June 25, 2002

報告者：中池竜一

Cognitive Turors: Lessons Learned

John R. Anderson, Albert T. Corbett, Kenneth R. Koedinger, and Ray Pelletier
Carnegie Mellon University

- tutor開発の10年間の歴史をレビュー
- 学生がLispや幾何学，代数学の問題をどのように解くかというプロダクションシステムモデルをACTで開発．
  → turorはそれらの認知モデルを通して開発された．
- ACT理論を元にした8つの原則によってtutorを構成．
- それらtutorの初期の評価は，いつもではないが，かなりの成果を示した．
  → 最も良い場合，生徒たちは決められた教示時間の1/3で熟達レベルに達した．
- tutorの利用から得た経験的な知見：
  → 学生はproduction-rule群のスキルを学習している．
- 最も良い教示作用(tutorial interaction)スタイル：
  → tutorは，短く直接的なエラーメッセージからなるフィードバックを即座に返答．
- tutorは人間の家庭教師というよりもむしろツールとして生徒に与えた方が良い働きをするようだ．
- 学生はtutorの表示を他の環境でも適用できる．
  → tutor環境をテスト環境へとマッピングできる．
- tutorの発展的な利用：
  → 問題解決インタフェースの選択や，領域専門家のガイダンス下のカリキュラム構成．
  → 各種環境下の問題解決認知モデルのデザインとそのproduction-ruleをもとにした教示の構築，
- そして実際の授業におけるtutorの配置．
  → (数学教師評議会)NCTM standardを達成するためのシステムに組み込まれている．

- the Advanced Computer Tutoring Project at CMU：過去10年以上の研究
  ・コンピュータをベースとした教育技術の開発→ cognitive tutors.
  ・生徒に学習するよう要求
  ・生徒に問題を与える．
  ・システムはこの問題を解決する能力を持ち，生徒がどのように解くかを予期できる．
  ・tutor → 生徒に働きかける人間の家庭教師のemulation

- 3段階の視点
  (a)1980年代半ばのtutor構築の一時的混乱
  (b)1980年代後半の評価の一時的混乱
  (c)実践的tutorシステム開発のための現在の努力

STAGE1:EARLY TUTOR BUILDING

- 1982年の，学習と問題解決に関するACT* 理論の完成:(J.R.Anderson, 1983)
  認知スキルの獲得に関する応用領域：
  幾何学の証明生成(J.R.Anderson, Greeno, Kline, \& Neves, 1981)
  LISPの初期のプログラミングスキル(J.R.Anderson, Farrell, \& Sauers, 1984)
- 認知スキルは非常に多くのgoal-related知識で構成
- 認知スキル獲得は，動作(actions)と結果(consequences)のための，課題目標と課題状態に関係する数多くのルールの定式化
- 理論では，このgoal-oriented知識はproduction-ruleの形で表現される．

幾何学の証明生成に関するルール：

```
IF          the goal is to prove two triangles are congruent
THEN        set as a subgoal to prove that corresponding parts are congruent.
```

LISPプログラミングに関するルール：

```
IF          the goal is to get the second element of the list
THEN        code "car" and set as a subgoal to pass to car an argument that
              is the tail of the list.
```

- 知識獲得の理論は，これらの教示を含んでいる．

The ACT Theory

- tutoring は，知識獲得のACT理論(ACT*理論 1983，ACT-R理論 1993)に深く関連．

-手続き的ー宣言的 弁別(Procedural-declarative distinction.)

- 宣言的知識(side-angle-side定理)と手続き的知識(証明で定理を使う技能)を区別
- 理論が仮定：

  goal-independentな宣言的知識は最初からシステムの中に入っている．
  （あるいは観察あるいは教示から直接エンコードされることができる）
  認知スキルは，宣言的知識を手続き的知識(production-rule)に変換することに依存している．

-知識の編集(Knowledge Compilation.)

