetAttribute("label", label.ToString());
1620                                //if (c == 1)
1621                                //{
1622                                //    xmlelem.SetAttribute("text", "λ");
```

図 6.40: CProgramsXML.cs（30）

```
1623                                //    xmlelem.InnerText = "λ";
1624                                //}
1625                                //else
1626                                //{
1627                                    xmlelem.SetAttribute("text", s1);
1628                                    xmlelem.InnerText = s1;
1629                                //}
1630                                X1.AppendChild(xmlelem);
1631                                X1 = xmlelem;
1632                                label++;
1633                                c++;
1634                            }
1635                            string content = List_content[i];
1636                            Regex r_content = new Regex(@"\{([\s\S]*)\}");
1637                            MatchCollection ms_content = r_content.Matches(content);
1638                            if (ms_content.Count > 0)
1639                            {
1640                                CheckYuJu(ms_content[0].Groups[1].Value, xmldoc, X1, L_index++);
1641                            }
1642                        }
1643                    }
1644                    i++;
1645                }
1646            }
1647        }
1648        catch (Exception ex)
1649        {
1650            MessageBox.Show("Conversion Failed:, " + ex.Message);
1651            return;
1652        }
1653    }
1654
1655
1656
```

図 6.41: CProgramsXML.cs（31）

次は，NTO 手法のプログラムを示す．本プログラムの実行ファイルは $NTO-Method-XML.cs$ という名前のスクリプト CS ーファイルである．このプログラムは変換した二つの XML 構文木から共通部分木を求める作業で，まず CreatTable 関数では XML ファイルから各節点のデータが二つの table に取り出しておく作業である．table から二つのデータを取り出して比較操作を行う．IdenticalData 関数では二つの table から同じ部分を分析しほかの二つの table に取り出しておく作業である．FenXiTable 関数は子節点が一番多い優先に取り出す．PiPei 関数では dt1 の各行を基準にして，dt2 のすべて行と比較作業を行い，同じ子節点を見つけ出す作業である．DiGui2 関数では見つけ出した関連の節点を新たな table に保存する作業である．DataSetFuzhi 関数では最大のマッチした部分を見つけ出す作業である．

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Xml;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        public int a = 1;//XMLの各節点のIDループ一回で＋1;
        public int Num = 0;
        DataTable tb1 = new DataTable();//XML1保存
        DataTable tb2 = new DataTable();//XML2保存
        int NodeCount = 1;
        /// <summary>
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void btn_xml1_Click(object sender, EventArgs e)
        {
```

図 6.42: NTO-Method-XML.cs（1）

```
31          Stream myStream = null;
32          OpenFileDialog openFileDialog1 = new OpenFileDialog();
33          openFileDialog1.InitialDirectory = "E:\\Txt変換XML2\\Mark\\WindowsFormsApplication1\\WindowsFormsApplication1
34          openFileDialog1.Filter = "txt files (*.xml)|*.xml";
35          openFileDialog1.FilterIndex = 2;
36          openFileDialog1.RestoreDirectory = true;
37          if (openFileDialog1.ShowDialog() == DialogResult.OK)
38          {
39              try
40              {
41                  if ((myStream = openFileDialog1.OpenFile()) != null)
42                  {
43                      XmlDocument xmlDoc = new XmlDocument();
44                      xmlDoc.Load(openFileDialog1.FileName);//XML読み込む
45                      NodeCount = 1;
46                      tb1.Clear();
47                      AttributeID(xmlDoc.ChildNodes[1]);//まずXMLにIDつける.
48                      for (int i = 0; i <= 2; i++)
49                          //defuldでtableに三つの列追加，第一列節点の数保存，第二列に根節点が保存，第三根節点ID保存.
50                      {
51                          DataColumn dc = new DataColumn();
52                          tb1.Columns.Add(dc);
53                      }
54                      CreatTable(xmlDoc.ChildNodes[1], tb1);//XML1のデータがTableに追加
55                      a = 1;//前回ループが終わったら戻る.
56
57                     // xmlDoc.Save("C://bookstore.xml");
58                  }
59              }
60              catch (Exception ex)
61              {
62                  MessageBox.Show("Conversion Failed: " + ex.Message);
63              }
64          }
65      }
66      /// <summary>
67      /// XML節点に属性ID追加
68      /// </summary>
69      /// <param name="oNode"></param>
70      public void AttributeID(XmlNode oNode)
71      {
72          try
73          {
74              for (int i = 0; i < oNode.ChildNodes.Count; i++)
75              {
76                  if (oNode.ChildNodes[i] is XmlElement)
77                  {
78                      a += 1;
79                      NodeCount++;
80                      ((XmlElement)oNode.ChildNodes[i]).SetAttribute("ID", a.ToString());
81                      XmlElement xe = (XmlElement)oNode.ChildNodes[i];
82                      if (xe.ChildNodes.Count > 0)
83                      {
84                          AttributeID(oNode.ChildNodes[i]);
85                      }
```

