JSiSE Research Report
vol.33,no.6(2019-3)

プログラミング教育必修化時代の到来に合わせた ET プログラミングの再考

石川貴彦
名寄市立大学保健福祉学部

Rethinking of ET Programming Based on Early Programming Education

Takahiko Ishikawa
Faculty of Health and Welfare Science, Nayoro City University

本研究では，2020 年から開始する小学校のプログラミング教育必修化に合わせて，筆者らが 2002 年に実践してきた宣言型言語・テキストベースの ET プログラミングを再考した．ここでは，文部科学省が公表したプログラミング教育の手引で示されたビジュアル型言語によるプログラム例などを，ET プログラミングで作成・比較することで，本プログラミングが，必修化で目指しているプログラミング教育の方法として適用可能かどうかを検討した．

キーワード: 小学校プログラミング教育の手引，ET プログラミング，ルール，置き換え，意味

1. はじめに

2020 年からの小学校におけるプログラミング教育必修化を見据えて，様々な準備が進んでいる最中である．文部科学省は，2018 年 11 月に「小学校プログラミング教育の手引（第二版）」[1]（以下，手引）を公表し，例示の中で SCRATCH などのビジュアル型プログラミングを採用している．これは，コーディングを覚えるのが目的ではないことと，動きの組み合わせを論理的に考える「プログラミング的思考」を育成することの２つを満たすには，ビジュアル型が適切であるという判断から採用されたものと思われる．しかしながら，単純な動きの組み合わせならばビジュアル型は理解しやすいが，手引で示した自動販売機の例はサブルーチンなどを用いる必要があるため，プログラムが大規模かつ複雑になり，実際の指導は動きの一部を再現するプログラムの作成に留めている．ここまでの段階になるとビジュアル型の優位性は薄れてしまい，プログラムも分散して動きが追いにくくなるので，学習者は理解困難に陥りやすい．

筆者らは，このような手続き型言語・ビジュアルベースのプログラミング教育に依存せず，宣言型言語・テキストベースによるプログラミング（以下，ET プログラミング[2]）教育の実践を 2002 年から独自展開してきた．論理的思考力の育成や問題解決を学ぶことを目的に，等価変換ルール（以下，ET ルール）の記述によるアルゴリズム構築という方法で大学生対象に取り組んできたが，当時のプログラミング教育の主流がコーディング中心だったことや，C++など実用性の高い言語の教育が望まれた時代だったので，筆者らの取組以外での実践はなかった．しかしながら，この ET ルールが手引で言うところの「動きの組み合わせ」に相当し，さらに各ルールに意味を付与していくので，動きの正しさを１つずつ検証しながら論理的に構築することが可能である．この構築がまさに「プログラミング的思考」の育成であると考えている．

そこで本研究では，プログラミング教育必修化の時代に合わせて，過去に実践してきた ET プログラミングを再考し，本プログラミングが現在求められている教育方法として適用可能かどうかを検討することを目的とした．

2. ET プログラミング

2.1 ET ルール

ET ルールは，問題の元の意味を損なわずに，別の問題に置き換えて簡単化するという動きを表現したものである．例えば小学校低学年の算数では，足し算や引き算の方法として，さくらんぼ計算を用いることがある．これは８＋７を計算するときに，後の７の下にさくらんぼに見立てた２つの円を書き込み，前の８と左の円を足して 10 になるような数を考える．次に，後の７から左の円を引いた数を右の円に書き込む．そうすると８＋７は 10＋５という問題を解くことと同意味になる．これが ET ルールの考え方であり，

８＋７　→　10＋５

のように解く問題の意味を変えることなく，計算を簡単化する．問題の置き換え関係は小学校低学年では他にもあり，

５円玉が 50 枚　→　５円玉 10 枚の塊が５つ

といったお金の計算でも同様である．このような関係性を踏まえ，以下に ET ルールのシンタックスを示す．

（述語 引数）→（述語 引数）.

左辺の括弧はヘッドアトムと呼び元の問題を表す．そして，右辺の括弧はボディアトムと呼び，置き換え後の問題を表している．なお，ET ルールには条件部を設けることができ，ヘッドアトムの後に中括弧で囲んだアトム（コンドアトム）を記述する．

（述語 引数),{(述語 引数)} →（述語 引数）.