- 生徒は様々な解釈的(interpretative)手続き，後づけの説明やアナロジーを用いて問題を解く．
- 学習プロセス(knowledge compilationと呼ぶ) ：解釈的手続きをproduction-ruleに変換
- 理論が仮定：

  問題解決活動の文脈の中で，宣言的知識が使われることでproduction-ruleが学習される．

-増強(Strengthening)

- 手続き的知識，宣言的知識の獲得は練習によって増強される．
- 弱い知識(weak knowledge)の適用は，つまずきやエラーを引き起こす．
- だから知識のエンコードが成功した後でも，さらなる練習により，スムーズでより素早くエラーが少なく遂行できる．

The Nature of a Cognitive Skill

- tutorが教えるものとして認知スキル(cognitive skill)という語を使っている．
- 認知スキルは，その領域の中で獲得したproduction-rule集合を参照するために使われる．
- 原則として，全ての領域知識はproduction-ruleあるいは宣言的知識で表現することが可能で，
  domain-independentな手続きによって解釈される．
  → しかし，それを行なおうとは考えていない．

- 幾何学における宣言的知識の例：

  side-angle-side 定理「二つの三角形の両辺とその間の角が等しいならば，その二つの三角形は合同である」
- 手続き的知識の例：

  何が間の角であるかを決める，サブゴールの設定，推論する．合同であると認識する

- 宣言的知識は，tutorの簡単な説明によって獲得される
- 手続き的知識(production-rule)は簡単な説明では獲得されず，実行によって獲得される．
- それゆえ，tutorは，

  それらスキルが監視され，表出されるような状況をセットアップし，
  それら獲得が形作られるための適切なフィードバックを与える．

Initial Work on Tutoring

- 初期のtutoringシステム開発の主な動機は，実践的授業よりも，ACT理論のテスト
  生徒にproduction-ruleモデルをもとにした活動をさせることで，学習が成功するかどうか．
- 1983年に LISP tutor と 幾何学 tutorの研究が始まった．
  LISP tutor：生徒が，短いプログラムを作るのを助ける．(figure.1)
  幾何学 tutor：幾何学的な証明を検索することを助け，proof-graphを作り提示する．(figure.2)

- これらは,コンピュータベースの教示はどのように実現すべきかという,幾つかのアイデアから実現された．

-Model.

生徒の一連のタスク遂行を予期するproduction-modelをtutorに組み込むべき．

-On-path actions.

モデルにより生成される正しい解決経路上に生徒がいるのならば，tutorは助言を行なわない．

-Off-path actions.

生徒が経路から外れたならば，経路に戻るように教示がフォーカスされる．

-Error feedback and help.

教示のタイプは2種類
・生徒の認知的な誤り("bug") → なぜそれが誤りなのか，メッセージで説明("bug message")
・生徒が助けを要求("help") → helpメッセージが正しい解をガイドする("help message")

- Model-tracing approach：
  コンピュータモデルが生成する経路と，生徒の解決の振舞いを関連づけようとする．
  plan-recognition problemの一種（計算的に生徒の振舞からそのプランを認識することは難しい）．
  → しかし，解釈経路上での生徒の行動を強調することにより対処．

Interacting With the LISP Tutor

- 24x80の文字ベースのターミナル
- 上部window：問題を記述する "tutor window"
- 下部window：生徒がprogramをタイプする "code window"
- figure.1：
  "tutor window" 次のような値を返す関数を定義しなさい．
  "code window" 最初に，テンプレートを表示．生徒は，それを拡張しながらプログラミングする
- TABLE.1：仮想的な生徒の学習（エラーを実際よりも多く行う）
  code windowは省略して，tutor windowの表示のみ
  9つのkeyとなるサイクル
  最終的に"LISP window"というLisp環境を呼出し，その中で生徒は実施する．

The Initial Incursion Into the Classroom

- 1984年に2,3の高校で幾何学tutor，CMUのコンピュータサイエンスのミニコースでLISP tutorを利用．
  ・その結果は，初期の予想を超えて，良かった．
- LISP tutorの評価：
  LISP tutor利用グループと，通常のLISP環境利用グループで，同じ練習を行なわせて比較．
  前者の方が，後者の30%短い時間で終えて，最後のテストも標準偏差が高かった．
  それを受けて，LISPについて半期のコースを作り，人文科学や社会科学の学生に教えた．
  現在でも成功しているコースの一つ．
- 幾何学 tutorも実際の授業で使用．Xerox D-machine，Peabody High School(1985-1988)
- 代数学I tutor．symbol-manipulations スキル(一次方程式の解決等)を教えるtutor．(figure.3)

