卒　業　研　究　報　告　書

令和6年度

研究題目

プログラミングスキル表現手法としての
スキルツリーの提案

指導教員　　上　野　秀　剛　　准教授

氏　　名　　薮　中　天　空

令和7年1月23日　提出

奈 良 工 業 高 等 専 門 学 校　　情 報 工 学 科

プログラミングスキル表現手法としての スキルツリーの提案

上野研究室　薮中天空

プログラミング初学者の効率的な学習には，学習者の能力（プログラミングスキル）を評価し，理解が不足している要素に対応した支援が必要である．しかし，現状のスキルの評価方式は比較的大きな範囲での課題やテストを実施することが多く，より詳細な単位のスキルの表現・評価ができない．また，プログラミングスキルは詳細なスキルの集合体であり，スキルには個々で独立して学習できるスキルと，学習したスキルの組み合わせとして学習するスキルがある．それぞれのスキルの間には，組み合わせや順序などの「関係」が存在するが，これは表形式などの単なる一覧では表現できず，現状では表現する方法がない．そこで，本研究では，プログラミングスキルの学習を助け，理解状況をより詳細かつ正確に表現する方法としてスキルツリーを用いることを提案する．スキルツリーはビデオゲームの分野で用いられている概念で，キャラクターが獲得できる順序関係の存在するスキルの集合体として表示され，各スキルの獲得状況を表すことができる．これらの特性からスキルツリーはプログラミングスキルの学習形態と構造が類似しており，表形式や木構造では実現できない構造の表現もできるため，プログラミングスキルの視覚的な表現方法として効果的であると考える．ケーススタディでは，実際にプログラミング教育で用いられているデータを参照し，詳細なプログラミングスキルの一覧を求め，スキルツリーを作成することで，プログラミングスキルの表現方法として適切かどうかを確認する．データは，国立高等専門学校全体のシラバスデータから収集し，分かち書きとn-gramによる自然言語処理でJava言語のプログラミング初学者範囲関連単語を抽出することで，詳細なスキルの一覧とした．参考データをもとに，サブスキル群とその「関係」を構成し，形状の観点から2種類のスキルツリーを比較した．スキルツリーを作成した結果，スキル同士の学習順序や組み合わせによる新たなスキルの学習の「関係」を表現することができたといえる．形状を比較すると，同じスキル群や「関係」を用いて形状の違うスキルツリーを作成すると，それぞれの得意とする表現や構造に違いがあることがわかった．

目 次

1 はじめに

プログラミング初学者の効率的な学習には，学習者の能力（プログラミングスキル）を評価し，理解が不足している要素に対応した支援が必要である．プログラミングスキルを評価する方法としては，課題やテストを実施することが一般的である．課題やテストは，条件分岐や配列といった比較的大きな学習範囲に対して問題が作成され，正否や100点法によって評価される．しかし，受講者の理解が不足しているのはより細かい単位の要素である場合も多い．例えば，switch-case文全体がわからないのではなく，defaultが選択された際の遷移が分からない場合があるが，このような詳細さでプログラミングスキルを表現・評価する方法は存在しない．

また，スキルは個々の要素として学習するものと学習済みのスキルの組み合わせとして学習するものがある．例えば，変数の四則演算を学習したいときには，プログラミングにおける四則演算の記述方法，数値を取り扱う変数の知識，変数を上書きする知識の三つのスキルがそれぞれ学習済みである必要がある．特に，組み合わせはスキルを細かい単位で見たときに重要であり，あるスキルを理解していたとしてもそのスキルを発展させたスキルを理解しているとは限らない．組み合わせるスキルが存在しているということは，スキル間には学習順序が存在している．前述した変数の四則演算を学習するときには，事前に獲得しなければならないスキルが複数存在しているため，学習の順序は決まっている．一方で事前に獲得するスキル同士は，自由な順序で学習できることがある．つまり，プログラミングの学習や授業では通常，参考書の目次やカリキュラムのシラバスに沿って学習を進めるが，必ずしもその順番が絶対ではないといえる．ここから，プログラミングスキルは細かい様々なスキルの集合であり，スキル間には組み合わせや順序などの「関係」が存在している．これは表形式などの単なる一覧では表現することはできない．また，事前に学習すべきスキルの学習が完了して初めて，それらの組み合わせによるスキルを学習できたかの評価に移る，といった「関係」に合わせた評価をすることが必要である．

本研究は，プログラミングスキルの学習を助け，理解状況をより詳細かつ正確に表現する方法としてスキルツリーを用いる事を提案する．これまで，プログラミングスキル研究の分野では，ソースコードの読み方や記述方法からプログラミング記述者のスキルを判定する研究が行われている[1, 2]．本研究では，学習者のプログラミングスキルを知識の観点から視覚化し，表現することを試みる．また，プログラミング教育の分野では，学習者が作成したソースコードや自己評価から学習者の理解状況や学習意欲を判定し，能力に合わせた演習課題を出題するシステムが提案されている[3]．しかし，この判定はそれぞれの問題ごとに行われるもので，詳細なスキルに着目した判定は実現できていないといえる．本研究

では，より詳細なプログラミングスキルの表現を実現することで，学習者のスキルを詳細に判定することを目指す．スキルツリーはビデオゲームの分野で用いられている概念で，ゲーム内のキャラクターの能力をどのように育成することができるか，また，これまでどのように育成したかを示す図表の総称である．スキルツリーはキャラクターが獲得できる順序関係の存在するスキルの集合体として表示され，それぞれのスキルの獲得・未獲得の状態を表すことができる．また，キャラクターの能力のデザインによって様々な形がある．本研究で表現したいプログラミングスキルは，学習順序が存在したり，学習の支援や能力の評価において学習者が獲得しているスキルの把握が重要であるため，スキルツリーという表現が適切であると考える．ただ，実際の表現方法として適切か，どのような形状が適しているかは検討の余地がある．ケーススタディでは，実際にプログラミング教育で利用されているデータを参照し，詳細なプログラミングスキルの一覧を求め，スキルツリーを作成することで，プログラミングスキルの表現方法として適切かどうかを確認する．本研究で，スキルツリーによるプログラミングスキルの表現を実現することで，これまでにはなかった，詳細なスキルの可視化と，順序や組み合わせを導入したプログラミングスキルの表現が可能となる．それにより，初学者のプログラミングスキルがより低コストで詳細に把握できるようになり，効率的な学習支援や問題作成につながる．