述語には ET プログラミングのビルトインである B 述語と，ユーザが自由に定義できる D 述語があり，D 述語から B 述語に置き換えていくプロセスが簡単化に当てはまる．引数には数や文字列，変数のほか，リストを扱うことができる．変数は*で始まるシンボルで表現し，リストは[*A|*R]のように表現する．

2.2 過去の ET プログラミング教育の実践

筆者らは，2002 年に大学１年生約 200 名を対象に ET プログラミング演習を実施した．学習内容は，テキストエディタを用いてプログラムを記述したり，インタープリタにプログラムをロードし実行したりするなどのプログラミングに関する基本操作と，数値計算やリスト処理に関するプログラムの作成を中心とした．学習方法は，教師による説明と独自に作成したテキスト，e-Learning システム[3]を併用したブレンディッドラーニングで行った．

数値計算の例では階乗を求める問題があり，

$$\begin{cases} ①\quad 0! = 1 \\ ②\quad N! = N\times(N\text{-}1)! \end{cases}$$

という２つの式について，以下のように ET ルールで記述する．なお fact は D 述語であり，＝，>，:= ，-，x は B 述語である．

R1: (fact 0 *X) → (= *X 1).
R2: (fact *N *X), {(> *N 0)} → (:= *N1 (- *N 1)),
(fact *N1 *X1),
(:= *X (x *N *X1)).

ルールを記述した後に，それぞれ意味を付与しながら置き換え関係の正しさを検証する．*R1* は「０の階乗の答 X は，X＝１に置き換える」という意味を持ち，*R2* は「数 N の階乗の答 X は，N が正整数のとき，N から１を引いた N1 を求める問題と，数 N1 の階乗の答 X1 を求める問題と，N と X1 の積 X を求める問題の３つに置き換える」という意味を持つ．つまり，①から *R1* を作成して意味を与え，同様に②から *R2* を作成して意味を与えるというプロセスから，プログラミングを学習するという方法である．このポイントは，付与した意味の正しさと，想定される問い合わせの範囲を網羅しているかどうか（0 以上の正整数に対して，全て解を求められるか）を検証すれば良く，再帰処理の過程を学習者に追わせないことである．

次に，リスト処理における当時の学習者のプログラム例を示す．与えられたリスト[1 2 3]の逆順のリスト[3 2 1]を求めるプログラムの作成を取り上げる．学習者からは次のような方略が提案された．

(A)先頭要素 1 を除いたリスト[2 3]を逆順にして[3 2]を求め，それと[1]を結合する問題に置き換える．
(B)先頭要素 1 をストック用リストの先頭に挿入し，残

りの[2 3]の先頭要素をストック用リストの先頭に挿入する問題に置き換える．

(C)末尾要素 3 を求めてリストの先頭に挿入し，残りのリスト[1 2]から末尾要素を求めて，リストの先頭に挿入する問題に置き換える．

(A)～(C)の方略は，以下の *RA*～*RC* のルールとしてそれぞれ表現された．なお，reverse，append，restlast は D 述語である．

RA (reverse [*A|*R] *X)
→ (reverse *R *X1), (append *X1 [*A] *X).
RB (reverse [*A|*R] *L *X)
→ (= *L1 [*A|*L]), (reverse *R *L1 *X).
RC (reverse *L *X)
→ (restlast *L *M *R), (reverse *R *X1),
(= *X [*M|*X1]).

RA は「[*A|*R]の逆順のリスト*X を求める問題は，リスト*R の逆順のリスト*X1 を求める問題と，*X1 と[*A]を結合したリスト*X を求める問題に置き換えることができる」という意味を持つ．このルールで問題を置き換えた後に，さらに append を置き換えるルールを追加して，全てのアトムが B 述語で表された問題に置き換えられれば，逆順のリストを求めることができる．演習終了後，学習者から自由記述を求めたところ，「難解な問題を簡単な問題に直して，段階を踏んで解けるようになった」，「部分部分を構成していって，最終的に 1 つのプログラムをとして機能できる楽しさを知った」，「ルールを積み重ねることでプログラムを完成させる方法によって，他の言語でプログラムを作るときにも応用できる考え方が身についた」などといった意見が得られた．このようにして，当時の ET プログラミングは，問題の置き換えを行うルールの段階的な組み合わせにより，学習者にプログラミング的思考の育成機会を提供したのである．

2.3 ET プログラミングの学習方法

前節で述べた過去の実践を踏まえ，ET プログラミングの学習プロセスを以下にまとめる．

(1) 与えられた問題を解くための方略を考える
(2) 方略に基づき ET ルールを記述する
(3) 記述したルールに意味を付与する
(4) ルールを実行し，置き換え後の問題を得る
(5) 置き換え後の問題を簡単化するルールを追加する
(6) 追加したルールに意味を付与する
(7) 問題が全て B 述語に置き換えられるまで(4)～(6)を繰り返す
(8) B 述語に置き換えられたら，問題の解が得られる

