MINDSTORMS 第4章 コンピュータの言語と人間の言語

久野　靖*

2010.8.25

1　イントロ

□ むかでのエピソード

□ 合理主義の今日一般に言われていること (ホント?)

- 思考が行動を妨げる
- 思考が学習を妨げる (!!)
- 自転車の乗り方:　「練習していればできるようになるよ」

□ 大勢の学者 (=行動主義) の見解

- ある種の知識は、言葉で表現したり、意識的な思考で把握できない
- ブルーナーの主張:　行動以外に表せない種類の知識

□ パパートの見解:　二分を拒否

- 言葉で全部表せることもないし、全く表せないこともない
- 知識の発展の歴史:　言葉や図解の効力を増すような技術の発展
- 自転車の例:「乗り方を覚えるのに効果がある程度でいいから知識を交流」

□ 4章の中心課題:　学習について話すための叙述的な言語の発展

- 「やってみる」のが最善とされる事柄 (身体技術等) に注目
- 体育を知的でない科目として扱うのと正反対

□ 科学者は形式的かつ叙述的な言語を使用

- 子供も身体的技術 (例:お手玉) の習得に同様の言語を使用可→科学者と共感する立場に身を置く
- 普段身体を使って何かしていることも「科学していること」と思うように
- デカルトの直交座標が日常に関連→デカルトに意味 + 自分自身に意義を見出し得る

*筑波大学大学院経営システム科学専攻

2　身体的技術の習得

□ 世間や教育心理学では→身体的技術習得に意識は役立たない

- スポーツを職業とする人 (例:コーチ) はそうではない

□ 例:　ティモシー・ガルウェイ「Innter Tennis」

- 自分を「分析的な自己」と「全体論的・直観的な自己」から成るとする
- いずれかが自分を随時的に支配
- テニスの習得には、いつ制御を取りいつ譲かをそれぞれの「自己」に教えることが大切

□ 「分析的」「全体的」の選択を学習者に委ねることは特徴的

- 学校のカリキュラム改革では「選択を上から与える」
- ガルウェイがすべて正しいとは言わないが「習得について組織的に考える」必要性については同意

3　コンピュータ言語

□ コンピュータ言語:　「機械を支配する手段」かつ「思考のための叙述的かつ新しい言語」

- コンピュータを知らない人でも「インプット」「アウトプット」「フィードバック」などの概念を」使う
- プログラム上の概念が身体的技術にどう適用できるか → プログラム行為を叙述的な考案の源泉とする

□ 多くの科学的/数学的進歩 → 組織的に考えるにはあまりに無定形と思われた事柄を表現する言葉や概念を与えた

- 例:　デカルトが蝿の飛ぶのを眺めていて解析幾何を考案 → 代数の言葉を空間と結びつける → 表象能力の前進
- 円の描き方を考案した子供も同様
- これまで全体的・視覚/筋肉運動知覚によってしか知られなかった事柄を分析的に表現する能力を得る

□ お手玉やむかでには微分方程式はやりすぎ → コンピュータ言語の形式が向いている

- コンピュータ → やるべきことを十分な正確さで叙述することが必要→ 強力な叙述形式の考案

□ コンピュータサイエンスという名称は不的確： 大半は「叙述ということ」「叙述のための言語」の科学

4 例: 構造的プログラム

□ キース (小学 5 年)： 図 10a のような線画をコンピュータに描かせたい

- まず行動を 1 歩ずつ記述 (図 10b) →バグがあるコード (図 10c)
- バグは悪ではなく「間違いを直す」のは普通のこと
- しかし何が悪かったのか想像がつかない (プログラムの書き方に起因)

□ 構造的プログラムのやり方では： 2 つの V と正方形を作って全体手順から呼ぶ

```
TO MAN
VEE
FORWARD 50
VEE
FORWARD 25
HEAD
END
```

- さらに VEE、LINE、HEAD を定義して全体を完成

□ ロバート (中 1) の言葉

- 「僕のプログラムは皆、頭に入る大きさになった」
- 「前は混乱したが、今はいっぺんに噛みこなせる以上は口に入れない」
- 理解できないほどの段階なしに大きな知的構造を作れる

