「教育臨床総合研究24　2025研究」

コンピュテーショナル・シンキングにおける抽象化能力の評価方法

A Study on Methods of Assessing Abstraction Skills in Computational Thinking

山　川　大　輝*　　深　見　俊　崇**
Daiki YAMAKAWA　　Toshitaka FUKAMI

要　旨

日本の小学校プログラミング教育で求められるプログラミング的思考は，コンピュテーショナル・シンキング（CT）の考えに基づいたものである．CTの本質である抽象化は，類似点の抽出や本質的ではない情報の無視，パターン認識やモデリングなどの多様な側面があり複雑であるため評価することが難しい．そこで，本研究では，CTにおける抽象化の定義と抽象化能力を測るフレームワークやリアルタイム評価ツールといった抽象化能力の評価方法を確認し，それらの課題をふまえScratchを用いた実践での誤ったアルゴリズムの修正活動を評価に用いる方法を検討した．抽象化能力を評価する際には，抽象化が適切に表現できる課題設定と児童・生徒が作成したプログラムを直接評価することが重要となる．

〔キーワード〕　プログラミング教育，コンピュテーショナル・シンキング，抽象化，評価

Ⅰ　はじめに

文部科学省（2020）の『小学校プログラミング教育の手引（第三版）』では，小学校におけるプログラミング教育の導入について，「情報化の進展による社会や人々の生活の変化，それに伴う将来の予測が困難な社会では，情報や情報技術を主体的に活用していく力や情報技術を手段として活用していく力が重要である」（p.8）と指摘している．この指摘を受け，小学校プログラミング教育の導入は検討された（文部科学省 2020, p.8）．以上の背景をふまえ，文部科学省（2020）は，「コンピュータを理解し上手に活用していく力を身に付けることは，あらゆる活動においてコンピュータ等を活用することが求められるこれからの社会を生きていく子供たちにとって，将来どのような職業に就くとしても，極めて重要なこととなっています」（p.1）と述べている．このように，コンピュータに対する理解とそれを活用する力は今後の社会を生きていく上で必須の能力であること，またそうした必須の能力を獲得するためのものとして，小学校におけるプログラミング教育が位置づけられている．

小学校プログラミング教育の内容について，文部科学省（2016）の「小学校段階におけるプログラミング教育の在り方について（議論の取りまとめ）」では，「子供たちに，コンピュータ

*島根大学大学院教育学研究科教育実践開発専攻
**島根大学教育学部

に意図した処理を行うよう指示することができるということを体験させながら，将来どのような職業に就くとしても，時代を超えて普遍的に求められる力としての『プログラミング的思考』などを育むことであり，コーディングを覚えることが目的ではない」と述べている（文部科学省 2016）．このように，日本のプログラミング教育は，プログラミングに関する技能を身に付けることが主たる目的ではなく，あくまでプログラミング的思考という思考方法を身に付けることが重要だと考えられている．ここで挙げられたプログラミング的思考とは，「自分が意図する一連の活動を実現するために，どのような動きの組合せが必要であり，一つ一つの動きに対応した記号を，どのように組み合わせたらいいのか，記号の組合せをどのように改善していけば，より意図した活動に近づくのか，といったことを論理的に考えていく力」（文部科学省 2016）と定義されている．

このプログラミング的思考の定義については，「いわゆる『コンピュテーショナル・シンキング』の考え方を踏まえつつ，プログラミングと論理的思考との関係を整理しながら提言された定義である」（文部科学省 2016）と補足されている．この記述から，プログラミング教育が獲得を目指すプログラミング的思考のベースとなった考え方がコンピュテーショナル・シンキング（以下CT）と捉えることができる．ところが，林（2018）は，プログラミング的思考の議論においてどのようにCTをふまえたかや詳細な検討がなされた記録もないため両者の関係性が曖昧なままに留まっていると指摘している．そこで，まずCTとは何かを確認する．

今日展開しているCTの言説の多くはWing（2006）のエッセイから広がったものである（林 2018, p.166）．このエッセイについては，中島（2015）の正式な日本語訳が存在するため，この訳を基に内容を確認していく．この訳ではCTを計算的思考と表現しているため，以下のエッセイ内容の説明については計算的思考と表現する．