The Eight Principles

- tutorを用いた練習とACT*理論間の堅い結び付きについての記述．
- 私たちが感じた，理論から裏付けられたtutorデザインに関する8つの原則

-Principle 1: 生徒の能力をproduction集合によって描写せよ

・tutor開発は， ターゲットとするスキルの正確なモデルが与えられるべき．
・認知モデルは，適切な履修目標を与えてくれ，生徒の行動を解釈してくれる．
・この原則は，コンピュータインタフェースに限ったものではなく，生徒とのやりとりや，カリキュラムによる学習促進にも適用できる．

-Principle 2: 問題解決に潜む目標構造を伝達せよ

・問題解決は，問題(goalの集合，sub-goal)の分解を含んでいると，ACT理論は仮定．
・生徒がスキルをマスターするのが難しいのは，問題解決を決める目標構造が十分伝達されていないから．
・暗黙的なgoal構造を，明確に表示(e.g.，proof graph)することで著しい成功．

-Principle 3: 問題解決の文脈の中で教示を与えよ

・状況に特殊な学習に関する研究(J.R.Anderson，1990)からの原則．
・現在のsituated-learningの動き(Collins，Brown，\&Newman，1989;Lave \& Wenger，1990)もこの原則を裏付ける．
・必要なポイントで，正確な教示を行なうことで，生徒は問題解決の障害となるものを発見できることが実験的に確認された．

-Principle 4: 問題解決知識の抽象的な理解を促進させよ

production-rule(手続き的知識)はあまり一般的なものではない．
練習では，help messageやerror message のように言語的に，正確な抽象概念を強化させた．

-Principle 5: working-memory の load を最小限にせよ

ACTでのproduction-ruleの学習：
・全ての関連情報(学習するruleに関連する条件やaction)が記憶内で活性化する必要がある．
・Sweller(1988)は，working-memory loadが高いと学習を阻害することを示す．
問題解決中は教示も最小限にすべき(既にマスターしたスキルに関連する教示は提示しない)
カリキュラム中でも，一度に少しずつ教えることは重要．

-Principle 6: エラーに関するフィードバックを即時に与えよ

これら原則の中で最も議論されるもの．
ACT*理論では，新しいルールは，問題解決過程の記録をトレースすることによって作られる．
→ 長いルールほど，エラーが正しいものになるまで待つ必要があり，統合するまで時間がかかる．
現在のACT-R理論では，問題解決の生成物(コード，証明等)からルールが学習される．
→ 一度に問題解決の重要な段階が一度に活性化するかどうかは問題ではない．

-Principle 7: 学習に関する教示はサイズは微量に調整せよ

ACT*のlearning operatorより：
→ 1つのproduction-ruleは多くのproduction-ruleから合成される．
ACT-R：
→ ACT*とは異なり，その他のメカニズム(J.R.Anderson，1993)により，この原則が推測できる．

-Principle 8: ターゲットとするスキルへ連続的に近づけよ

生徒がスキルを実行しようとする時，最初は全てのステップを実行できないことが多い．
tutorは，生徒の途切れたステップを埋めることができる．

STAGE2:THE EVALUATIONS AND EMPIRICAL STUDIES

- 最近のtutorに関する研究の紹介．
  幾何学，代数学，LISP tutorの評価

スキル獲得や，生徒のモデリング，フィードバックのコントロールや内容の特徴について

The Geometry Tutor

- pilot study:多数の高校で多くの成果（1985-1986）
- tutor有り／無しで，同じ教師によるクラス間比較(1986-1987)

→ 証明スキルに関する final test(紙と鉛筆）の生徒のパフォーマンスを予測しようと試みた．
→ 下の式はscaleを0-80でとった生徒のパフォーマンスの予測．

```
35+7.5*(letter grade in algebra)
  +14 if access to tutor one on one
  +7  if access to tutor two on one
```