□ キースの場合

- 以前も部分手順に触れたことはあったが拒否 (直線的構造になじむ)
- MAN をデバッグする時はじめて必要に

□ 大人「どうしたらいいか分かってるでしょう?」子供「部分手順に書き直さないといけないんじゃないか?」

- 正しいやり方を押しつけるのでなく、自分の探求で自分のものに (コンピュータの柔軟性と力による)

□ 直線的 vs 階層構造： 身体技術にも現れる

- 2 人の小 5 生マイケル (外向的)、ポール (内向的)
- ポール： 早くから構造的プログラム、マイケル: 直線的にこだわる
- 2 人が竹馬乗りにチャレンジ
- マイケルは直線的に手順を思い浮かべ繰り返し練習して習得
- ポールの方が早く習得 (2 人とも驚く)
- ポールは「バグ」を発見して修正する方針
- 「足を前進させるのでなく、竹を前進させようとする。足は竹が運んでくれる」→プログラミングを身体的技術習得に応用
- 少なくとも「特色を表明することを可能に」
- マイケルではより明らか： 2 人のやり方の違いに気付く

5 お手玉

□ お手玉： 数学を使って身体的技術を学ぶよい例

- 構造的プログラムという考えが身体的事項に織り込まれることで力

□ 2 つの方法： シャワー式とカスケード式

- カスケード式の方が投げ方が 1 種類なので習得しやすい

□ 問： 習得したい人が言葉で分析的に描写しようとすると助けになるか妨げになるか?

□ 言葉による描写では混乱を招くようなものもある

1. ボール 1 と 2 を左手、ボール 3 を右手にして始める。
2. ボール 1 を右手へ高く放物線状に投げる
3. ボール 1 が頂上にある時に、ボール 3 を左手へ、同様の高い放物線状に投げるが、ボール 3 はボール 1 の軌道の下側に投げるように気をつける。
4. ボール 1 が右手に到着し、ボール 3 が頂点にある時に、ボール 1 を、捕えボール 2 をボール 3 の軌道の下側に投げる…

- これは強引な直線状のプログラム
- 構造的プログラムの方がいいことを示す

□ 目的： TO JUGGLE という動作の手順を作ること

- 部分手順として TOSSRIGHT、TOSSLEFT を使う
- この手順はタイミングも考慮する必要 (TOSSxxx は捕らえることまで含む)

□ 並行処理の表現 → WHEN デーモン

- 例： 「WHEN HUNGRY EAT」(条件が満たされたらそのつど x を実行)

□ ここでは 2 つの WHEN デーモン

```
WHEN なんとか TOSSLEFT
WHEN なんとか TOSSRIGHT
```

- 頂上で動いている向きが問題 (図 13x) →それをとりいれる

```
TO KEEP JUGGLING
  WHEN TOPRIGHT TOSSRIGHT
  WHEN TOPLEFT TOSSLEFT
```

- ないし、もっと簡単に

```
TO KEEP JUGGLING
  WHEN TOPx TOSSx
```

□ 教えるために応用するには? → いくつかの仮定を認識

- 1. 学習者は左投げと右投げができる
- 2. 頂上で左向き、頂上で右向きというきっかけになる状態が分かる
- 3. TO KEEP JUGGLING という手順に従ってこれらを組み合わせられる

□ 段階 1: 学習者の投げ方を調べる (1 つのボールを片手から他方に投げさせる)

- 多少手を加える必要があることがしばしばある

□ 段階 2: 組み合わせられるかを調べる

```
TO CROSS
  TOSSLEFT
  WHEN TOPRIGHT TOSSRIGHT
  END
```

- 左右のボールの交換。通常これでもすぐにはできない

□ 段階 3: 手順にバグがあるか調べる

- バグの例: 投げたボールを目で追う → 2 番目の投げができない

□ 段階 4: デバッグ

- 例: 目で追うことが問題なら、ボール 1 つ目で追わずに投げる練習
- 放物線の頂上付近に目を止めたまま投げる (しばらくの練習でできる)
- TO CROSS に戻る