この学習プロセスは，ビジュアル型のように部品を配置して直感的にプログラミングを始めるのではなく，(1)で問題を吟味してから(2)でプログラムを書き，(3)でプログラムの正しさを検証するというスモールステップで進めていく．そして，プログラムを完成させて解を得ることを目標とするのではなく，(4)でプログラムを 1 つ書いて途中結果を得て，解に一歩近づくという簡単化を目標とする．そのため，一気にまとめて作るのではなく，(7)のように置き換え関係を組み合わせていきながら，徐々にプログラムを作りあげるのが ET プログラミングの学習方法である．

3. 手引の例題に対する ET ルールの記述

3.1 通電の制御

手引では，小学 6 年理科の電気の単元において，プログラミングを通して学習する場面を挙げている．その中で，照明を効率よく利用するための方法を考える活動が設けられ，センサーの作動設定や制御の手順について試行錯誤しながらプログラムを作成する．図 1 は手引で示されたプログラム例である．

図 1　通電を制御するプログラム例
（手引 p. 26 の図を引用）

図１を見ると，ループ処理と分岐処理を組み合わせた１つのまとまったプログラムであり，この理解は初学者にとって１つの難関になる．ET プログラミングでは，独立した４つのルールで記述できる．

L1: (lamp) → (sensor *X),(switch *X),(lamp).
L2: (sensor *X) → (read *X).
L3: (switch *X), {(<= *X 100)} → (wait on 10).
L4: (switch *X), {(> *X 100)} → (wait off 1).

L1 には「照明のオンオフ(lamp)は，センサー(sensor)の情報*X を受け取り，*X からスイッチ（switch）を制御して，照明のオンオフ（lamp）を受け付ける」という意味を与える．末尾で lamp アトムを呼び出しているのでループ処理になるが，意味を把握させれば，繰り返すという動きを追わせずに理解を促すことができる．次に，*L2* は*X を読み取って sensor に返すルールである．そして，*L3* には「スイッチの値*X が 100 以下のとき，スイッチを入れて 10 秒待つ」という意味を与え，*L4* には「スイッチの値*X が 100 より大きいとき，スイッチを切って１秒待つ」という意味を与える．*L3* と *L4* が分岐処理に該当するが，意味を個々に確認すれば，排他的に捉えることができるので，枝分かれするという動きを意識させずに理解を促す．

3.2 自動販売機

手引に示されている総合的な学習の時間の例として，自動販売機のプログラムの作成がある．ここでは，硬貨の種類の判別や温度管理などの判断を自動で行っていることに気付かせ，自動販売機の動きの一部を再現するプログラムを作成すると述べられている．太田[4]が作成した SCRATCH によるプログラム(図 2)では，サブルーチンを用いて，硬貨の投入，返却，購入ボタンの３つを記述している．図２の規模であっても機能は限定的であり，商品が１種類のみだったり，100 円しか投入できなかったり，おつりの返却ができなかったりする状況である．大規模なプログラムになると，機能別に作成することが必須になるが，これは学習者にとっても，教える教師にとっても，相当難しいプログラミングとなるだろう．他にも，自動販売機のスプライトを教師が用意する必要があり，プログラミングスキルとは別に作画スキルも要求される．

図２　太田による自動販売機のプログラム例

自動販売機のプログラムを ET ルールで作成するには，「自動販売機は，販売している商品を展示し，硬貨を受け付け，投入した金額で購入できる商品を提示し，選択した商品とおつりを返す機能を持つ機械である」ということを，最初に方略として考える．この方略に基づいて以下のルールを記述する．

V1: (jidou *X *R) → (display *D),(slot *C),(buy *D *C *D1),
(give *D1 *X *C *R).

そして，*V1* に「自動販売機で商品*X とおつり*R を受け取るという問題（jidou）は，販売している商品*D を展示（display）し，硬貨*C を受け付け（slot），*D から*C の金額内で買える商品*D1 を提示（buy）し，*D1 から選択した商品*X と，*C のおつり*R を渡す(give)という問題に置き換える」という意味を与える．次に *V1* によって置き換えられた display アトムは，リストを使って以下のように記述できる．

V2: (display *D) → (= *D [[ジュース 130]
[コーヒー 130]
[お茶 100]]), (print *D).