以下，2章では準備について説明し，3章で提案手法，4章でケーススタディとして本研究で実際に行ったデータ収集やスキルツリーの作成過程を示す．5章では，本研究のまとめと今後の発展について説明する．

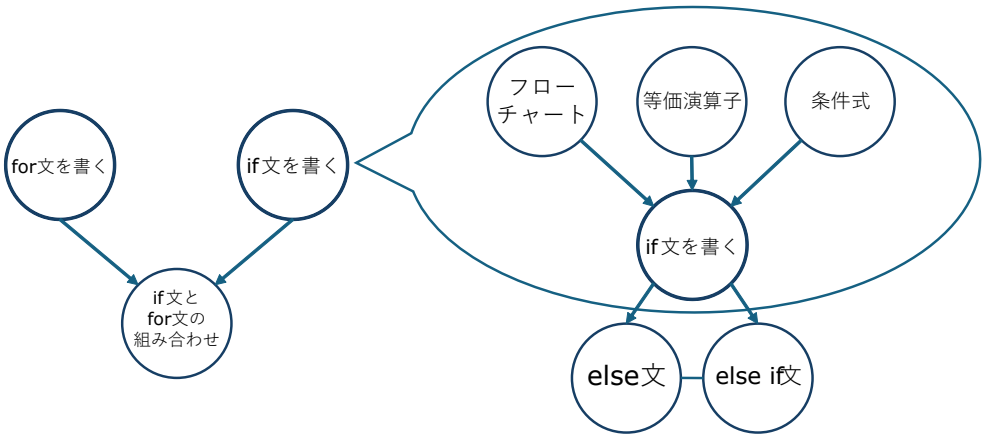

図 1 スキルの組み合わせの構造

2 準備

2.1 プログラミングスキル

プログラミングスキルとは，一般的にはプログラミング言語を利用してプログラムを記述したり，実行したりすることでタスクを解決できる能力であると理解されていることが多い．プログラミングスキルの評価・分析を目的とした論文は複数存在している [1, 2]. 既存研究におけるプログラミングスキルの評価基準として，コーディングの速度やプログラム実行結果の正確性，デバック能力など様々な観点が挙げらている．本研究ではプログラミングスキルを，プログラミングを記述する際に必要なif文やfor文などの知識の集合であると定義する．

プログラミングスキルを構成するスキルには，個々に独立して学習するものと，既に学習したスキルの組み合わせとして学習するものがある．図1にスキルの組み合わせの構造を示す. 例えば，if文を書く，for文を書くというのはそれぞれがスキルであるが，if文とfor文を組み合わせた文を書くことができるというのは，それぞれのスキルとは別のスキルである．特に組み合わせは，それぞれのスキルを詳細に理解するためにもより重要である．例えばif文を書く，というスキルはフローチャートや等価演算子，条件式のスキルを学習し，組み合わせることで初めて理解できる．そこから，else if文やelse文のスキルが組み合わせられる．このとき，if文を理解しても，else if文,else文を理解できるとは限らない．

また，組み合わせが存在するということは，組み合わせる前のスキルと後のスキルには学習に必要な順序が存在する可能性が高い．例えばif文とfor文を組み合わせるというスキルを学習するためには，if文とfor文のそれぞれのスキルを事前に学習することが必要である．ただ，その場合にif文とfor文の間に，学習の順序はなく，どちらを先に学習しても良い．ここから，プログラミングスキルは細かいスキルに分かれており，組み合わせや順序などの「関係」が存在する．この

「関係」は単純な一覧や目次のような形式では適切に表現できない．本研究では，それぞれのスキルを学習することで，学習することができるようになるより詳細なスキルを，それぞれのスキルの「サブスキル」と呼称する．例えばif文というスキルでは，else if文やelse文のスキルがif文の「サブスキル」となる．同様に，if文というスキルもフローチャートや等価演算子，条件式のスキルの「サブスキル」であるといえる．

本研究では，知識の観点からプログラミングスキルを構成するスキルとサブスキルがどのような形態を持つかを明確にすることで，プログラミングスキルのより詳細かつ正確な表現を提案する．プログラミングスキルを構成するスキルとサブスキルの関係を整理することで，学習者の理解が不足しているスキルや，次に学習するべきスキルの順序，特定の問題を解くのに必要なスキルの組み合わせを理解することを助ける．

2.2 スキルツリー

スキルツリーはビデオゲームの分野で用いられている概念で，ゲーム内のキャラクターが獲得できるスキルや，獲得済のスキルを示す図表の総称である．図2にビデオゲームPath of Exileで使われるスキルツリー[1]の一部を示す. 図中のアイコンが書かれた丸が1つのスキルを表す. スキルツリーは多数の小さなスキルとサブスキルの集合体として構成される．例のゲームの場合，それぞれのスキルが分野ごとに色分けされていて，近接戦闘に関わるスキルは赤色，魔力に関わるスキルは青色，移動速度や攻撃速度に関わるスキルは緑色で示されている．また，スキル同士をつなぐ線はスキル同士の隣接関係を表す. スキルツリーは，開始時点では獲得可能なスキルが1つ，または複数存在し，それらのスキルを獲得することで，さらに隣接するサブスキルを獲得可能になる．ここから，それぞれのスキルには獲得する順序があることがわかる．キャラクターは，ゲームによってレベルアップにより得られるスキルポイントや収集した素材を消費することでそれぞれのスキルを獲得することができる．スキルの獲得を繰り返すことでキャラクターは強くなり，また獲得できるスキルの種類が増えたり強力なスキルを獲得することができるようになったりする．例のゲームの場合，中央の大きな円からスキルツリーが開始され，最初に周囲の6つのスキルが獲得可能な状態になっている．プレイヤーは獲得するスキルを選択し分岐したり合流したりしながらそれぞれのサブスキルを獲得することで，自由にキャラクターのステータスを育成することができる．また，合流するスキルがある場合，ゲームによって前提となる全てのスキルを獲得することで合流するスキルを獲得可能になる場合と，前提のスキルのどれか1つを獲得することで合流するスキルを獲得可能になる場合がある．例のゲームの場合では，図の右上や下ではスキルツリーを進めていくと，