□ 段階 5: 3 つのボールに進む

- 2 つのボールを片手、もう 1 つを別の手に持って始める
- TO CROSS から TO KEEP JUGGLING への移行は難しくない

□ お手玉の分析で博論を書いた人も

- プログラミングの概念がデバッグを容易にし、教育を効果的に

□ 複雑な過程を叙述する→基本単位に分割→その中のバグ発見は容易

- 複数のバグがあると発見が難しい→分割すれば 1 つのバグに
- 例: 初心者がいきなり 3 つの玉でやろうとした場合→複数のバグ

□ 反復練習しているうちに「たまたま」効果的な動作が発現

- 例: たまたま目の動きが少ない状況→これを習得
- これでもできるが、時間が掛かる

□ 学習過程を意識し制御することでずっと効果的に学習できる

- 瞬時にできるわけではない。何時間も掛かるのが 20〜30 分に

6 デバッグへの抵抗

□ ロゴに始めて来た子供→うまく行かないと消してしまう

- 最初から書き直して「1 発で」やりとげようとする
- 「小さい間違いはあるけれどよいプログラム」 vs 「失敗」
- 学校では「間違いは悪」→間違いを検討したがらない
- プログラミング体験→間違いを検討し直すことの効用を信じるように

□ ロゴ環境では子供は自分の間違いに寛容になっていく

- ロゴの教師の態度にもよる: 教師も学習者であり、誰でも間違いから学ぶ

□ 例: クリスマス企画にメリークリスマスと描くプログラム

- 時間がなくなったので教師が家でプログラムを書いてくる
- 先生のプログラムが正しくなかった時、子供は「先生にもプログラムをどう直していいか分からない」ことを発見

□ 通常の教室: 教材そのものに研究課題を生み出す要素が欠けている

- プログラミングでは: 教師と生徒が知的な共同研究

- 教師も出会ったことがない状況→知らなかったふりをする必要もない
- 「先生のいうとおり」でなく「先生のするとおり」で学習
- 教師のすべきこと→完全に理解できるまで問題を追求

□ ロゴ環境は小学生が (日常生活では無い完璧さで) 理解できる問題を提供

- 例: 屋根が下に描かれてしまうプログラム
- 「タートルになってやってみよう」でなぜそうなったかは完璧に理解
- 大人も REPEAT [FORWARD 100 RIGHT 60] で三角形とかよくある
- タートルの行動を完全に理解→子供にとって他に類のない経験

7 タートルの価値

□ 反論: 子供はタートルについて完全に理解していないし、その背後の機構についても知らないではないか?

- だから子供を煙に巻いているだけではないか?

□ お手玉でも同様。手順による記述で、本質的な要素が把握できたか、または複雑性を覆い隠すことで神秘性を持たせたか?

- 例: ニュートンは物体を点に抽象化することで宇宙を理解 →本質の把握か、複雑性の隠蔽か?

□ 科学者のような考え方とは→これらの認識論的問題に直観的理解を持つこと

- タートルで学ぶことは子供にこれらの問題に接する機会を与える

□ タートルの世界が閉じた世界だということは子供にもすぐ分かる

- これを用いて独立したさまざまなマイクロワールドを提供することも (次章)
- 子供はマイクロワールドの属性を無関係な質問に煩わされずに探求
- この探求の習慣を科学的な理論の習得に移しかえ可能

□ コンピュータ世界に内在する単純さ→複雑な企画を実現可能

- 例: 組み立て式のおもちゃ、複雑な構造のあそび→実行の段になると物や人間に関わる制限にぶつかる
- コンピュータのプログラムの場合はそのような制約なしに安全に複雑性に対する感覚を身につけさせられる

□ 現代の科学や工学→複雑な企画を実現する機会を生み出すいっぽう、簡素さの持つ力も教えてくれている

- 挿話: デボラ (小 6) は学校の勉強に問題があり、タートルにも恐る恐る
- どんな小さな試みにも教師の同意を求める
- ある時:方向転換に「RIGHT 30」の命令しか使わないと決めた →マイクロワールドの中に自分の決めたマイクロワールド
- その制約下で何ができるか探求することに興味
- 教師が簡単な方法を教えても自分の意思で制約を維持
- やがて行動全般に大胆さが現れる変化

□ コペルニクス、ガリレオ … 物理学とは関係ないような迷信的依存から人々が逃れることを可能に

- デボラの経験も「数学的な理論の成功が道具以上の役割」
- つまり「理念の力、精神の力を肯定」