図 6.43: NTO-Method-XML.cs（2）

```
86                  }
87              }
88          }
89          catch(Exception ex)
90          {
91              MessageBox.Show("Conversion Failed: " + ex.Message);
92          }
93      }
94      /// <summary>
95      /// 節点textとIDをTableに繰り返し追加方法
96      /// </summary>
97      /// <param name="oNode">XML各節点</param>
98      /// <param name="Dt">節点textとIDをTableに追加</param>
99      public void CreatTable(XmlNode oNode, DataTable Dt)
100     {
101         try
102         {
103             XmlElement xe2 = (XmlElement)oNode;
104             DataRow dr = Dt.NewRow();//New一行
105             dr[1] = xe2.GetAttribute("text");
106             if (!string.IsNullOrEmpty(xe2.GetAttribute("ID")))
107             {
108                 dr[2] = xe2.GetAttribute("ID");
109             }
110             else
111             {
112                 dr[2] = "1";
113             }
114             int index = 2;
115             for (int c = 0; c < oNode.ChildNodes.Count; c++)
116             {
117                 if (oNode.ChildNodes[c] is XmlElement)
118                 {
119                     for (int e = Dt.Columns.Count; e < (oNode.ChildNodes.Count * 2 + 3); e++)
120                         //Default3列，列数の数が確認，追加0列.
121                     {
122                         DataColumn dc = new DataColumn();
123                         Dt.Columns.Add(dc);
124                     }
125                     XmlElement xe1 = (XmlElement)oNode.ChildNodes[c];
126                     string zw1 = xe1.GetAttribute("text");
127                     dr[index += 1] = xe1.GetAttribute("text");
128                     dr[index += 1] = xe1.GetAttribute("ID");
129                 }
130             }
131             dr[0] = (index / 2).ToString();//第一列に節点数=index/2
132             Dt.Rows.Add(dr);
133             for (int i = 0; i < oNode.ChildNodes.Count; i++)
134             {
135                 if (oNode.ChildNodes[i] is XmlElement)
136                 {
137                     XmlElement xe = (XmlElement)oNode.ChildNodes[i];
138                     if (xe.ChildNodes.Count > 0)
139                     {
140                         CreatTable(oNode.ChildNodes[i], Dt); //繰り返す，節点がなくなるまで
```

図 6.44: NTO-Method-XML.cs（3）

```
142                 }
143             }
144         }
145         catch( Exception ex)
146         {
147             MessageBox.Show("Conversion Failed: " + ex.Message);
148         }
149     }
150     /// <summary>
151     /// Sel XML2
152     /// </summary>
153     /// <param name="sender"></param>
154     /// <param name="e"></param>
155     private void btn_xml2_Click(object sender, EventArgs e)
156     {
157         Stream myStream = null;
158         OpenFileDialog openFileDialog1 = new OpenFileDialog();
159         openFileDialog1.InitialDirectory = "E:\\Txt変換XML2\\Mark\\WindowsFormsApplication1\\WindowsForm
160         openFileDialog1.Filter = "txt files (*.xml)|*.xml";
161         openFileDialog1.FilterIndex = 2;
162         openFileDialog1.RestoreDirectory = true;
163         if (openFileDialog1.ShowDialog() == DialogResult.OK)
164         {
165             try
166             {
167                 if ((myStream = openFileDialog1.OpenFile()) != null)
168                 {
169                     XmlDocument xmlDoc = new XmlDocument();
170                     xmlDoc.Load(openFileDialog1.FileName);//XML読み込む
171                     tb2.Clear();
172                     NodeCount = 1;
173                     AttributeID(xmlDoc.ChildNodes[1]);// //まずXMLにID追加
174                     for (int i = 0; i <= 2; i++)
175                     //defuldでtableに三つの列追加，第一列節点の数保存，第二列に根節点が保存，第三根節点ID保存.
176                     {
177                         DataColumn dc = new DataColumn();
178                         tb2.Columns.Add(dc);
179                     }
180                     CreatTable(xmlDoc.ChildNodes[1], tb2);//XML2中のデータがTableに追加
181                     a = 1;//毎回ループしてから0に戻る.
182 
183                     //xmlDoc.Save("C://bookstore2.xml");
184                 }
185             }
186             catch (Exception ex)
187             {
188                 MessageBox.Show("Conversion Failed: " + ex.Message);
189             }
190         }
191     }
192     /// <summary>
193     /// データが分析，二つのtableから同じ節点を見つけ出す
194     /// </summary>
195     /// <param name="sender"></param>
196     /// <param name="e"></param>
```