→IQよりも，algebra classの学年の方が予測できた．2人で1台のコンピュータを使う場合．
→第3者(Schofield \& Evan-Rhodes，1989; Wertheimer，1990)の評価：学習のモチベーションが上がった．

The Algebra Tutor

- 学校での評価(1987-1988)：実験class間に差は無し．

→ tutorのインタフェースと授業のインタフェース(紙と鉛筆)の間にさほど差がなかったためだろう．
→ 一次方程式の操作はかなり簡単で，tutorの教示無しにスキルを獲得した学生もいた．

The LISP Tutor

- 授業で，tutor利用と，標準LISP環境利用を比較した実験．

→ 練習時間が30%短く，post-testでスコアが43%良かった

- 実験室での評価(Corbett \& Anderson，1991)：各生徒のペースで，同じ問題とテキスト使う．

→ 練習は64%早く終わり，post-testでスコアが30%高かった．

- 良いデザインが行なわれたtutorは，標準的な環境の3分の1の時間で，より高い達成へと至ることを意味している．
- LISP tutorは，LISP，Prolog，そしてPascal tutorへと一般的なプログラミング環境に発展

Componential Analysis of Learning

- 認知科学及び教育学における現在の論点(e.g.，Shephard，1992)：

→ 学習の構成要素を書き下すことが可能かどうか．
→ tutorという文脈の中では，その答はyes．

- LISP tutorからのデータを細かく分析し，本質的な4つの要因にまとめることができた(J.R.Anderson，et al，1989)．

-Production practice

生徒は，練習中，production ruleを頻繁に使うので，rule適用に関してパフォーマンス上昇．
rulesと表面的なcode symbols(car，+，write)間のマッピングにより
→ それがruleであることや，表面的ではない重要なユニットであることを示すことができる．
ACT理論では，これはproduction-ruleの強化が原因であると考えられる．
figure.4：新規ruleを与えた場合と，既知の古いruleを与えた場合のprogram coding時間の減少曲線．
→ 古いruleの場合が減少が大きい．以前の練習の効果．

-Within-problem practice effects.

rule-specificな効果を取り除いたときの，特定の問題における生徒の進歩：
→ ルール適用の時間や正確さの改善
ACT理論では，繰返しアクセスすることによる，問題の宣言的表象の強化が原因．

-acquisition factor.

生徒のパフォーマンスの要因分析より，

→ 練習中に新しく手に入れたruleをどのくらいうまく遂行できるか．
この要因の本当の意味ははっきりとしていない．
→ 個人差，あるいはそのデータをどのくらい注意深く参照したか

-Retention factor.

初期の練習から得たruleをうまく保持できるか．
獲得の要因とは直行(orthogonal)している．
個人差，あるいはそのデータをどのくらい注意深く参照したか

Knowledge Tracing

- LISP tutorは，knowledge tracingと呼ばれる生徒のモデリング能力を持ち，学習させるのに用いる．
- 生徒は，次のセクションに進む前に，そのセクションで紹介された個々のルールを練習する．
  → これは，生徒の達成レベルに大きな影響を与える(J.R.Anderson et al., 1989)
- tutorの konwledge traceingの基礎となる学習とパフォーマンスのモデル
  → posttestのパフォーマンスを予期することができる(Corbett \& Anderson, 1992)
  → posttestを正確に解くことができる確率は，個々の必要なルールを学習した生徒の確率から求められる．

Locus of Feedback Control

- 原則に従い，tutorは即座にフィードバックを返す．
- フィードバックについて検討するために，3つの異なるバージョンを追加．
  (1)**no-feedback tutor**：即座ではないフィードバック．目標達成方法を助言しない．
  → 生徒はtutorにそれが正しいかどうかは聞くことができる（生徒は主に独力で行なう）．
  (2)**error-flagging tutor**：エラーの場所を太字で示す即時フィードバック．修正は行なわない．
  (3)**demand tutor**：即座のフィードバック．生徒が聞いてくるまで助言をしない．
  → 聞いてきたら，最初の誤りに対して通常の即座フィードバックを行なう．
- 4種のtutorをそれぞれ用いた4群が，5回のレッスン．
- no-feedbackを除く3つのtutorは，posttest(紙と鉛筆，標準LISP環境)からは重要な違いは見られなかった．
- no-feedbackに比べて，3つの tutorは共に正解率が高かった．(43\% < 55(58)\%)
- 練習時間については，figure.5.
- 直接，生徒に質問した場合は，通常の即座フィードバックを高く評価した(信用できない)．