Wing（2006）は，計算的思考について次のように説明している（中島 2015, p.584-585）．

1. 計算論的思考は計算プロセスの能力と限界の上に成立しているもので，計算の主体が人間であるか機械であるかは問わない．
2. 計算論的思考は，コンピュータ科学者だけではなく，すべての人にとって基本的な技術である
3. 計算論的思考は問題解決，システムのデザイン，そして基本的なコンピュータ科学の概念に基づく人間の理解などを必要とする．
4. 計算論的思考とは再帰的に考えることであり，並列処理であり，命令をデータとし，データを命令とすることである．
5. 計算論的思考とは巨大で複雑なタスクに挑戦したり，巨大で複雑なシステムをデザインしたりするときに，抽象化と分割統治を用いることである．
6. 計算論的思考とは予防，防御，そして最悪のシナリオからの復帰という観点を持ち，そのために冗長性，故障封じ込め，誤り訂正などを用いることである．
7. 計算論的思考はヒューリスティックな推論により解を発見することである．

これらのとおり，Wing（2006）は，CTに対する端的な定義の記述をしておらず，CTとは

何であって何でないのかを列記した外延的措定をするに留まっている（林 2018, p.167）．このようなCT自体の定義が曖昧であるという特質から，CTには様々な構成要素が存在しており，研究者間で構成要素が異なっている．Shute et al.（2017）は，CTの構成要素として分解，抽象化，アルゴリズム，デバッグが文献の中で最も頻繁に出てくる構成要素であると示している（Shute et al. 2017, p.151）．その中でも，Wing（2008）はCTの構成要素について，抽象化がCTの本質であると指摘している（Wing 2008, p.3717）．

本稿では，CTの中心的概念である抽象化を概観し，抽象化能力を評価するための方法を考察する．

Ⅱ　CTにおける抽象化

CTの中心概念が抽象化であることを確認したが，本章では抽象化とは何かを確認し，CTにおいて抽象化がどのように捉えられているのかを概観する．

前提として，Mirolo et al.（2021）は以下の3点を挙げている（Mirolo et al. 2021, p.617）．

1. 抽象化という概念は，多種多様な思考プロセスに適用される．
2. 何らかの形で，事実上すべての有用な学習は，偶発的な経験からある種の抽象化を意味し，教育者によってもその意味は異なる．
3. ラテン語の語源から，抽象化とは基本的に「引き出されたものや引き離されたもの（something pulled or drawn away）」を意味する．

Mirolo et al.（2021）は，これらの前提をふまえ，類似点の抽出と本質的ではない特徴の無視を抽象化の基本的な意味として挙げている（p.617-618）．類似点の抽出に関して，これが抽象化の最も基本的な意味であり，異なる対象に共通する特徴を認識し，そのような特徴を持つすべての対象をグループ化する新しいカテゴリーを作成または定義するプロセスを指す（Mirolo et al. 2021, p.617-618）．このプロセスはCTの文脈で，抽出と呼ばれる（Cetin and Dubinsky 2017, p.71）．この抽出の具体例としては，童謡の類似点を探ったり，その構造を説明するアルゴリズミックな（特に反復的または再帰的な）パターンを探したりすることが挙げられる（Mirolo et al. 2021, p.618）．一方，本質的でない特徴の無視については，類似点の抽出と関係した内容である．類似性を抽出する際，他の特徴は本質的でないとして無視される（Mirolo et al. 2021, p.618）．これに関して，児童がペアで取り組むアンプラグドな抽象化課題がある（https://www.barefootcomputing.org/resources/abstraction-unplugged-activity）．ある児童がカードの束から名詞を読み取り，それが何を表しているかを相手に当ててもらうために，素早くスケッチを描いてもらうというものである．この実践の中で児童は，重要ではない詳細を無視し，最も重要なものだけを含めることを学ぶことができる．

次に，特にCTにおいて抽象化がどのように捉えられているかを確認する．Wing（2008）は，CT における抽象化は，2つの点で数理科学や物理化学の抽象化と比べて豊かで複雑になる傾向があると述べている（Wing 2008, p.3717）．まず1点目について，CTにおける抽象化では，数学や自然科学での抽象化とは異なり，必ずしもエレガントで定義しやすい代数的特性を必ず

しも享受していない（Wing 2008, p.3717-3718）．たとえば，コンピューティングの分野では，スタックという特定の形式でデータを保持することがあるが，整数のように，スタック同士を足すということはできない（Wing 2008, p.3717-3718）．2点目については，CTにおける抽象化は最終的に物理世界の制約の中で動作するように実装されるため，ディスクが一杯になったり，サーバーが応答しなくなったりした場合はどうなるのかといった，不具合が発生した場合のことも考慮しなければならない（Wing 2008, p.3718）．