図 6.45: NTO-Method-XML.cs（4）

```
197     private void btn_ok_Click(object sender, EventArgs e)
198     {
199         try
200         {
201             if (tb1.Rows.Count <= 0)
202             {
203                 MessageBox.Show("Conversion Failed: NO file");
204                 return;
205             }
206 
207             if (tb2.Rows.Count <= 0)
208             {
209                 MessageBox.Show("Conversion Failed: NO file");
210                 return;
211             }
212 
213             //tb1.Rows[0].Delete();//第一行にEmpty
214             //tb2.Rows[0].Delete();
215             IdenticalData();
216         }
217         catch(Exception  ex)
218         {
219             MessageBox.Show("Conversion Failed: "+ex.Message);
220         }
221 
222     }
223     /// <summary>
224     /// 二つのtableから同じ節点を見つけ出してほかの二つのtableに保存
225     /// </summary>
226     public void IdenticalData()
227     {
228         try
229         {
230             DataTable t1 = new DataTable();//両方の同じ節点二保存
231             DataTable t2 = new DataTable();
232             t1 = tb1.Copy();//Copy
233             t1.Rows.Clear();//table1のデータがEmpty
234 
235             t2 = tb2.Copy();
236             t2.Rows.Clear();
237 
238             string s1_txt = "";//tb1中の各行の節点のtext
239             string s1_id = "";//tb1中の各行の根節点のID
240             int s1_count = 0;//tb1中の各行の根節点の数(根節点を含み)
241             int col1 = tb1.Columns.Count;
242 
243             string s2_txt = "";
244             string s2_id = "";
245             int s2_count = 0;
246             int col2 = tb2.Columns.Count;
247             foreach (DataRow r1 in tb1.Rows)//tb1表ループ
248             {
249                 bool Iscommen1 = false;//
250                 s1_count = Convert.ToInt32(r1[0].ToString());
251                 s1_txt = r1[1].ToString();
```

図 6.46: NTO-Method-XML.cs（5）

```
253                 foreach (DataRow r2 in tb2.Rows)
254                 {
255                     s2_count = Convert.ToInt32(r2[0].ToString());
256                     s2_txt = r2[1].ToString();
257                     s2_id = r2[2].ToString();
258                     if (s1_txt == s2_txt && s1_count == s2_count && s1_count > 1)
259                         //節点に数と根節点と比較して同じかどうかを確認，ただし節点数が1より多きい場合
260                     {
261                         bool IsCommen = true;//二つのデータが同じと仮定
262                         for (int dex = 3; dex <= (2 * s2_count - 1); dex = dex + 2)
263                         {
264                             if (r1[dex].ToString() != r2[dex].ToString())//中のある節点が違うと
265                             {
266                                 IsCommen = false;//tb1とtb2のデータが違うならbreak:
267                                 break;
268                             }
269                         }
270                         if (IsCommen == true)//tb1とtb2のデータ同じなら
271                         {
272                             Iscommen1 = true;//设設定はtrue;
273                             t2.ImportRow(r2);
274                         }
275                     }
276                 }
277                 if (Iscommen1 == true)
278                 {
279                     t1.ImportRow(r1);//tableに追加
280                 }
281             }
282             FenXiTable(t1, t2);
283         }
284         catch (Exception ex)
285         {
286             MessageBox.Show("Conversion Failed: " + ex.Message);
287         }
288
289     }
290
291     /// <summary>
292     /// 二つの表の同じところ分析，先は深さ優先，次は幅優先
293     /// </summary>
294     /// <param name="dt1"></param>
295     /// <param name="dt2"></param>
296     public void FenXiTable(DataTable dt1, DataTable dt2)
297     {
298         try
299         {
300             #region ひとつの表の同じ部分を削除
301             DataTable dt3 = new DataTable();
302             DataTable dt4 = new DataTable();
303             dt4 = tb2.Copy();
304             dt4.Rows.Clear();
305             DataView dv = dt2.DefaultView;
306             dt3 = dv.ToTable(true, new string[] { "column3" });//各行の根節点のID同じかどうか確認
307             foreach (DataRow r in dt3.Rows)
```

図 6.47: NTO-Method-XML.cs（6）

```
{
    foreach (DataRow r2 in dt2.Rows)
    {
        if (r[0].ToString() == r2[2].ToString())
        {
            dt4.ImportRow(r2);
            break;
        }
    }
}
dt2.Rows.Clear();
dt2 = dt4.Copy();//dt2に繰り返し与える
#endregion

#region
string s1_txt = "";
int s1_count = 0;
int col1 = tb1.Columns.Count;
int col2 = tb2.Columns.Count;

string s3_txt = "";
string s3_id = "";
int s3_count = 0;
DataSet ds_all1 = new DataSet();
DataSet ds_all2 = new DataSet();
foreach (DataRow r1 in dt1.Rows)//(T1)XML1をループする
{
    #region 毎回ループするとマッチした部分をスキープする.
    bool b = false;
    foreach (DataTable dt in ds_all1.Tables)
    {
        foreach (DataRow r5 in dt.Rows)
        {
            if (r5[2].ToString() == r1[2].ToString())
            {
                b = true;
            }
        }
    }
    if (b)
    {
        continue;
    }
    #endregion
    int r3_index = 1;
    DataSet ds = new DataSet();
    DataSet ds2 = new DataSet();

    #region 判读有没有r2重复的数据，比如有没有另外一个bde
    int commendcount = 0;//先にすべてのマッチ部分を確認する，もしdt1中にdbeが存在すれば，既にある最大部分と比較して大きいほうがとる.
    foreach (DataRow r3 in dt1.Rows)
    {
        s3_count = Convert.ToInt32(r3[0].ToString());
        s3_txt = r3[1].ToString();
        s3_id = r3[2].ToString(); //根節点のID
```