[1] https://jp.pathofexile.com/passive-skill-tree

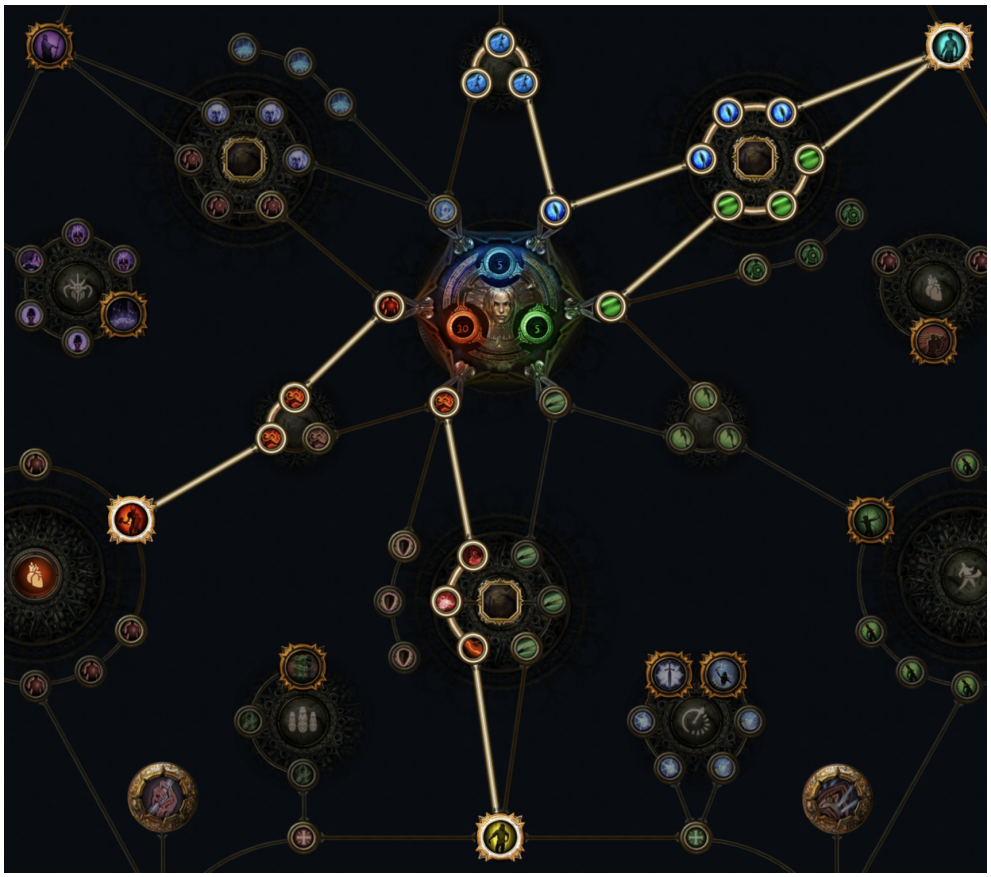

図 2 ビデオゲーム Path of Exile のスキルツリー

別の分野のスキルが合流し，新たなスキルになっている部分があり，例のゲームではどちらか一方の分野のスキルを獲得することで，合流したスキルを獲得可能となる．このサブスキルを解放する繰り返しによって，同じキャラクターでもプレイヤー次第で成長させる能力に優先順位をつけたり，様々なプレイスタイルを選択できる．

スキルツリーの形状はゲームによって樹状，放射状など様々なものがある．形状の違いによって，前段落で説明したスキルツリーの開始地点，スキル獲得の順番や合流の仕方は様々に変化するため，ゲームやキャラクターの特色を表現することができる．これらの性質こそがスキルツリーの醍醐味であり，システムとしての利点でもある．

また，スキルツリーという名称については，木構造を連想するような名称であり，また同様の機能であってもゲームによって名称が異なる場合がほとんどである．本研究ではスキルツリーという名称がゲーム業界で知名度が高く，広く浸透していると考えるため，ゲーム業界で用いられている形式を転用する，というテーマをわかりやすく伝えるために「スキルツリー」という名称を利用する．類似する表現としてスキルマップが挙げられる場合があるが，スキルマップについ

てはスキルを羅列しただけの単なる表形式の名称としてビジネスのシーンで用いられていることが多く, 表現したいプログラミングスキルの形式としても本研究において適切でないと考える.

本研究では, スキルツリーが多数のスキルの集合体で構成され, 詳細なスキルを順序や組み合わせの関係を設定して表現できる性質や, 取得するスキルを選択できる性質に着目し, プログラミングスキルの表現方法として用いることを提案する. また, スキルツリーはゲーム業界で利用される際に基本的に無向グラフであり, スキル選択の自由度を高い要因の一つとなっている. 今回ケーススタディで作成するスキルツリーは全体を無向グラフとして, 向きを示す必要がある（明確に順序が一通りである, または組み合わせに関わる）箇所に部分的に有向グラフを適用している.

2.3 分かち書き

日本語における分かち書きとは, 語と語や文節と分節の間に空白を挟んで記述することである. 日本語の自然言語処理を行う際には, 単語の分かち書き（単語分割：word segmentation）は最も基本的かつ重要な課題である. 英語やフランス語など欧米言語の正書法(orthography)では, 単語と単語の間に空白をいれるため, 単語の認識は大きな問題にはならない. しかし, 日本語のように単語の間に区切り記号を持たない言語は, 単語の認識が根本的な問題となり, コンピュータによる検索やデータ処理が非常に難しい[4].