そして，Shute et al.（2017）は，さまざまな文献の定義をふまえてCTにおける抽象化とは，問題や解決策の中にパターンを見つけることだと説明している（Shute et al. 2017, p.151）．また，Shute et al.（2017）はCTについてのフレームワークも作成しており，そこでは抽象化を複雑なシステムの本質を抽出することとし，3つのサブカテゴリーを挙げている（p.153）．

（a）データ収集と分析：複数の情報源から最も関連性の高い重要な情報を収集し，多層的なデータセット間の関係を理解する
（b）パターン認識：データ／情報構造の根底にあるパターン／ルールを特定する
（c）モデリング：システムがどのように動作しているか，あるいはシステムが将来どのように機能するかを表すモデルやシミュレーションを構築する．

このほかに，Mirolo et al.（2021）は，CTにおける抽象化の多くの運用上の定義は，プログラミングと関連していると指摘し（p.625），重要な側面として，プログラムを用いて抽象概念を具現化することがCT特有のものであることを述べている（Mirolo et al. 2021, p.626）．また，Cetin and Dubinsky（2017）は，コンピュータ・サイエンスにおける抽象化の最も一般的な意味は抽出であることを指摘している（p.71）．CTとコンピュータ・サイエンスは同一のものではないが，いずれもコンピュータに深く関わるものであるため，この点についても考慮すべきである．

Ⅲ　CTにおける抽象化能力の評価

ここまで，抽象化の定義やCTにおける抽象化がどのように捉えられているのか概観してきた．CTにおいて抽象化は中心的概念であるが，その内容はここまで述べてきたように非常に複雑なものである．しかし，CTの定義や概念が多様であることから，CTについて広く受け入れられている評価がないという問題がある（Shute et al. 2017, p.149）．研究者は様々な研究において独自のCT尺度を開発し適用する傾向があるが，CTの知識またはCTに対する態度のために最も一般的に使用される評価方法はアンケート調査である（Shute et al. 2017, p.149）．アンケート調査については，コンピュータ・サイエンスに対する態度や，配列やループのようなプログラミングの構成要素に関する概念的理解，CTに対する理解や将来CTを教育に取り入れることに対する態度などを問うものなどが存在している（Shute et al. 2017, p.149）．だが，プログラムを用いて抽象概念を具現化することがCT特有のものである（Mirolo et al. 2021, p.626）との指摘からも，抽象化能力をアンケート形式で評価することができるのかについては疑問が残る．本章では，CTにおける抽象化能力の評価方法としてアンケート形式以外でど

のようなものが存在しているのか確認する．

まず，抽象化の能力を評価するための方法として，専用のフレームワークを作成するというものがある．クロアチアでは，カリキュラムの学習成果を反映したCTモデルが，エビデンス中心の設計アプローチを用いて作成された．Bubica and Boljat（2022）の調査では，作成したCTの評価デザインパターンが，プログラミング言語や児童・生徒の性別に関する以前の経験とは無関係に，抽象化とアルゴリズム的思考の概念の理解を評価するのに適していることが示された．以下はクロアチアにおけるCT評価のうち，特に抽象化に関わるデザインパターンである（Bubica and Boljat 2022, p.440）．

（A1）問題の基本的な特徴を記述する
（A2）与えられた問題の限界を特定する
（A3）論理式を評価する
（A4）論理演算子を使用する
（A5）与えられた条件（問題）に対する論理式を作成する
（A6）アルゴリズムや解決策における定数と変数を区別する
（A7）アルゴリズムに変数を適用し，問題の変数特性をモニターする
（A8）アルゴリズムや解法における変数の値の変化を定義し，モニターする

Bubica and Boljat（2022）は，このデザインパターンを用いて，抽象化能力を評価するための問題を複数作成している．問題の内容は，ミツバチのMayaに指示を与え，ゴールにたどり着かせるというものである．ここでは，問題例を2点紹介する．1点目は以下のような内容である（Bubica and Boljat 2022, p.457）．

Mayaは, 提示された迷路（図1）の中から自分の好きな黄色い花を探すことになりました．Mayaは花がどこにあるか知っていますが，すべての障害物，グレー色のマス（通ることができないマス），そして迷路の端がどこにあるか知りません．