図 6.48: NTO-Method-XML.cs（7）

```
362         s3_id = r3[2].ToString(); //根節点のID
363         if (s1_txt == s3_txt && s1_count == s3_count && s3_count > 1)//もしT1、T2の的根節点が同じ，ただ節点数も同じ
364         {
365             bool IsCommen3 = true;
366             for (int dex = 3; dex <= (2 * s3_count - 1); dex = dex + 2)//各節点にループし，もし違う節点があればbreak，もしあればTableについか
367             {
368                 if (r1[dex].ToString() != r3[dex].ToString()) //もし中にひとつの節点でも違うなら，違うと判別する．
369                 {
370                     IsCommen3 = false;
371                     break;
372                 }
373             }
374             if (IsCommen3 == true)
375             {
376                 commendcount++;
377 
378             }
379         }
380 
381     }
382     if (commendcount <= 0)
383     {
384         PiPei(r1, dt1, dt2, ds, ds2, ds_all1, ds_all2, true);//true示す，マッチのなければ最大の取り出す了
385     }
386     else
387     {
388         PiPei(r1, dt1, dt2, ds, ds2, ds_all1, ds_all2, false);//false示す，ほかのマッチ可能
389     }
390     int commendNum = 0;//
391     if (commendcount > 0)
392     {
393         foreach (DataRow r3 in dt1.Rows)
394         {
395             s3_count = Convert.ToInt32(r3[0].ToString());
396             s3_txt = r3[1].ToString();
397             s3_id = r3[2].ToString(); //根節点のID
398             if (s1_txt == s3_txt && s1_count == s3_count && s3_count > 1)
399             {
400                 bool IsCommen3 = true;
401                 for (int dex = 3; dex <= (2 * s3_count - 1); dex = dex + 2)
402                 {
403                     if (r1[dex].ToString() != r3[dex].ToString())
404                     {
405                         IsCommen3 = false;
406                         break;
407                     }
408                 }
409                 if (IsCommen3 == true)//すべての節点にループ終わり，子節点すべてマッチ．
410                 {
411                     commendNum++;
412                     if (commendcount == commendNum)
413                     {
414                         PiPei(r3, dt1, dt2, ds, ds2, ds_all1, ds_all2, true);
415                     }
416                     else
```

図 6.49: NTO-Method-XML.cs（8）

```
                    {
                        PiPei(r3, dt1, dt2, ds, ds2, ds_all1, ds_all2, false);
                    }
                }
            }
            r3_index++;
        }
    }
    #endregion
}
#endregion
#region 分析できたdataset中のdatatableに木構造の形に
int ds_tablecount = 1;
int SameCount = 0;
foreach (DataTable dt_XML in ds_all1.Tables)
{
    XmlDocument xmldoc = new XmlDocument();            //節宣言
    XmlDeclaration dec = xmldoc.CreateXmlDeclaration("1.0", "utf-8", null);
    xmldoc.AppendChild(dec);            //ひとつの根節点追加
    XmlElement xmlelem1 = xmldoc.CreateElement("funcType");
    xmlelem1.SetAttribute("level", "0");
    xmldoc.AppendChild(xmlelem1);
    XmlElement xmlelem_main = xmlelem1;
    int a = 1;
    int count = 0;
    foreach (DataRow r in dt_XML.Rows)
    {
        if (a == 1)//level0
        {
            XmlElement xmlelem2 = xmldoc.CreateElement("funcContents");
            xmlelem2.InnerText = r[1].ToString();
            xmlelem2.SetAttribute("ID", r[2].ToString());
            xmlelem_main.AppendChild(xmlelem2);
            xmlelem_main = xmlelem2;
            for (int i = 3; i <= Convert.ToInt32(r[0].ToString()) * 2; i = i + 2)
            {
                XmlElement xmlelem3 = xmldoc.CreateElement("funcContents");
                xmlelem3.InnerText = r[i].ToString();
                xmlelem3.SetAttribute("ID", r[i + 1].ToString());
                xmlelem_main.AppendChild(xmlelem3);
            }
            count = Convert.ToInt32(r[0].ToString());
            SameCount =SameCount+ Convert.ToInt32(r[0].ToString());
        }
        else
        {
            count = count + Convert.ToInt32(r[0].ToString()) - 1;
            SameCount = SameCount + Convert.ToInt32(r[0].ToString()) - 1;
            XmlNode node = xmldoc.SelectSingleNode("//funcContents[@ID=" + r[2].ToString() + "]");
            if (node != null)
            {
                for (int i = 3; i <= Convert.ToInt32(r[0].ToString()) * 2; i = i + 2)
                {
                    XmlElement xmlelem3 = xmldoc.CreateElement("funcContents");
                    xmlelem3.InnerText = r[i].ToString();
```