日本語の分かち書きでは, 名詞や形容詞, 動詞語幹, 活用語尾, 助詞, 助動詞などの語を, 意味を担う最小の言語要素である形態素(morpheme)として検出して分割する. そのような日本語の分かち書きのための自然言語処理のツールとして, 本研究ではMeCab[2]を用いる. MeCabはオープンソースの形態素解析エンジンであり, 日本語文章の分かち書きによく用いられる. MeCabでは様々な種類の辞書を用いることができ, 読み取った文を解析し, それぞれの単語の品詞などを分割して表示することができる. また, オプションから読み取った文を分かち書きした結果を出力する機能がある. 本研究では, MeCabによって国立高等専門学校のシラバスの情報を分かち書き処理し, 品詞を検知する機能を利用して特定の品詞のみを抽出することで, スキルを表す単語を抽出する.

2.4 n-gram [3]

n-gram（Nグラム）とは, 単語を単位として扱うことを考え, 何らかの列を連続するn個の組の列（nは並べる個数）にした表現である. テキストデータを分析する際に単語や文字の連続性に着目して出現頻度を統計的に扱うために用いられ

[2]https://taku910.github.io/mecab/
[3]https://mojitoba.com/2019/09/23/what-is-ngram/

る．n-gramモデルは感情分析，テキスト分類，テキスト生成など，単語の配列が関係するテキスト解析アプリケーションの多くで役立てられており，自然言語処理における重要な概念の1つである．

テキストデータに対してn-gramを適用する例を示す.「太郎　は　りんご　を　買　った」という形態素による分かち書きが適用された文がデータとして与えられたとする．ここで，この文のn-gramを考える．n=3，すなわち3-gramは連続する3個の単語のまとまりであり，例文の中には以下の4つの3-gramが含まれる．

- 太郎　は　りんご
- は　りんご　を
- りんご　を　買
- を　買　った

同様にして，n=2のときは以下の5つの2-gramが含まれる．

- 太郎　は
- は　りんご
- りんご　を
- を　買
- 買　った

本研究では，単語の連続によって構成される単語を検出するために分かち書き処理を施したシラバスのテキストデータに対して，n-gramを適用する．適用して得られるn-gram群は分かち書きによって失われる可能性のある複数単語で構成された単語も検出可能である．複数単語で構成された単語には，例えば2次元配列（2ー次元ー配列）やif else文（ifーelseー文）などがある．そのため，特に一般語にはないプログラミング特有の表現が頻出するような本研究での処理では，辞書にないデータが処理の過程で各単語に分けられる可能性が高く，n-gramの適用が有用である．複数のシラバスから得られるn-gramの出現頻度を参考にプログラミングスキルを表す単語列の候補を抽出し，スキル群の作成に利用する．

3 提案手法

本研究では，プログラミングスキルの表現方法としてスキルツリーを用いることを提案する．これまでには，教育データ活用の分野で，単元ごとの学習者の能力の可視化のためにスキルツリーに類似した表現が用いられている[5]．2.1節で，プログラミングスキルは細かいスキルに分かれており，また順序や組み合わせなどの「関係」が存在し，またこの「関係」は単純な一覧や目次のような形式では適切に表現できないことを示した．一方，2.2節では，スキルツリーが多数の小さなスキルとサブスキルの集合体として構成されていること，スキルを獲得することで，隣接するスキルを獲得可能になり，分岐したり合流したりするスキルのルートからプレイヤーが獲得するスキルを選択することができることを示した．以上のことから，プログラミングスキルの学習形態とスキルツリーの性質は構造が類似しているといえる．

実際に，プログラミングスキルをスキルツリーで表現することを考える．図1では，解説のためにスキルの組み合わせの構造を図示したが，これがプログラミングスキルをスキルツリーに適用した際のイメージを表しているといえる．まず，if文を書くスキルを獲得するためには，事前にフローチャートや等価演算子，条件式のスキルを獲得し，組み合わせる必要があるため，ここには獲得順序と組み合わせの関係がある．また，if文を書くスキルを獲得することで，サブスキルである「else文」や「else if文」のスキルを獲得可能となる．同時に，for文を書くスキルを獲得していれば，if文とfor文を書くスキルの共通のサブスキルである，「if文とfor文の組み合わせ」のスキルが獲得可能となる．このようなプログラミングスキルの複雑なスキル同士の「関係」は，単純な表形式で表現することは難しいが，スキルツリーを利用することによって表現することが可能になると考える．

また，2.2節ではスキルツリーの形状は，ゲームによって樹状，放射状など様々なものがあり，形状の違いによって開始地点やスキル獲得の順番，組み合わせの仕方が様々に変化するため，ゲームやキャラクターの特色を表現する事ができることを示した．そのため，プログラミングスキルにおいても，その特色を表現することができる適切な表現が存在することが示唆される．ケーススタディでは，プログラミングスキルの表現に適した形状の模索が必要であるため，複数の異なる形状のスキルツリーを作成し，その特性を比較する必要がある．

ここで，一般に「ツリー」と呼ばれる木構造の表現について考える．木構造は，データ構造の一部で一つの要素(ノード)が複数の子要素を持ち，子要素が複数の孫要素を持ち，という具合に階層が深くなるほど枝分かれしていく構造のことである．木構造を構成する要素は「ノード」，ノード間の繋がりを「エッジ」と呼ぶ．繋がったノード同士は親子関係を持ち，親を持たない始祖のノードを「根ノード」という．根ノードに近いノードが「親ノード」で，遠い側が「子ノード」であ