V2 には「展示する商品*D は，ジュース 130 円，コーヒー130 円，お茶 100 円を表示する」という意味を与える．展示する商品を増やすにはリストを長くし，商品や料金以外に在庫数や温度管理の区別も与えるな

らば，[ジュース 130 10 冷]のようにリストの要素を増やすことで拡張できる．以降の slot，buy，give アトムを置き換えるルールは次のように記述する．

V3: (slot *C) → (read *C).
V4: (buy [[*D *P]|*Z] *C *D1), {(> *P *C)}
→ (buy *Z *C *D1).
V5: (buy [[*D *P]|*Z] *C *D1), {(<= *P *C)}
→ (= *D1 [[*D *P]|*D2]), (buy *Z *C *D2).
V6: (buy [ ] *C *D1) → (= *D1 [ ]).
V7: (give *D1 *X *C *R) → (print *D1), (read *A),
(match *A *D1 *X *C *R).
V8: (match *D [[*D *P]|*Z] *X *C *R)
→ (= *X *D), (:= *R (- *C *P)).
V9: (match *A [[*D *P]|*Z] *X *C *R), {(/= *A *D)}
→ (match *A *Z *X *C *R).
V10: (match *A [ ] *X *C *R) → (false).

V4～*V6* は，投入した金額*C と商品の価格*P を比較し，*C が*P より下回る場合は展示する商品から除外し，*C が*P 以上の場合は購入可能リスト*D1 に追加するという動きとなる．*V7*～*V10* は，選択した商品*A と購入可能リスト*D1 の中の商品が一致したとき，*A を*X として返し，*C から商品の代金*P を引いた値を*R に返すという動きとなる．どれも一致しない場合は失敗を返す．このように複数の商品を販売し，おつりを返却する機能を持つ自動販売機のプログラムは 10 ルールで作成でき，手引で示されている一部の作成に制限しなくとも，完成できるサイズに収まる．

3.3 プログラミングの楽しさや面白さ，達成感などを味わえる題材

手引では学校の裁量で時間を確保し，各教科等とは別にプログラミング体験を実施することも想定している．例として，キャラクターを動かしてランダムに降ってくる星を獲得するプログラムを挙げており，ゲーム性を持たせることで，楽しさや達成感を与えるように配慮している．本稿では，ゲーム性を持たせた題材として数理パズルのプログラムを扱う．前述の過去の取組で，筆者らは「パズルプログラミング」という演習を大学生対象に実施しており，覆面算の計算や数独パズルなど，元々ET ルールが得意とする例題を教材化している．教材の１つであるハノイの塔は，図３(1)のように棒Aに刺さった直径の異なる円盤を１枚ずつ移動して，(4)のように棒 C に全て移動するというパズルである．ただし，直径の小さい円盤の上に直径の大きい円盤は置けない制約がある．山田・有吉[5]は，ハノイの塔は小学校でのプログラミング教育が狙いとしているプログラミング的思考の育成に適した教材であると述べており，大学生対象の実践においても，興味深いという意見を得た題材である．しかしながら，解法のパターンは単純なので，ひとたび理解すると簡単でつまらないという意見も同時に得た．つまり，小学校の題材としたとき，導入で児童の興味を引きつけ，パターンがわかると簡単に解けるので，達成感を得やすいプログラミングであると言える．

ハノイの塔を解くためには，(2)のように一番下に刺さっている最も大きな円盤だけが A に残り，それ以外の円盤が B に移動している状態であれば，最も大きな円盤を C に移動することができる．そして，(3)のように最も大きい円盤を C に移動し，それから B にある残りの円盤を C に移動すればよい．この方略を ET ルールで記述すると以下のようになる．

H1: (hanoi *N *A *B *C), {(> *N 0)}
→ (:= *N1 (- *N 1))), (hanoi *N1 *A *C *B),
(format “/s --> /s” (*A *C)),
(hanoi *N1 *B *A *C).
H2: (hanoi 0 *P1 *P2 *P3) → .