図 6.50: NTO-Method-XML.cs（9）

```
                                xmlelem3.SetAttribute("ID", r[i + 1].ToString());
                                node.AppendChild(xmlelem3);
                            }
                        }
                    }
                    a++;
                }
                if (!Directory.Exists(Application.StartupPath + "//XMLNodes"))
                {
                    Directory.CreateDirectory(Application.StartupPath + "//XMLNodes");
                }
                xmldoc.Save(string.Format(@"XMLNodes//[{1}]{0}.xml", "a" + ds_tablecount.ToString() + "", count.ToString()));//最後出力パス
                ds_tablecount++;
            }
            //MessageBox.Show("Similarity: " + SameCount.ToString()+ "");
            //MessageBox.Show("Similarity: " + NodeCount.ToString() + "");
            MessageBox.Show("Similarity: " + ((double)SameCount)/NodeCount + "");
            tb1.Clear();
            tb2.Clear();
            #endregion
        }
        catch(Exception ex)
        {
            MessageBox.Show("Conversion Failed: " + ex.Message);
        }
    }
    /// <summary>
    /// dt1の各行を基準として，dt2の各行と比較する，もしマッチすれば子節点と比較
    /// </summary>
    /// <param name="r1"></param>
    /// <param name="dt1"></param>
    /// <param name="dt2"></param>
    /// <param name="ds"></param>
    /// <param name="ds2"></param>
    /// <param name="ds_all1"></param>
    /// <param name="ds_all2"></param>
    /// <param name="IsEnd"></param>
    public void PiPei(DataRow r1,DataTable dt1,DataTable dt2,DataSet ds,DataSet ds2, DataSet ds_all1,DataSet ds_all2,bool IsEnd)
    {
        try
        {
            string s1_txt = "";
            string s1_id = "";
            int s1_count = 0;
            int col1 = tb1.Columns.Count;
            string s2_txt = "";
            string s2_id = "";
            int s2_count = 0;
            int col2 = tb2.Columns.Count;
            int MaxNodeNum = 0;
            s1_count = Convert.ToInt32(r1[0].ToString());
            s1_txt = r1[1].ToString();
            s1_id = r1[2].ToString(); //根節点のID
            foreach (DataRow r2 in dt2.Rows)
            {
```

図 6.51: NTO-Method-XML.cs（10）

```
526         {
527             #region 毎回ループするとマッチした部分をスキープする.
528             bool b = false;
529             foreach (DataTable dt in ds_all2.Tables)
530             {
531                 foreach (DataRow r5 in dt.Rows)
532                 {
533                     if (r5[2].ToString() == r2[2].ToString())
534                     {
535                         b = true;
536                     }
537                 }
538             }
539             if (b)
540             {
541                 continue;
542             }
543             #endregion
544
545
546
547             Num = 0;
548             s2_count = Convert.ToInt32(r2[0].ToString());//各節点の数
549             s2_txt = r2[1].ToString();//根節点text
550             s2_id = r2[2].ToString();//根節点ID
551             if (s1_txt == s2_txt && s1_count == s2_count && s1_count > 1)
552             {
553                 bool IsCommen = true;
554                 for (int dex = 3; dex <= (2 * s2_count - 1); dex = dex + 2)
555                 {
556                     if (r1[dex].ToString() != r2[dex].ToString())
557                         //もし中にひとつの節点でも違うなら，違うと判別する.。
558                     {
559                         IsCommen = false;
560                         break;
561                     }
562                 }
563                 if (IsCommen == true)
564                 {
565                     DataTable data1 = new DataTable();
566                     for (int e = data1.Columns.Count; e < (s2_count * 2 + 3); e++)
567                     {
568                         DataColumn dc = new DataColumn();
569                         data1.Columns.Add(dc);
570                     }
571                     DataTable data2 = new DataTable();
572                     for (int e = data2.Columns.Count; e < (s2_count * 2 + 3); e++)
573                     {
574                         DataColumn dc = new DataColumn();
575                         data2.Columns.Add(dc);
576                     }
577                     data1.ImportRow(r1);
578                     data2.ImportRow(r2);
579                     Num = s2_count;
580                     for (int i = 4; i <= Convert.ToInt32(r1[0].ToString()) * 2; i = i + 2)
```

図 6.52: NTO-Method-XML.cs（11）

```
580             for (int i = 4; i <= Convert.ToInt32(r1[0].ToString()) * 2; i = i + 2)
581             {
582                 bool Isc = false;
583                 int Col1 = 0;
584                 bool Isd = false;
585                 int Col2 = 0;
586                 foreach (DataRow r3 in dt1.Rows)
587                 {
588                     if (r1[i].ToString() == r3[2].ToString())
589                     {
590                         Isc = true;
591                         break;
592                     }
593                     Col1++;
594                 }
595                 foreach (DataRow r4 in dt2.Rows)
596                 {
597                     if (r2[i].ToString() == r4[2].ToString())
598                     {
599                         Isd = true;
600                         break;
601                     }
602                     Col2++;
603                 }
604                 if (Isc && Isd)
605                 {
606                     if (dt1.Rows[Col1][0].ToString() == dt2.Rows[Col2][0].ToString())//当T1、T2根節点数と節点数同じ
607                     {
608                         bool IsCommen2 = true;
609                         for (int dex = 3; dex <= (2 * Convert.ToInt32(dt1.Rows[Col1][0].ToString()) - 1); dex = dex + 2)
610                         {
611                             if (dt1.Rows[Col1][dex].ToString() != dt2.Rows[Col2][dex].ToString())
612                                 //もし中にひとつの節点でも違うなら，違うと判別する.
613                             {
614                                 IsCommen2 = false;
615                                 break;
616                             }
617                         }
618                         if (IsCommen2)
619                         {
620                             Num = Num + Convert.ToInt32(dt1.Rows[Col1][0]);
621                             DiGui2(dt1.Rows[Col1], dt1, dt2.Rows[Col2], dt2, data1, data2);
622 
623                         }
624                     }
625                 }
626             }
627             ds.Tables.Add(data1);
628             ds2.Tables.Add(data2);
629         }
630     }
631     if (Num > MaxNodeNum)
632     {
633         MaxNodeNum = Num;
634     }
```