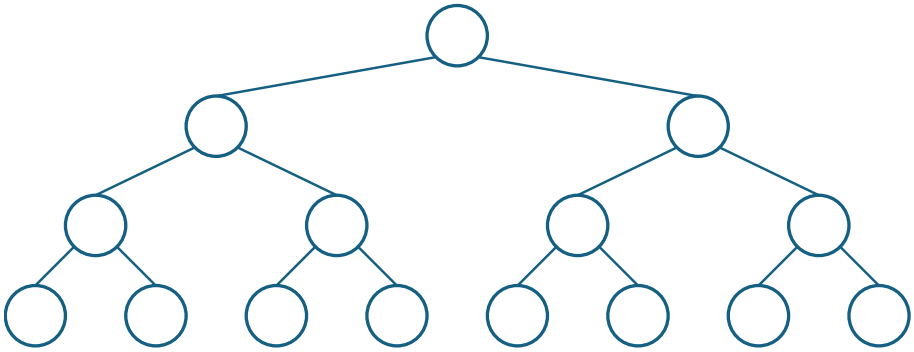

図 3 木構造表現の例

る．図3に，木構造による表現の例を示す．木構造表現の特徴として，親ノードは複数の子ノードを持つことができるが，子ノードはただ一つの親ノードを持つことしかできない．この特徴により，すべてのノードのそこに至るまでの親子，孫などの関係がただ一つであるため分かりやすい，深さや幅などを明確に算出できるため，計算や探索に用いることができるなどの利点がある．木構造の形式を用いれば，プログラミングスキルのある程度詳細なスキルや順序の「関係」があるスキルを表現することができるといえる．しかし，組み合わせの「関係」を表現する場合，親ノードを複数持つ子ノードの表現が存在することになり，木構造という表現の範疇を超え，利点も失ってしまう．このため，木構造による表現は，プログラミングスキルの表現としては適切ではないことが示唆されるため，木構造については，本研究のケーススタディでは扱わないこととした．

本研究のケーススタディでは，樹状と放射状の2種類の異なる形状のスキルツリーを作成し，その特性を比較する．スキルツリーは木構造表現とは違い，親ノードを複数持つ子ノードの表現も可能であり表現方法に制約はないため，スキルの詳細さ，学習の順序や組み合わせなどの「関係」を自由に表現することができる．樹状のスキルツリーは，図3に示した木構造表現と視覚的に類似しており，始点が最上部，または最下部に置かれている場合が多い．始点から順に上から下に，または下から上に向かってスキルを獲得していくが，木構造表現とは違い親ノードを複数持つ子ノードの表現が可能になっている．放射状のスキルツリーは，中心に始点が置かれており，始点から順に獲得するスキルが広がっていくような形でスキルを獲得する表現の形式となる．図4に，放射状のスキルツリーによる表現の例を示す．

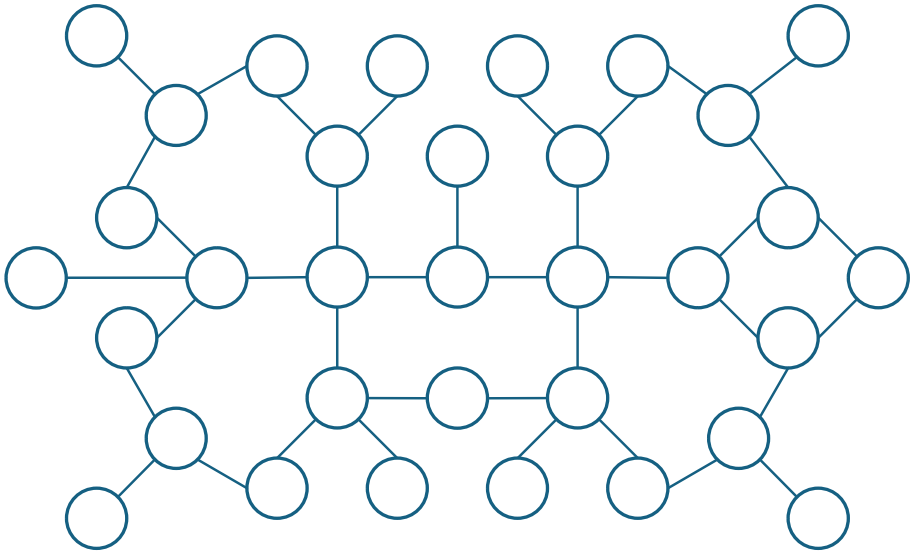

図 4 放射状のスキルツリー表現の例

4 ケーススタディ

本章では提案手法を用いて，プログラミング講義で学習するプログラミングスキルをスキルツリーとして表現するケーススタディについて説明する．

4.1 スキル・サブスキル群の収集

本ケーススタディでは複数高専の講義内容からスキル群を収集する．国立の高等専門学校で用いられている情報系学科のプログラミング系入門科目のシラバスからデータを収集する．文部科学省の全国の国公私立高等専門学校の学科一覧[4]より，国立高等専門学校の情報系に分類される学科，および複合系でコース名に「情報」とつくコースを抽出する．各学科・コースで最初に行われるプログラミング系科目を入門科目と見なし，それらの科目のシラバスを国立高等専門学校機構の高専 Web シラバス[5]で参照し，授業計画から「授業内容」および「週ごとの到達目標」の項目を抜き出す．図 5 に，2024 年度の奈良高専の情報工学科で第二学年を対象に開講されているプログラミング初学者向け科目の一つである「プログラミング 1」のシラバス[6]内の授業計画を示す．

抜き出した 42 高専，44 学科，47 科目のシラバスデータを自然言語処理によって解析し，サブスキルの候補として利用する．最初にシラバスデータに対して MeCab を用いた分かち書きを行う．MeCab で用いる辞書には，unidic-lite を使用する．分割した単語のうち，名詞のみを抽出した上で，以下のストップワードに該当する単

[4] https://www.mext.go.jp/a_menu/koutou/kousen/tokushoku/001.htm
[5] https://syllabus.kosen-k.go.jp/Pages/PublicSchools
[6] https://syllabus.kosen-k.go.jp/Pages/PublicSyllabus?school_id=28&department_id=15&subject_id=0032&year=2023&lang=ja

授業計画

		週	授業内容	週ごとの到達目標
後期	3rdQ	1週	変数と演算子	変数と演算子を理解する。
		2週	if文	if文を理解する。
		3週	switch文	switch文を理解する。
		4週	while文、for文	do文、while文、for文を理解する。
		5週	多重ループ	多重ループを理解する。
		6週	基本型	基本型と型変換を理解する。
		7週	中間試験	授業内容を理解し、正しく解答する。
		8週	試験返却と解説	自身の答案を見直し、理解が不十分な点を解消する。
	4thQ	9週	配列	配列を理解する。
		10週	多次元配列	多次元配列を理解する。
		11週	メソッド	メソッドを理解する。
		12週	文字列	文字列の仕組みを理解する。
		13週	総合演習	ここまでの知識を組み合わせて使うことができる。
		14週	総合演習	ここまでの知識を組み合わせて使うことができる。
		15週	期末試験	授業内容を理解し、正しく解答する。
		16週	試験返却と解説	自身の答案を見直し、理解が不十分な点を解消する。