Feedback Content

- 問題解決をしている生徒に対して何を助言すれば良いか．
- 2つの観点
  (1) 生徒がエラーを起こした時に，そのエラーについて助言する．
  (2) 生徒が手助けを求めたり，助けを必要としているように見えたら，助言を与える．
- LISP tutor：(J.R.Anderson et al., 1989)
  → 学習速度以外に，統計的な差異は見出されなかった．生徒の評価は(1)が高い．
- 幾何学 tutor：(McKendree, 1990)
  → 1回目は差異無し： 理解することなく，tutorを通じて修正を行う生徒がいた．
  → 2回目(message調整後)は，練習途中及び最終の達成度に差がでた．
- LISPは比較的早く達成するが，幾何学は時間と達成度のトレードオフを考える必要がある．
- の方が知的に見えるので，生徒の態度も異なるだろう．

STAGE3:PRACTICAL DEPLOYMENT

- 研究の成果にも関わらず，あまり一般では使われなかった．主にCMUにおける利用．
- scale-downした幾何学tutorがMac SE上で実装され，国内でたまに授業に用いられるように．
- 人工知能は巨大で，それが動く機械は多くない．
- 現在ではそのような機械も多くなりアメリカの授業で使われても良いはず．

- 問題点を明らかに．

１．教育者が教えたいことについて言及しようとしていなかった．
数学教師がより一般的な推論や問題解決を主張しているとき，幾何学tutorでは証明スキル指導に焦点を当てていた．
２．tutorを利用して試験に合格した後，生徒に何が起こったのかを考えていなかった．
成功の達成を見るためにfinalテストを使っていたが，カリキュラムのより大きな目的に適していなければならない．
３．tutorは，自由性があまりなく，教師のニーズや信念に答えるような調整が簡単にできない．
４．授業においてtutorを広めるための方法をほとんど理解していなかった．
プロジェクトに長時間従事する教師がいなかったことに関係するかもしれない．

- より広範囲の授業に対応できるtutor等を現在開発中．

Interface Construction and the Issue of Transfer

- 最初は，認知スキルのprodction-ruleモデルに注視して，インタフェースデザインについてはあまり顧みられていなかった．
- 現在の我々の観点は，ある特定のインタフェースにおける問題解決の教授である．
- 重要な問題は，転移である．
  コンピュータ上のtutorから紙と鉛筆の授業場面への転移
  tutorが教えようとしているスキルからターゲットスキルへの転移を可能にするインターフェースをデザインする必要

- 転移を3つのレベルに区別

-Identical productions.

・tutorが練習させるproduction-ruleとターゲット領域のruleが同じ場合，総合的な転移が予期．
　e.g.，ターゲット領域がコンピュータシステムで，tutorがそれについて教える時
・tutorのruleとターゲット領域のruleが重なる時，重なっていない部分は転移できない場合もある
　e.g.，LISP toutorにより素晴らしいcodeを書けるようになっても，syntaxに関する知識は言葉で補う必要がある

-Translating actions.

・tutorのruleとターゲット領域のruleが異なっていても，学んだruleのactinをターゲット領域で実行できるかも知れない．
　→ LISP tutorにおけるmenuからのsymbol入力を他の環境でも実行．
　→ 幾何学 tutorで用いたproof-graphを他の証明でも利用．

-Declarative transfer.