図 6.53: NTO-Method-XML.cs（12）

```
636             if (IsEnd)
637             {
638                 DataSetFuzhi(ds, ds2, ds_all1, ds_all2);
639             }
640         }
641         catch (Exception ex)
642         {
643             MessageBox.Show("Conversion Failed: " + ex.Message);
644         }
645     }
646     /// <summary>
647     /// 繰り返し関連のマッチ文，table中に保存
648     /// </summary>
649     /// <param name="r1"></param>
650     /// <param name="dt1"></param>
651     /// <param name="r2"></param>
652     /// <param name="dt2"></param>
653     /// <param name="data1"></param>
654     /// <param name="data2"></param>
655     public void DiGui2(DataRow r1, DataTable dt1, DataRow r2, DataTable dt2,DataTable data1,DataTable data2)
656     {
657         try
658         {
659             for (int e = data1.Columns.Count; e < (Convert.ToInt32( r1[0].ToString())* 2 + 3); e++)
660             {
661                 DataColumn dc = new DataColumn();
662                 data1.Columns.Add(dc);
663             }
664             for (int e = data2.Columns.Count; e < (Convert.ToInt32(r2[0].ToString()) * 2 + 3); e++)
665             {
666                 DataColumn dc = new DataColumn();
667                 data2.Columns.Add(dc);
668             }
669             data1.ImportRow(r1);
670             data2.ImportRow(r2);
671             for (int i = 4; i <= Convert.ToInt32(r1[0].ToString()) * 2; i = i + 2)
672             {
673                 bool Isc = false;
674                 int Col1 = 0;
675                 bool Isd = false;
676                 int Col2 = 0;
677                 foreach (DataRow r_1 in dt1.Rows)
678                 {
679                     if (r1[i].ToString() == r_1[2].ToString())
680                     {
681                         Isc = true;
682                         break;
683                     }
684                     Col1++;
685                 }
686                 foreach (DataRow r_2 in dt2.Rows)
687                 {
688                     if (r2[i].ToString() == r_2[2].ToString())
689                     {
690                         Isd = true;
```

図 6.54: NTO-Method-XML.cs（13）

```
691                         break;
692                     }
693                     Col2++;
694                 }
695                 if (Isc && Isd)//もし同じID節点があれば
696                 {
697                     if (dt1.Rows[Col1][0].ToString() == dt2.Rows[Col2][0].ToString())//T1、T2の根節点等しい
698                     {
699                         bool IsCommen2 = true;
700                         for (int dex = 3; dex <= (2 * Convert.ToInt32(dt1.Rows[Col1][0].ToString()) - 1); dex = dex + 2)
701                         {
702                             if (dt1.Rows[Col1][dex].ToString() != dt2.Rows[Col2][dex].ToString())
703                                 //もし中にひとつの節点でも違うなら，違うと判別する.
704                             {
705                                 IsCommen2 = false;
706                                 break;
707                             }
708                         }
709                         if (IsCommen2)
710                         {
711                             Num = Num + Convert.ToInt32(dt1.Rows[Col1][0]);
712                             DiGui2(dt1.Rows[Col1], dt1, dt2.Rows[Col2], dt2, data1, data2);//再帰
713                         }
714                     }
715                 }
716             }
717         }
718         catch(Exception  ex)
719         {
720             MessageBox.Show("Conversion Failed: " + ex.Message);
721         }
722     }
723     /// <summary>
724     /// 最後最大の共通部分を求め
725     /// </summary>
726     /// <param name="ds"></param>
727     /// <param name="ds2"></param>
728     /// <param name="ds_all1"></param>
729     /// <param name="ds_all2"></param>
730     public void DataSetFuzhi(DataSet ds, DataSet ds2, DataSet ds_all1, DataSet ds_all2)
731     {
732         try
733         {
734             DataTable dt_copy = new DataTable();
735             DataTable dt_copy2 = new DataTable();
736             int max1 = 0;
737             foreach (DataTable dt in ds.Tables)
738             {
739                 int count = 0;
740                 foreach (DataRow dr in dt.Rows)
741                 {
742                     count += Convert.ToInt32(dr[0]);
743                 }
744                 if (count > max1)
```