図5 奈良高専プログラミング1 授業計画

語を候補から除外する．ここで，これらの単語はプログラミングスキルのサブスキル候補として利用される可能性が低い単語や，シラバス特有のスケジュールや説明に関連する単語であり，コンピュータによる処理で結果に不要なノイズとなるため除外している．

ストップワード一覧

プログラム，プログラミング，前期，後期，期末，試験，シラバス，同上，学習，内容，講義，解説，実習，説明，答案，総合

次に，抽出が完了した単語列に対し，n-gramを適用する．n-gramはn=1〜5とし，またn-gramの適用と同時に，検出したそれぞれのn-gramに対して頻度データを計測する．ここで，n-gramのn=1 5としたのは，1から5-gramまでの適用と同時に頻度データを計測したものを確認したとき，5-gramを超えないと発見できないような頻度が5以上の単語列が，新規に検出されることがないと判断したためである．最後に，頻度が5以上のn-gramの一覧を著者と指導教員の2名が独立して目視で確認し，2名ともがプログラミングスキルと見なしたものを採用する．ここで確認したn-gramの一覧は900項目程度である．2名のうち片方のみがプログラミングスキルと見なしたものは，2名で相談した上でプログラミングスキルと見なすか決定する．なお，本ケーススタディでは言語をJavaとし，Javaを学習するプログラミング系入門科目で取り扱う範囲であるかを選択基準とする．ここで，Javaのみを対象としているのは，今回の研究期間で複数の言語について分析することは難しく，対象を絞ったためである．また，奈良高専の初学者向け科目ではJavaを取り扱っており，今後の研究で授業への反映などに利用する場合に円滑に活用できる可能性を考慮した．初学者の範囲では，言語特有の機能を利用することは少な

く，基礎的な構造が類似している範囲の学習が主となるため，言語を変えてのスキルツリー作成は難しくないと考える．また，ファイル入出力やクラスは範囲外とする．これらの範囲はプログラミング記述能力とは違ったファイルに関連する知識が能力が必要な場合があり，今回のスキルツリー作成の対象であるプログラミング入門科目の範囲では詳しく触れないことが多い，または言語特有の機能や仕組みであるスキルであるために除外する．ここまでの操作で完成したn-gramのリストをサブスキル群としてスキルツリーを作成する．これらの手作業は，スキルツリーを授業などで利用する場合には教員が実施し，カリキュラムおよび言語に合わせてn-gramのリストを再編する必要がある．

4.2 スキルツリーの例

収集したサブスキル群を元に2種類の形状の異なるスキルツリーを作成した．作成は著者の主観によって実施した．それぞれのスキル名については，節4.1で作成したn-gramのリストを参考に，n-gramの出力を組み合わせ，一部改変し，より各項目が表すスキルに沿うように設定した．また，授業や教科書で学習した順序や著者の経験から，スキルやスキル間の順序，組み合わせの関係を作成した．図6に樹状のスキルツリーを示す．それぞれのノードがサブスキルを表し，エッジはノードの「関係」を示す．あるスキルを獲得することで関係がある（隣接する）スキルの獲得が可能になる．また，エッジには向きを有するものが存在し，端点が矢印になっている．矢印の向いているスキルは逆側のスキルを獲得することで獲得可能になるが，矢印の向いているスキルから逆側のスキルは獲得できない．ノードの色が異なるスキルは，そこに向けられている2つ以上の矢印付きのスキルを全て獲得することで獲得可能となる．また，エッジが交差する場合，見やすさのためにエッジの色を変えている場合がある．

このスキルツリーでは，最上部の頂点に始点を設定し，スキルツリーを作成した．始点の名称は便宜的に「プログラミングスキル」とした．スキルの学習は，始点に隣接するスキルから順に学習していくものとする．作成したスキルツリーを見ると，上のサブスキルほどより基本的なスキルが多く，下に進むにつれ，サブスキルの関係の分岐が増えたり，より高度なスキルを扱っている．スキルツリーの上下については，始点を上としてスキルツリーを作成した場合の視覚的な上下を指している．基本的には子ノードが親ノードの下になるようにスキルツリーを作成したが，配置の関係でその限りではない場合もある．そのため，視覚的にサブスキルの学習順序を理解することに適していたといえる．また，サブスキルが配置されたおおよその高さで，それぞれのサブスキルの学習に必要な知識の量や難易度をおおまかに計ることができるといえる．