・ある領域のactionが他領域で直接利用できなくても，潜在的な能力の転移が起きる場合がある．
　→ LISP tutorによりプログラミング練習により，LISPの式の評価も同様に上達する．

- tutor開発において行なわれたインタフェース構築の2つのアプローチ
  自分自身で構築する（e.g.，production-modelのためのインタフェース）．
  既に存在するソフトウェアの一部分を付け加えていく．

Curriculum Construction

- 教育のコミュニティからの詳述．
  教育者側が生徒に解いて欲しい課題は何か？
  ・NCTM standards に達して欲しい．　→ tutorが広く応用できる．
  ・問題解決インタフェースとして，コンピュータ上での入力を重視．
  　→ 幾何学スケッチ帳(Geometer's Sketchpad)の利用
- とはいえ，tutorのインタフェースにはそれほど自由度はない．
- 特定のインタフェースの中には，教師が生徒に解いて欲しい問題の概念や構造(e.g.,代数記号処理)を含んでいる．

→ しかし，それを生徒に明示的に教師が教える必要があるかもしれない．
- tutorは，知識のトレースや学習の達成を支援するが，教師は， それ以上の知識を持っている必要がある．

Production System Modeling

- 認知モデル構築者としての我々の主たるタスクは，生徒のモデルを構築すること．
- たいてい，一つ以上の解決パスがあり，理想的な生徒のモデルは，その全てを生成することができる．
- productioni-ruleはworking-memory内の情報を返答する．
  現在の状態は何か．目的は何か．
- 問題解決のためのproduction-ruleが明確化したら，生徒の行動とのマッチさせる．

→ 文章の理解を行なうような，自然言語のモデルについては考えていない．
- そして，その結果から生徒の現在状態を推測し，誤り訂正などを行なう．

Declarative Instruction

- 外側から与える宣言的な知識から領域における能力の一部は獲得できる．
- tutorが与える様々な種類の教示の原則．最小限で，必要以上の事は言わない．

→ 教科書からは支持されない

→ 研究からは支持．(Reder \& Anderson, 1980; Reder, Charney, \& Morgan, 1986)
- tutorの2種類の宣言的な教示
  ・Error message(errorの即時訂正)，
  ・Help message(尋ねられた場合に教示)
- Help messageに関する問題（1つのメッセージを与えるか，解決に至るより明示的な一連のメッセージを与えるか）

→ 現在は，生徒ができるだけ自分で解けるよう指導する
- もし，一人での学習ならば，明示的な一連のメッセージは必要だが，授業の中では，教師の助けを得ることができる．

Deployment in the Classroom

- tutorの実際の広まり方がどのように行なわれたのかを検討するのは興味深い．
- CMUでは，tutorは，自分のペースや一人での学習する環境として使われた．
- 教師無しでもうまく行なわれた理由：
  (a) 私たちが教えたいことを伝えるためのtuterをデザインした．
  (b) 授業を越えて総合的にコントロールした
  (c) 比較的熟達した生徒も来ており，概ね，computerに慣れていた．
- アメリカの高校数学の授業における利用．

合衆国中で様々な種類のカリキュラムがあり，最初は，その中のごく一部の教師の間で使われた．
十分に熟達していない生徒に教師無しでtutorを使わせた．中には行き詰まる生徒もいた．
授業においてtutorは成功するようになり，現在，Pittsburgh高校で代数学，幾何学，そしてPASCALプログラミングに関する30の授業でtutorが用いられている．
- 実験室で一人で行う学習と，教室で行われるペアの学習の違いの検討

より深い理解をしている者が手助けする可能性
- 教師がtutorを"teaching assist"として利用

REFLECTIONS

- ACT*のテストという最初の目的から長い道のりだったが，成果から，新しいACT-R理論(J.R.Anderson, 1993)が導かれた．
- 私たちは，最初のコンセプトであった，人間のエミュレーションからのtutoringを，放棄した．

→ 学習環境としてのtutor(学習に役立つ情報や問題を与える)．

→ 生徒の問題解決の認知モデルを持っているので，学習を助長することができる．
- ソフトウェアエンジニアリングの問題点についての言及．
- 高校教育の内容や目的についての問題点の描写．

The Curriculum Issue

tutor が実現するカリキュラムの特徴に関して，人々が持つ不安について答えた.

10 年間の研究が，何をどのように教えその教示の効果をどのように調査するかという，確かな観点を与えられたと信じたい.
プロジェクトで確信しているのは，課題となる問題は productio-rule 集合で表現できるということ.