図 6.55: NTO-Method-XML.cs（14）

```
743                     }
744                     if (count > max1)
745                     {
746                         max1 = count;
747                         dt_copy.Clear();
748                         dt_copy = dt.Copy();
749                     }
750                 }
751                 int max = 0;
752                 foreach (DataTable dt in ds2.Tables)
753                 {
754                     int count = 0;
755                     foreach (DataRow dr in dt.Rows)
756                     {
757                         count += Convert.ToInt32(dr[0]);
758                     }
759                     if (count > max)
760                     {
761                         max = count;
762                         dt_copy2.Clear();
763                         dt_copy2 = dt.Copy();
764                     }
765                 }
766                 dt_copy.TableName = "Table" + (ds_all1.Tables.Count + 1);
767                 ds_all1.Tables.Add(dt_copy);
768                 dt_copy2.TableName = "Table" + (ds_all2.Tables.Count + 1);
769                 ds_all2.Tables.Add(dt_copy2);
770             }
771             catch( Exception ex)
772             {
773                 MessageBox.Show("IE过: " + ex.Message);
774             }
775         }
776     }
777 }
```

図 6.56: NTO-Method-XML.cs（15）

最後に，DM 手法のソースプログラムを示す．本プログラムの実行ファイルは $DM-Method-XML.cs$ という名前のスクリプト CS ーファイルである．このプログラムには三つの関数が存在する．partial 関数では，XML を分析各節点の値を取り，各 table 中におく作業である．PaiXu 関数では，XML.XPath Method を使いマッチ条件をとる関数である．button3Click 関数では各行の最大値をとり，子節点の数回比較して最大 MAX 値をとる関数である．

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Xml;

namespace XML
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

        }
        DataTable dt1 = new DataTable();
        DataTable dt2 = new DataTable();
        int MaxLabel = 0;
        /// <summary>
        /// SelectFile XML
        /// </summary>
        /// <param name="dt">各XMLにひとつずつDataTable対応</param>
        public void SelectFile(DataTable dt)
        {
            Stream myStream = null;
            OpenFileDialog openFileDialog1 = new OpenFileDialog();
            openFileDialog1.InitialDirectory = "E:\\TxtchangeXML2\\XMLMark\\XML\\XML\\XML";
            openFileDialog1.Filter = "txt files (*.xml)|*.xml";
            openFileDialog1.FilterIndex = 2;
            openFileDialog1.RestoreDirectory = true;
            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                try
                {
                    if ((myStream = openFileDialog1.OpenFile()) != null)
                    {
                        XmlDocument xmlDoc = new XmlDocument();
                        xmlDoc.Load(openFileDialog1.FileName);
                        dt.Clear();
                        PaiXu(xmlDoc, dt);
                    }
                }
                catch (Exception ex)
                {
                   // MessageBox.Show("Conversion Failed: " + ex.Message);
                }
            }
        }
        /// <summary>
        /// XMLを分析，各節点の値を取り，各tableno中におく，XML.XPath方法を使いマッチ条件をとる．
```

図 6.57: DM-Method-XML.cs（1）

```
56          /// </summary>
57          /// <param name="xmlDoc"></param>
58          /// <param name="dt"></param>
59          public void PaiXu(XmlDocument xmlDoc, DataTable dt)
60          {
61              //*************************************最大のlabel*****************************************//
62              XmlNode MaxLabelNode = xmlDoc.SelectSingleNode("//funcContents/@label[not(.<//funcContents/@label)]");
63              if (MaxLabelNode != null)
64              {
65                  MaxLabel = Convert.ToInt32(MaxLabelNode.Value);
66
67              }
68              //*************************************最大のlevel*****************************************//
69              int levelMax = 0;
70              XmlNode levelNode = xmlDoc.SelectSingleNode("//funcContents/@level[not(.<//funcContents/@level)]");
71              levelMax = Convert.ToInt32(levelNode.Value);
72
73              //*************************************最大的子節点数，Tableに最大列数を追加*************************
74              int childMax = 0;
75              XmlNode childNode1 = xmlDoc.SelectSingleNode("//funcName/@numofchildren[not(.<//funcName/@numofchildren
76              XmlNode childNode2 = xmlDoc.SelectSingleNode("//funcArgType/@numofchildren[not(.<//funcArgType/@numofch
77              XmlNode childNode3 = xmlDoc.SelectSingleNode("//funcContents/@numofchildren[not(.<//funcContents/@numof
78              if (childNode1 != null)//もしfuncName存在すれば，numofchildren（子節点の数）値をえる．
79              {
80                  if (Convert.ToInt32(childNode1.Value) >= childMax)
81                  {
82                      childMax = Convert.ToInt32(childNode1.Value);
83                  }
84              }
85              if (childNode2 != null)//もしfuncArgType存在すれば，numofchildren（子節点の数）値をえる．
86              {
87                  if (Convert.ToInt32(childNode2.Value) >= childMax)
88                  {
89                      childMax = Convert.ToInt32(childNode2.Value);
90                  }
91              }
92              if (childNode3 != null)//funcContents存在すれば，numofchildren（子節点の数）値をえる．
93              {
94                  if (Convert.ToInt32(childNode3.Value) >= childMax)
95                  {
96                      childMax = Convert.ToInt32(childNode3.Value);
97                  }
98              }
99              //*************************************tableに列追加，childMax以上求める最大の子節点数*****************
100             for (int i = 0; i <= childMax; i++)
101             {
102                 DataColumn dc = new DataColumn();
103                 dt.Columns.Add(dc);
104             }
105             //*************************************データをtable中いれる，第一列が節点数とする*******************//
106
107             //*****************funcType値追加****************//
108             int mainindex = 0;
```

図 6.58: DM-Method-XML.cs（2）