図7に，同じサブスキル群を用いて作成した放射状のスキルツリーを示す．このスキルツリーでは，始点（プログラミングスキル）を中心としてサブスキルが

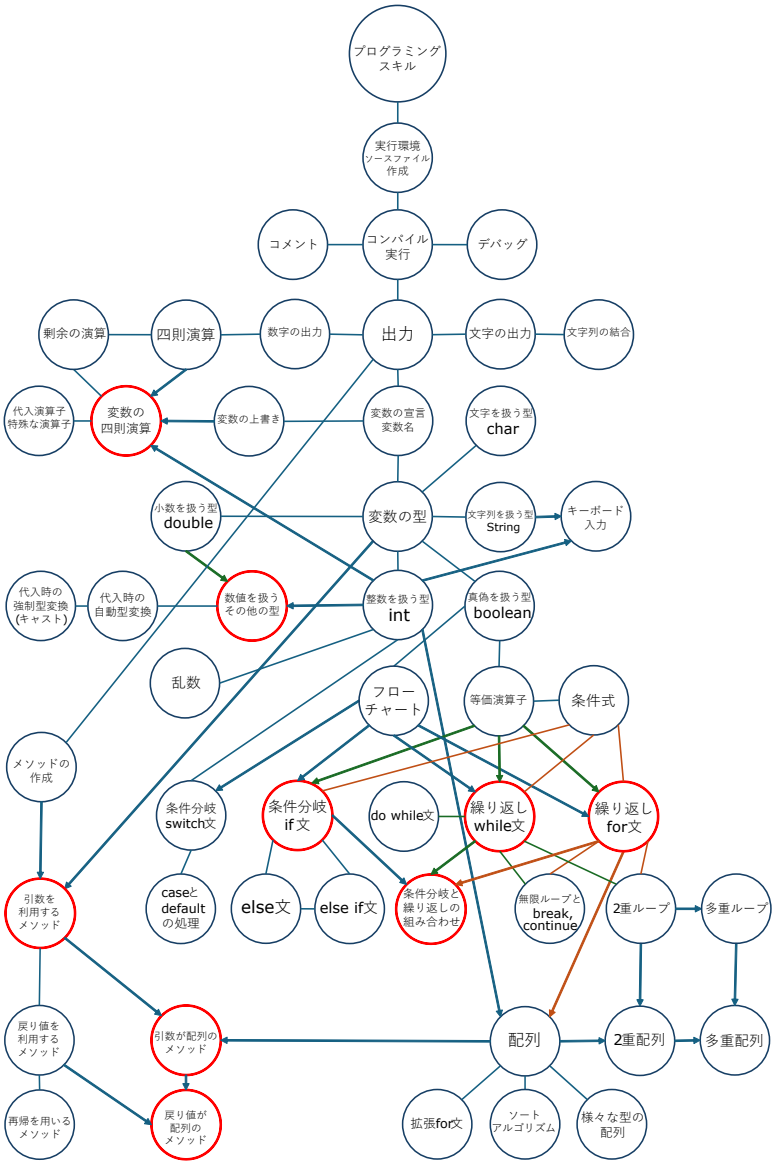

図 6 樹状のスキルツリー

周囲を取り囲む形でスキルツリーを作成した．作成したスキルツリーを見ると，スキルが拡散して広がっていくため，関係の遠いスキル同士が離れ，関係があるスキル同士がより近くにまとまって表現される．そのため，獲得するサブスキルの関連分野（条件分岐，繰り返しや配列，メソッドなど）をより把握しやすくなっていたといえる．また，中央から広がっていくようにスキルを獲得していくため，これまでに獲得したスキルが全体のうちのどの程度の量なのかを獲得した面積から直感的に捉えることができる．一方で，異なる分野同士に存在するサブスキル間のつながりが遠く見えたり，複数のエッジをまたがるエッジがあり，見やすさの観点では樹状のスキルツリーに劣るといえる．また，スキルツリーを作成する際に，見やすさを確保するためエッジが関係のないノード上を交差しないように

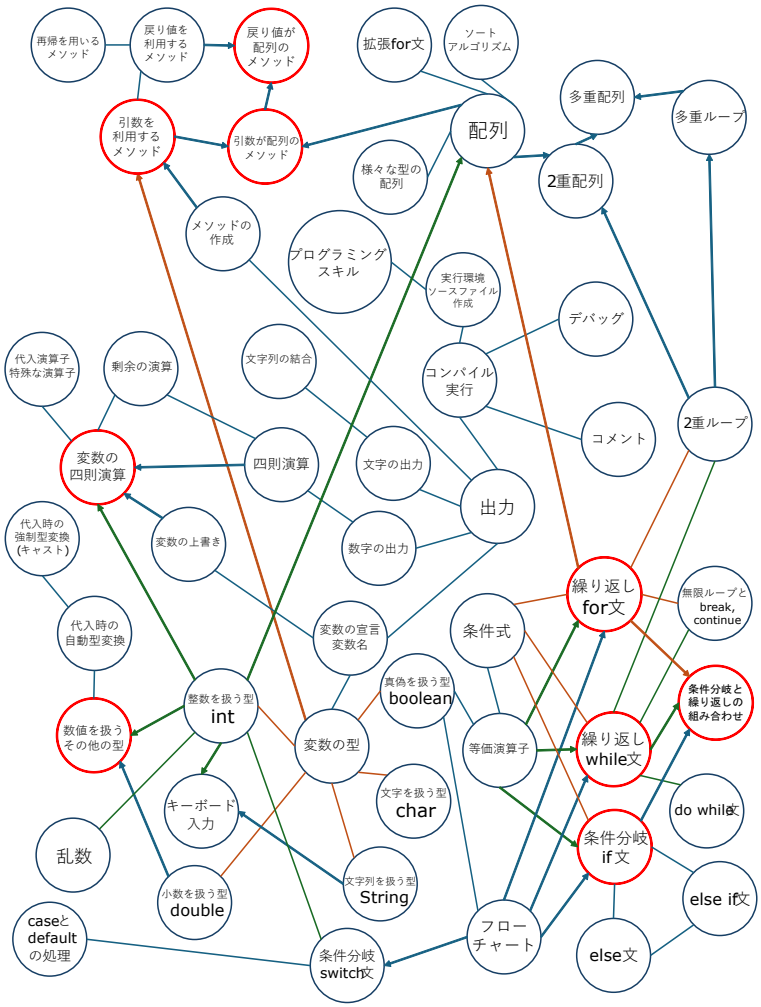

図7放射状のスキルツリー

しなければいけない．そのため，ノードとエッジの配置に気をつける必要があり，シンプルな樹状のスキルツリーに比べ作成にコストがかかるといえる．

作成した2つのスキルツリーを比較すると，樹状のスキルツリーではスキル間の順序関係，放射状のスキルツリーでは組み合わせの関係を表すことに優れていると考えられる．また，見やすさの観点では，樹状のスキルツリーが放射状のスキルツリーに比べ，エッジの交差も少なくよりシンプルに見やすく表現できていた．プログラミングスキルは，特に初学者の範囲では参考書で学習するような単元の内に留まらず，様々なサブスキルの組み合わせが新たなサブスキルの学習につながる．よってこれは，単元内での組み合わせがより把握しやすくなるといえる放射状のスキルツリーと相性が悪かった可能性がある．また，スキルツリーの作成は図表の作成では専門的な知識がない著者が行ったため，図表作成の専門知識を有する作成者であれば，より整理されたり見やすいスキルツリーを作成できる可能性がある．