Mayaが好きな花を見つけるのを手伝ってあげてください．Mayaは，Action Labyrinth（図1）に表示されているすべての手順を踏んで花にたどり着きます．Action Labyrinthでは，Action Walk（図1）も使用します．

Mayaは，図1のように，常に1列目と1行目から動き始めます．Mayaの視線の方向は関係ありません！

Mayaはいつも迷路を進む際に，まずAction Labyrinthを使用します．そして，Action Labyrinthの中でさらにAction Walkが使用されます．

Action Labyrinthとその中のAction Walkを使って迷路を移動した場合，Mayaは5歩歩いた後，どの位置（行，列）にいるでしょうか？

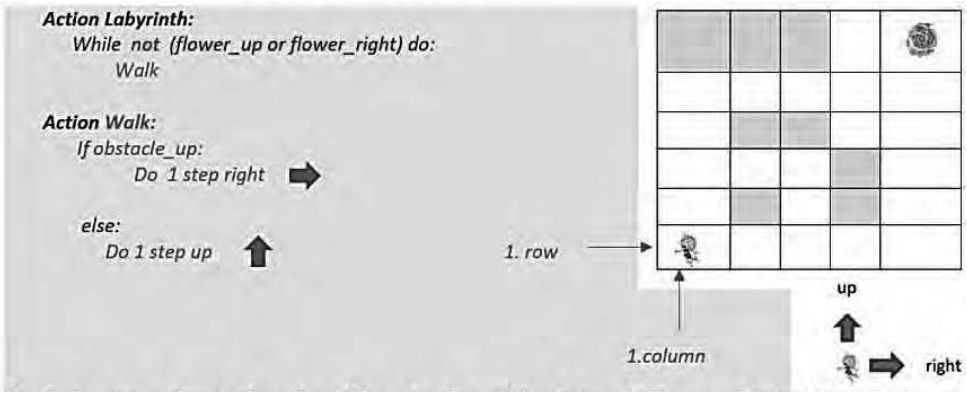

図１　Action Labyrinth ,Action Walk と迷路
（Bubica and Boljat 2022, p.457 より引用）

図1のとおり，Action Labyrinthでは花が上または右になければAction Walkを行う．そして，Action Walkは障害物が上にある場合に右に1歩進み，そうでない場合には上に1歩進む．処理の順番はAction Labyrinthを始めに行い，その後にAction Walkを行う．このような条件をふまえて最終的にMayaがどこにいるのかを問う問題となっているが，この問題は抽象化能力のうち，問題を理解する力を見取るための問題であり，これは先に挙げたデザインパターンのうち，A1とA3を組み合わせたものである（Bubica and Boljat 2022, p.441）．

2点目は図2の選択肢の中で，Mayaが花にたどり着けないものを選ぶ選択式の問題である．この問題の中で，MayaはAction Walkを行うが，それは前述したものと同じ動きであり，上と右にしか進むことができない（Bubica and Boljat 2022, p.457）．この問題は，問題の主要な特徴（制約）を特定する抽象化能力を評価する問題であり，A2 + A3の組み合わせによって設定されている（Bubica and Boljat 2022, p.441）．