H1 には，「N 枚の円盤を A から B を経由して C に移動するという問題は，N-1 枚の円盤を A から C を経由して B に移動し，N 枚目の円盤を A から C に移動し，N-1 枚目までの円盤を B から A を経由して C に

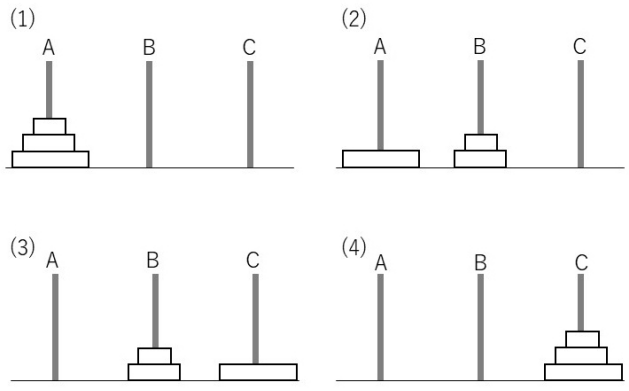

図３　ハノイの塔の解法パターン

移動するという問題に置き換えることができる」という意味を与える．*H2*は「円盤が 0 枚のときは何もしない」という意味を持つ．このように，ハノイの塔は２つの ET ルールだけで解くことができ，これも意味でプログラムを捉えることで，再帰計算を追わせないように配慮する．

4. 考察

SCRATCH のようなビジュアル型プログラミングは，基本は手続き型言語なので，順次処理，条件分岐，繰り返しといった動きを組み合わせて，１つのまとまりを作って問題を解く．まとまりが大きくなる場合にはサブルーチンを使って部品化し，部分問題の答を用いて大きな問題を解くというアプローチとなる．これに対し ET プログラミングは，問題の解くための方略を考えてルール化し，そのルールに意味を与えて，問題の置き換えの正しさを検証するというアプローチをとる．したがって，方略を立てる，ルールを書く，意味を与えるという，それぞれの段階でプログラミング的思考を育成するので，反復的に学習できる機会を提供している．そして，ルールが他の計算に影響せず完全に独立していることと，ルールの適用による変換結果を返すので，途中の段階で実行してもエラーにはならないことが，プログラミングのしやすさをもたらしている．さらに，ハノイの塔で示した通り，プログラムのサイズが小さく，作成に時間をかけなくても済むことから，１回の授業で完結できる見込みがある．

その一方で，ET プログラミングの欠点としては，図形描画などのグラフィック面が弱く，絵や動きで児童の興味・関心を引くのが難しいことが挙げられる．図形描画に関する B 述語はサポートされているが，スプライトを動かすといった直感的な操作ではないので，手引にある正多角形の作図などは，ビジュアル型プログラミングが向いている．また，D 述語は全角日本語を使用できるが，括弧やカンマなどは半角小文字で記述するので，全角のままで括弧を付けるとシンタックスエラーが起こる．それを避けるために D 述語を半角小文字にし，ルールを全て半角小文字で統一しているが，D 述語は英単語で名付けることが多く，児童にとっては英単語の意味の理解がネックになる．

5. おわりに

本研究では，筆者らが過去に実践した ET プログラミングを再考し，プログラミング教育の手引で示された例題を ET ルールで記述することによって，本プログラミングが現在求められている教育方法として適用可能かどうかを検討した．

その結果，手引で示された例題は，方略を立てて ET ルールを導き，そのルールに意味を与えるという方法で対応できることを示し，プログラミング的思考を反復的に育成する機会をもたらした．そして，ルールの独立性やサイズの小ささがプログラム作成を容易にし，より大きなプログラムに取り組めることも，自動販売機の例から提示した．

今後は小学校教員と協力して，学習指導案の作成や授業の機会を設けるなど，ET プログラミングの実践可能性や教育上の課題について調査していきたい．

参 考 文 献

(1) 文部科学省: “小学校プログラミング教育の手引（第二版）”, http://www.mext.go.jp/a_menu/shotou/zyouhou/detail/1403162.htm, (2018) (2019 年 1 月 12 日確認)

(2) 石川貴彦, 赤間清, 小池英勝, 三高康嗣: ”等価変換型プログラミング言語 ET の導入による学習の構想”, 日本教育工学会論文誌, Vol.27, Suppl, pp.33-36, (2003)

(3) 石川貴彦, 赤間清, 三高康嗣: ”プログラミング教育のための学習支援システムの開発”, 平成 15 年度情報処理教育研究集会講演論文集, pp.97-100, (2003)

(4) 太田剛: “自動販売機をプログラミングするーフローチャートやいろいろな設計ー”, http://beyondbb.jp/Materials/StudentT03_VendingMachine_170215.pdf#search=%27%E8%87%AA%E5%8B%95%E8%B2%A9%E5%A3%B2%E6%A9%9F+SCRATCH%27, (2017) (2019 年 1 月 24 日確認)

(5) 山田耕太郎, 有吉優菜: “数理パズルを使ったアルゴリズム教育の実践と評価”, 比治山大学紀要, No.24, pp.67-73, (2017)