5 おわりに

本研究では, プログラミング初学者の効率的な学習のために,プログラミングの学習を助け, 理解状況をより詳細かつ正確に表現する方法としてスキルツリーを用いることを提案した. スキルツリーはビデオゲームの分野で用いられている概念で, ゲーム内のキャラクターの能力をどのように育成することができるか, また, これまでどのように育成したかを示す図表の総称である. スキルツリーは, 獲得する能力を選択したり, 獲得した能力の組み合わせで新しい能力を獲得したりすることができる性質をもつ. ここで, スキルツリーの性質に注目したとき, プログラミングスキルの学習形態と構造が類似しているといえる. また, 表形式や木構造では実現できない構造の表現もできるためプログラミングスキルの視覚的な表現方法として効果的と考えた. ケーススタディでは実際にスキルツリーを作成した. プログラミングスキルのサブスキル群を実際にプログラミング教育で用いられているデータから抽出するため, 国立高等専門学校全体のシラバスデータを調査し, プログラミング系入門科目のシラバスデータを収集した. 収集したシラバスデータに対し, 分かち書きとn-gramによる自然言語処理を施し, シラバス上の名詞データを取り出した. 名詞データの頻度順データから, シラバスで頻出した名詞データで今回取り扱う範囲であるJava言語の入門範囲に関連する単語を著者と指導教員によるチェックで抜き出し, スキルツリー作成に用いるプログラミングスキルのサブスキル群の参考とした. 参考データをもとに, サブスキル群とその「関係」を構成し, 形状の観点から2種類のスキルツリーを作り比較した.

作成したスキルツリーでは, スキル同士の学習順序や組み合わせによる新たなスキルの学習の「関係」を表現することができたといえる. 形状を比較すると, 樹状のスキルツリーでは, 単純に上から下に向かってほとんどのスキルの学習順序が進むため, 学習順序を理解するのが容易であった. また, スキルの高さからそれぞれのスキルの学習に必要なスキルの量やおおよその難易度を計ることができた. 一方, 放射状のスキルツリーでは, 中央から学習が始まり, 分野ごとに拡散していくため, より近い分野のスキルを把握しやすいといえた. また, 学習した量が全体のうちのどの程度かを, 獲得した面積から直感的に捉えることができると感じた. ただ, 初学者の範囲では, 複数分野に「関係」を持つスキルも多く, 放射状のスキルツリーはエッジの交差による見づらさや, 作成コストの高さで樹状のスキルツリーに劣ると感じた. これらのことから, 同じサブスキル群とその「関係」を用いて形状の違うスキルツリーを作成すると, それぞれのスキルツリーが得意とする表現や構造に違いがある事がわかった.

本研究の今後の発展として, 現状の作成方法では, スキルツリーの作成の段階が作成者の主観に委ねられるため, 作成者によって結果が左右される可能性が高

いことが予想される．そのため，サブスキル群やその「関係」，スキルツリーの構造についてより正確に設定するために基準を設けること，より客観的な基準を設けるため，本研究のケーススタディをもとに複数人に対してサブスキル群やその「関係」，スキルツリーの構造に関するインタビューを行うことが挙げられる．インタビューの結果から，サブスキル群の詳細さやスキルツリーの構造を変化させる，それぞれのノードに割合を持たせるなど特異な形状を導入するなどの工夫を導入したり，スキルツリーの作成自体を複数人で行うことで，より初学者の学習で効果的なスキルツリーを作成できると考える．スキルとその「関係」から，スキルツリーを自動的に描くことを考えた場合には，スキル同士の近さは視覚的に重要である．幅優先探索によりそれぞれのスキルの始点からの最短距離を算出することで，始点からの距離が遠いほどスキルが離れていくようなアルゴリズムや，共通の親子関係の数が多いスキル同士をより近くするようなアルゴリズムを用いることで，スキル同士の近さを表現できる可能性があると考える．親子関係は隣接するスキルだけでなく，距離2以上のスキルについても重みを減らしてアルゴリズムに組み込むことで，より良い表現ができる可能性がある．また本研究では，サブスキル群の収集でシラバスデータを用いた抽出を実施したが，参考書など他媒体から抽出したデータを用いたり，組み合わせたデータを用いたりすることでサブスキル群が変化する可能性がある．実際のプログラミング学習で活用する場合には，授業の中で各単元や課題，テストなどとスキルツリーを組み合わせ，履修状況や得点とスキルツリーを連携して学習者のプログラミングスキルの自動判定をシステム化できれば，初学者自身のスキルの把握や，理解が不足している要素に対応した支援でより効果的に機能すると考える．また，対象者の範囲を広げ，プログラミング上級者範囲なども含めると，より専門的であったり分野が分かれた学習をするため，効果的なスキルツリーの構造も変化する可能性がある．

謝辞

本研究で，研究の実施や論文作成に関するコメントやアドバイスでご尽力いただきました上野准教授に，厚く御礼申し上げます. また，査読対応をしていただいた岩田准教授には，研究の妥当性や文の構成など不足していた箇所に対する貴重なご意見をいただきました．深く感謝いたします.

参考文献

[1] 爲近瑛太, 納富一宏, "コーディングシーケンス分析によるプログラミングスキル判定", バイオメディカル・ファジィ・システム学会大会講演論文集34, バイオメディカル・ファジイ・システム学会, pp.2-4 (2021).

[2] 大澤佑至, 高岡詠子, "Javaプログラミング単位認定型e-Learning授業におけるプログラミングスキルに関する解析", 情報処理学会研究報告コンピュータと教育(CE), Vol.64, pp.87-94 (2008).

[3] 田口浩, 糸賀裕弥, 毛利公一, 山本哲男, 島川博光, "個々の学習者の理解状況と学習意欲に合わせたプログラミング教育支援", 情報処理学会論文誌, Vol.48, No.2, pp.958-968 (2007).

[4] 保田明夫, "形態素解析と分かち書き処理", テキスト・マイニング研究会, pp.145-180 (2006).

[5] 緒方広明, 堀越泉, "LEAFシステムの活用からみる今後の初等中等教育における教育データ利活用のあり方の提案", 教育データの利活用に関する有識者会議（第10回）, 文部科学省 (2023).