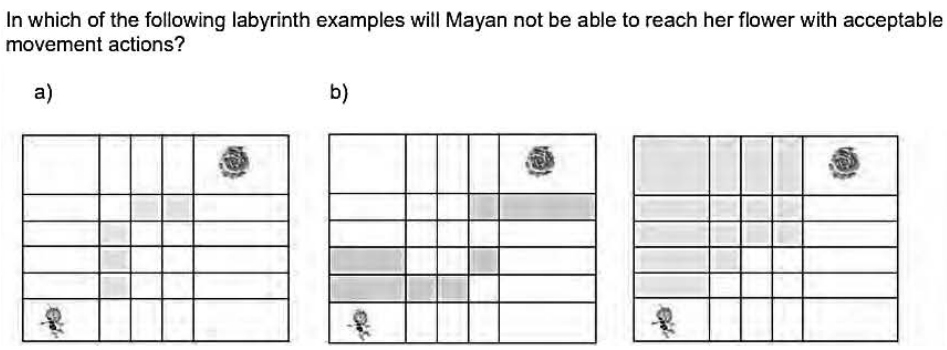

図２　ゴールにたどり着けないものを選択する問題
（Bubica and Boljat 2022, p.458 より引用）

また，教室におけるCTの評価の難しさに関しては，教師の指導をサポートするために，児童・生徒ごとの学習進捗データを提供するリアルタイム評価が重要だという指摘も存在する（Shute et al. 2017, p.156）．この指摘について，児童・生徒の問題に取り組む姿や反応から抽象化能力を評価するということがリアルタイム評価の例として示されていたが，このような方法の他に，Scratchプログラムを活用した学習の場合，Dr.Scratch（https://drscratch.org/）を活用するという方法も存在する．Dr.Scratchは，抽象化と問題分解，論理的思考，同期化，並列化，フロー制御のアルゴリズム的概念，ユーザー対話性，データ表現という7つの基準に基づいて，CTを評価する（Kown et al. 2018, p.3）．Dr.Scratchのページに作成したScratchのプロジェクトのURLを入力することで，CTに関わる能力について項目ごとにフィードバックが出力される．フィードバックは即時的であり，URLを入力してすぐCTのスコアを確認することができる．また，評価についてはBASICとDEVELOPINGの2種類があり，異なったレベルの評価を行うことができる．図3にDr.Scratchのフィードバック画面を例示する．

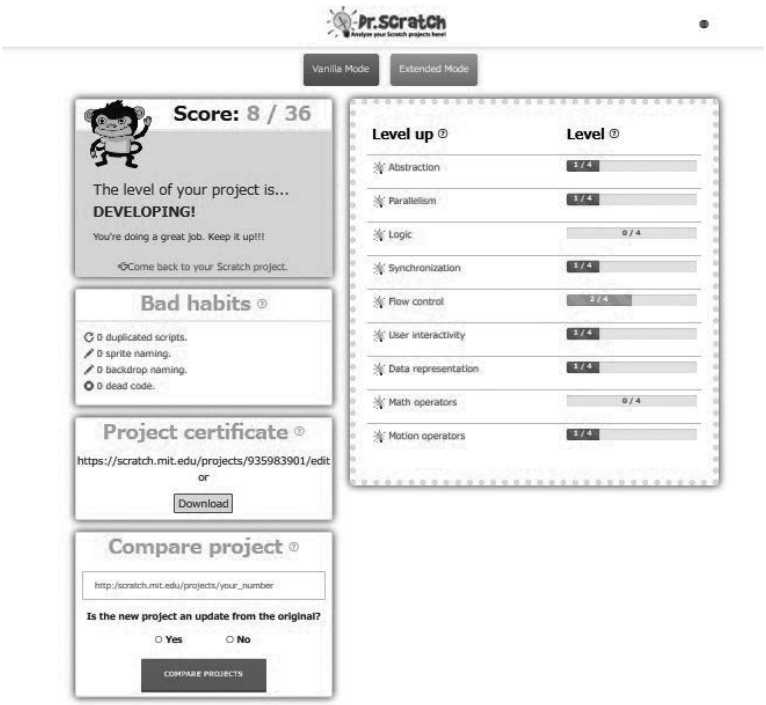

図３　Dr.Scratch のフィードバック画面
（https://drscratch.org/）より

本章では，CTの抽象化能力を評価するための具体的な方法について確認してきた．抽象化能力を評価する方法には，専用のフレームワークを作成し，そのデザインパターンを用いて抽象化能力を評価するための問題を作成するという方法（Bubica and Boljat 2022）やScratchのプロジェクトのURLを入力することでCTに関わる能力について項目ごとに即時的なフィードバックを行うDr.Scratchを活用したリアルタイム評価（Kown et al 2018）が挙げられる．

Ⅳ　CTにおける抽象化能力の評価方法の提案及び考察

CTの抽象化能力を評価するための具体的な方法について確認してきたが，先に挙げた2つの評価については課題が存在する．本章では，Ⅲ章で確認した評価方法に対する課題を確認し，その課題に対しての評価の提案を挙げる．