```
108             int mainindex = 0;
109             XmlNode Node_funcType = xmlDoc.SelectSingleNode("//funcType");//たとえ;XMLに"int"
110             DataRow dr1 = dt.NewRow();
111             dr1[0] = "1";//个数为1
112             dr1[1] = Node_funcType.Attributes["text"].Value;
113             dt.Rows.Add(dr1);
114             mainindex++;
115 
116             XmlNode Node_funcName = xmlDoc.SelectSingleNode("//funcName");//たとえXMLに"main"
117             DataRow dr2 = dt.NewRow();
118             dr2[0] = "1";
119             dr2[1] = Node_funcName.Attributes["text"].Value;
120             dt.Rows.Add(dr2);
121             mainindex++;
122 
123             XmlNode Node_funcArgType = xmlDoc.SelectSingleNode("//funcArgType[@level=" + mainindex + "]");//たとえXMLに"void"
124             if (Node_funcArgType != null)
125             {
126                 DataRow dr3 = dt.NewRow();
127                 dr3[0] = "1";
128                 dr3[1] = Node_funcArgType.Attributes["text"].Value;
129                 dt.Rows.Add(dr3);
130                 mainindex++;
131 
132             }
133             for (int i = mainindex; i <= levelMax; i++)
134                 //最大レベル数LevelMax,レベルごとループ時，各レベルのデータをtableに追加，一行を一層に代表する
135             {
136                 XmlNodeList items = xmlDoc.SelectNodes("//funcContents[@level=" + i + "]");
137                 if (items.Count > 0)
138                 {
139                     int row_index = 1;
140                     int Itemcount = items.Count;//各レベルのデータ
141                     DataRow dr = dt.NewRow();//实例化Row
142                     dr[0] = Itemcount.ToString();
143                     foreach (XmlNode n in items)
144                     {
145                         if (n != null)
146                         {
147                             dr[row_index] = n.Attributes["text"].Value;
148                             row_index++;
149                         }
150                     }
151                     dt.Rows.Add(dr);//最後tableの中に追加
152 
153                 }
154             }
155         }
156         /// <summary>
157         /// </summary>
158         /// <param name="sender"></param>
159         /// <param name="e"></param>
160         private void button3_Click(object sender, EventArgs e)
161         {
```

図 6.59: DM-Method-XML.cs（3）

```
160         private void button3_Click(object sender, EventArgs e)
161         {
162             try
163             {
164                 int MarkCount = 0;//同じ節点を保存
165                 int Row_index = 0;
166                 foreach (DataRow r in dt1.Rows)//ループdt1(sel XMLをdt1の中に)
167                 {
168                     int c = 1;
169                     int sum1 = Convert.ToInt32(r[0].ToString());//節点数(同じlevel節点の数)
170                     int MaxNumber = 0;
171                     if (dt2.Rows.Count < (Row_index + 1))
172                     {
173                         break;
174                     }
175                     #region 各行の最大値をとる，子節点の数回比較する，最後MAXをとる
176                     for (int j = 1; j <= sum1; j++)
177                     {
178                         c = 1;
179                         int Number = 0;
180                         int d = 1;
181                         for (int i = j; i <= sum1; i++)//tb1ループ
182                         {
183                             int sum2 = Convert.ToInt32(dt2.Rows[Row_index][0].ToString());
184                             c = d;
185                             for (int a = c; a <= sum2; a++)//tb2ループ
186                             {
187                                 bool Isequer = false;
188                                 if (r[i].ToString() == dt2.Rows[Row_index][c].ToString())
189                                 {
190                                     Number++;
191                                     c++;
192                                     d++;
193                                     Isequer = true;
194                                     break;
195                                 }
196                                 else
197                                 {
198                                     c++;//マッチない，ループ停止
199                                 }
200                                 if (Isequer)
201                                 {
202                                     break;
203                                 }
204                             }
205                         }
206                         if (Number >= MaxNumber)
207                         {
208                             MaxNumber = Number;
209                         }
210                     }
211                     #endregion
212                     Row_index++;
213                     MarkCount = MarkCount + MaxNumber;
```

図 6.60: DM-Method-XML.cs（4）

```
211                 #endregion
212                 Row_index++;
213                 MarkCount = MarkCount + MaxNumber;
214             }
215             //結果出力
216             MessageBox.Show("Similarity:" + (((double)MarkCount/MaxLabel)).ToString() + "");
217 
218         }
219         catch (Exception ex)
220         {
221             MessageBox.Show("Conversion Failed: " + ex.Message);
222         }
223     }
224     /// <summary>
225     /// button---XML1
226     /// </summary>
227     /// <param name="sender"></param>
228     /// <param name="e"></param>
229     private void XML1_Click(object sender, EventArgs e)
230     {
231         SelectFile(dt1);
232     }
233     /// <summary>
234     /// buttom--XML2
235     /// </summary>
236     /// <param name="sender"></param>
237     /// <param name="e"></param>
238     private void XML2_Click(object sender, EventArgs e)
239     {
240         SelectFile(dt2);
241     }
242 
243 }
244 }
245 
```

図 6.61: DM-Method-XML.cs（5）