Wing（2014）はCTの定義について，問題を定式化し，その解決策を，コンピュータが効果的に実行できるような方法で表現するための思考プロセスであると指摘している（Wing 2014）．Bubica and Boljat（2022）は抽象化を評価するにあたってデザインパターンを用いて，抽象化能力を評価するための問題を複数作成している．問題の内容は，前述したとおり，Action LabyrinthとAction Walkという条件が与えられた上で，5歩先のMayaがどの地点にいるのか，問題文を参照してゴールにたどり着くことができないものを選択するといったものである．このような課題を設定する場合，CTの定義のうち問題を定式化するという部分は達成できると考えられるが，コンピュータが効果的に実行できるような方法で表現するという部分については実現できていない．一方，Dr.Scratchについては定量的なスコアを提供するが，CT概念の理解を評価するには十分ではないという指摘も存在する（Kown et al 2018, p.3）．Dr.Scratchの評価項目の中に抽象化は存在するが，その評価基準は特定のブロックやアルゴリズムに依存してポイントが変動する．内容として，1ポイントを得るためには異なるキャラクターを異なるプログラムで制御している，2ポイントを得るためにはクローンを使用している，3ポイントを得るためにはユーザー定義ブロックを使用している，4ポイントを得るためには”broadcast”，”wait”，”repeat”のようなブロックを少なくとも3つの命令で使用するか，複数の条件を持つ”if”ブロックを使用して，発展的にクローンを使用しているというものである．これはフィードバック画面の抽象化の欄をクリックすることで，評価基準を参照することができる（https://www.drscratch.org/learn/Dimensions/Abstraction/）．このように特定のブロックやアルゴリズムに依存して評価を行う方法では，特定のブロックは使用しているが問題のパターンを特定することができていない，そもそも意図した動作をするプログラムではないといったケースについて，高い抽象化のスコアを与えてしまう可能性がある．

これら評価方法の課題に対する提案として，直接的に抽象化能力を評価したものではないが山川・深見（2024）の実践を挙げる．山川・深見（2024）は，Scratchを利用して，「ライオンを正しくうごかそう」という命令の順序が正しく設定されておらず，うまく動くことのできないライオンを，ブロックを並べ替えて正しく動かそうとする課題を実践した（山川・深見 2024, p.40-41）（図4）．山川・深見（2024）は，この実践をCTのうちアルゴリズム的思考を育むものとして設定している（p.38）．山川・深見（2024）の実践は，直接的に抽象化能力を評価するものではないと述べたが，アルゴリズムと抽象化の関係について，アルゴリズムの定義は入力を受け取り，必要な出力を生成するための段階的な手順の抽象化であり，CTの定義をふまえると，アルゴリズムも一種の抽象化であると認識されている（Wing 2008, p.3718）．

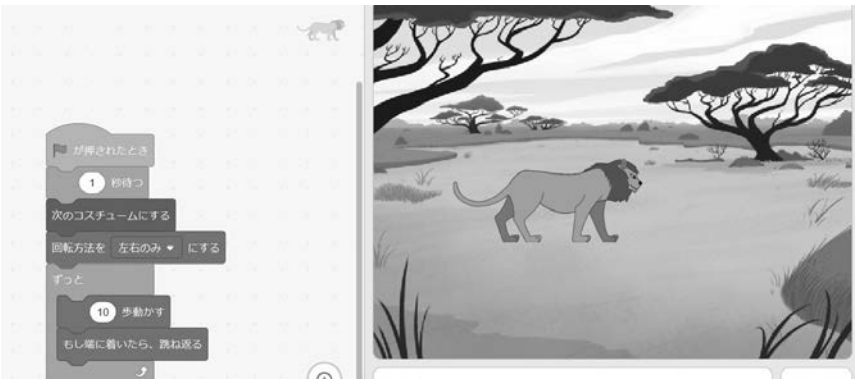

図４「ライオンを正しくうごかそう」
（山川・深見 2024, p.41 より引用）

山川・深見（2024）の課題のように誤ったアルゴリズムの修正すべき部分を分析し，正しいアルゴリズムを検討するという課題は，Ⅲ章で挙げたBubica and Boljat（2022）の評価デザインにおける，（A3）論理式を評価する，（A5）与えられた条件（問題）に対する論理式を作成するという2つを満たすものである．そして，山川・深見（2024）は，この課題を既に類似した問題に取り組んだのちの確認用の課題として設定している（p.40）．このような位置づけの場合，（A1）問題の基本的な特徴を記述する，（A2）与えられた問題の限界を特定する，などの問題から本質以外を無視するという抽象化の側面が獲得できない可能性もあると考えられる．その一方で，既に類似した問題に取り組んだのちに確認するための課題という位置づけの場合，これまでの課題から定式化されたパターンをコンピュータが効果的に実行できるように再度表現することになるため，前述したBubica and Boljat（2022）の評価における課題である表現面を達成することができるとも捉えられる．また，ブロックを並べ替えて，理想とする動きへと近づけていくという活動に取り組むためには，事前に用意されたブロックの意味と理想とされるプログラムの動きを児童･生徒が理解している必要がある。それゆえ，問題のパターンを特定することができていない，そもそも意図した動作をするプログラムではないといったケースの場合でも，高い抽象化のスコアを与えてしまう可能性があるというDr.Scratchの課題についても克服することができると考えられる．

本章ではこれまで取り組まれてきた抽象化の評価方法の課題を確認し，それらの解決策として，山川・深見（2024）の実践を分析・提案を行った．しかし，山川・深見（2024）は，実践における課題として，児童の個人差への対応と評価方法を挙げている（p.44）．児童の個人差への対応については，問題を解くことが遅い児童や早く解けてしまい，指定した学習活動以外の活動を行う児童が見られたこと，アンケートの結果からも個人差が見られたことを挙げ，問題を解くペースが速い児童と遅い児童のいずれも学習活動を行うことができるような課題設定と実践の方法を検討する必要がある（山川･深見 2024, p.44）．また，評価方法については，山川･深見（2024）もアンケート調査を採用しており，児童の制作物から評価を行う方法が適切であったと述べている（p.44）．

今後抽象化を評価するにあたって，児童・生徒が取り組んだプログラムそのものを評価することは不可欠であるだろう。とりわけ，抽象化を評価する場合には，問題のどこを児童が抽象

化するのかということをこちらが意図しておくことが重要である．Wing（2008）は，抽象度の高いものを扱うには，「正しい」抽象度を定義することが重要であると指摘している（Wing 2008, p.3718）．問題の本質にかかわる思考プロセスであるからこそ，何を本質として，何を不要なものとして切り捨てるのかを明確にしなければ抽象化を評価することは困難だからである．

付記

本稿は，山川大輝・深見俊崇（2024）コンピュテーショナル・シンキングにおける抽象化能力の評価の課題．日本教育工学会 2024年秋季全国大会（第45回大会）講演論文集：451-452, の発表内容を大幅に加筆・修正したものである．

参考文献

Bubica, N., and Boljat, I.（2022）Assessment of Computational Thinking – A Croatian Evidence-Centered Design Model. *Informatics in Education*, 21(3): 425-463

Cetin, I., and Dubinsky, E.（2017）Reflective Abstraction in Computational Thinking. *The Journal of Mathematical Behavior*, 47(1): 70-80

Kwon, K.J., Lee, S. 0J., and Chung, J.（2018）Computational Concepts Reflected on Scratch Programs. *International Journal of Computer Science Education in Schools*, 2(3): 2018-09-22, https://doi.org/10.21585/ijcses.v2i3.33.

Mirolo, C., Izu, C., Lonati, V., and Scapin, E.（2021）Abstraction in Computer Science Education: An Overview. *Informatics in Education*., 20(4): 615-639

文部科学省（2020）小学校プログラミング教育の手引（第三版）．https://www.mext.go.jp/content/20200218-mxt_jogai02-100003171_002.pdf（2025年3月29日最終閲覧）

文部科学省（2016）小学校段階におけるプログラミング教育の在り方について（議論の取りまとめ）．https://www.mext.go.jp/b_menu/shingi/chousa/shotou/122/attach/1372525.htm（2025年3月29日最終閲覧）

林向達（2018）Computational Thinkingに関する言説の動向．日本教育工学会研究報告，16：165-172

Shute, V. J., Sun, C., and Asbell-Clarke, J.（2017）Demystifying Computational Thinking. *Educational Research Review*, 22：142-158

Wing, J. M.（2006）Computational Thinking. Communications of the ACM, 49(3): 33-35
訳：中島秀之（2015）計算論的思考．情報処理，56(6)：584-587

Wing, J. M.（2008）Computational Thinking and Thinking about Computing. *Philosophical Transactions. Series A, Mathematical, Physical, and Engineering Sciences*, 366: 3717-3725

Wing, J. M.（2014）Computational thinking benefits society. 40th Anniversary Blog of Social Issues in Computing. http://socialissues.cs.toronto.edu/2014/01/computational-thinking/（2025年3月30日最終閲覧）

山川大輝，深見俊崇（2024）コンピュテーショナル・シンキングに基づく小学校低学年を対象としたプログラミング教育の実践．島根大学臨床総合研究,23：31-46