解　説

知的CAIシステム探訪†

伊　藤　紘　二††

1. はじめに

現在までに，知的 CAI の理想を一身に実現した実用システムはないが，視点を絞ったところで，それぞれに興味あるシステムが試作されている．

本稿では，知的 CAI システムの欧米での試作例につき，先駆的なシステムが果たした役割を手短に概観したあと，それによってつくられた流れの最近の発展を探訪し，各システムの達成したことと，残された課題を論ずる．

2. 先駆的システム[1),2)]

知的 CAI の草分けとなった Carbonell の SCHOLAR[3)]は，故郷ブラジルの地理というスタティックな対象分野ではあるが，ユニット化された知識を階層的に連絡した知識表現によってはじめて，多様な質問に答えられる対話の理解と生成が実現できることを示した．知識単位ごとの簡単なオーバレイモデルではあったが，学習者モデルの有効性を実証した．

対話の研究は，自然に，ソクラテス的対話の研究に向かう．ソクラテス的対話では，事例を説明できる仮説を立てさせ，これを他の事例に適用させて，矛盾を発見させ，仮説を修正させる．WHY (Stevens and Collins)[3)]は，一般性のある因果関係を階層的に表したスクリプト表現と，それを具体的事例に適用する手続きをもち，ソクラテス的対話によって，雨が降るための一般性のある条件を発見させる試みである．

スクリプト表現を用いているかぎりでは，学習者モデルはオーバレイであるが，関係する物理量の間の増減関係の表現を導入することによって学習者のもつ誤った概念を見つけ出すことができる．

＊プロムナード＊

Carbonell は，知的 CAI における知識表現と，それに基づく指導方略の研究が，教育法自体の研究にとって，大きな意味をもつであろうことを，早くも予言している．

WHY の対話ルールは，生徒と教師の間の対話の念入りな分析に基づいたものであった．以下に紹介する知的 CAI の研究の多くも，一度壁にぶつかり，教育現場での protocol によって，それを乗り越えている．

WHY の知識表現は，扱う対象に依存したものであり，学習者のもつ基本的な物理的概念の欠陥や欠如を見いだすことができない．しかし，WHY 以後，それが目指した「原理に基づく説明」は，「使える知識の伝達」において，中心的役割を果たすことが，ますます明らかにされてゆく (QUEST, GUIDON 2 参照)．

結局，物理過程に関する学習は，対象過程を，要素的な因果関係，また，複数の視点からの基本原理の与える拘束というプリミティブの組み合わせによって理解する過程である．したがってシステムは，この過程のモデルをもたなくてはならないが，組み合わせ的爆発を防ぐために，それは，必要な部分を「走らせてみる」ことができる「計算可能モデル」としての「メンタルモデル」でなければならない．このようなモデルによるならば，学習者モデルの記述と診断は，モデルにおけるプリミティブの間違った利用，不足，過剰として実施できる．

さて，問題解決のために「走らせることのできる」モデルとしては，人間に，自分が問題解決に用いていることを認識させることのできる表現をもった「ガラス箱モデル」のほかに，人間が用いる方法とは異なるが，システムとしては，それを用いて効率よく問題を解くことができる「暗箱モデル」がある[1)] (Brown and Burton)．

汎用性のあるガラス箱をつくることには多大な困難が予想されたであろう．研究は，「暗箱モデル」を使った応答的環境によって，何が達成できるかを追う方向へ向かった．

＊　　＊

学習者は，具体的な問題解決をとおして，汎用性のあ

† Intelligent Tutoring Systems: A Guided Tour by Kohji ITOH (University of Tokyo).

†† 東京大学工学部

る知識を手続き化し，使える知識に変換する[24] (Anderson) という．しからば，問題解決を支援する環境をつくることによって，学習者が，自らのうちにメンタルモデルを育てるのを支援することができよう．そのような試みを「暗箱モデル」を用いて行ったのがSOPHIE プロジェクトである．教育目標は，電子回路の故障診断のエキスパートを育てることであった．

SOPHIE (Brown and Burton. et al.) のバージョンI[1]は，暗箱として汎用の電子回路シミュレータSPICE をもち，それに，問題を部分問題に分解するための手続きと概念階層の意味ネットワークとが付け加えられた．また，自然言語による談話レベルの対話を効率よく実現するため，ATN タイプで，構文，意味カテゴリ（実は応答生成関数）を併用する Semantic Grammar とコンテキストの管理を用いたパーサが用いられた．

システムが故障を設定する．学習者が，測定の指示，故障仮説，部品交換の指示，help などを入力すると，システムは，測定の有効性の評価，測定結果，仮説を支持する/しない測定，測定と矛盾しない仮説のリストなどを返す．

SOPHIE-I は，暗箱によるシステムであったため，「なぜそのように診断するのか」について，原理的な説明を生成することができなかった．バージョンII，III で，これを可能にする試みがなされたが，対象とする回路に特化されたものでしかなかった．

＊プロムナード＊

SOPHIE のようなシステムを用いれば，素子を駄目にする心配から消極的になるといったこともなく，「我慢強く」時間の制約もないこのような教師の下では，自分の誤りから学ぶ貴重なチャンスを逸することがない．

SOPHIE プロジェクトのチームは，それがぶつかった4つの問題を解決すべく，独立した基礎的研究を行ったが，それぞれが，その後の知的 CAI の研究の大きな流れをつくりだした．その一つ，仮説選択に対する情報の有効さの判断を訓練する BLOCKS tutor[2]は，GUIDON プロジェクトの診断戦略につながり，コーチングの側面をとりあげた WEST は，MENO プロジェクトへ，学習者の振る舞いに基づくその知識の推定をとりあげた BUGGY は，バグモデルと学習者モデルの諸研究へつながった．

一方，SOPHIE プロジェクトは，汎用性のあるガラス箱メンタルモデル—原理的説明が生成できるような一の枠組みを，実現することができなかった．その後，de Kleer は，この基礎的な問題を追及し続け，部品の機能とシステムの構造から因果モデルを導く envisioning，ならびにシステムの諸変量の間の増減に関する依存関係を導く confluence なる概念を中心とした定性的推論とよばれるモデルを提唱するにいたった[4]．

しかしながら，envisioning は，部品の機能がカプセル化されていないと使えない．また，初心者のナイーブな物理と，エキスパートがもつに至る定性的モデル思考とは，質が異なるはずであろう．この2点については，QUEST が，ある程度の解答を与えることになる．

一方，実際の故障診断では珍しくない複合故障の取り扱い，そして，複数の仮説を同時に扱って，推論結果を比較しながら，矛盾するものを落としてゆくプロセスは，ATMS[5] (de Kleer) によって実現されることとなる．

最後に，当時の技術的制約もあって SOPHIE に大きく欠けていたのは，イメージに依存したインタフェースによって，メンタルモデルの形成を強力に支援できるという視点であった．この方向は，STEAMER が実現する．

＊ ＊

主体的学習環境としてのゲームに適切なコーチをつけ，学習者が具体的課題にコミットしている状況において解決方略 (issue) を指摘することにより，優れた発見的学習支援システムを構成しうる．算術ゲームである WEST (Burton and Brown) は，要所において，自分の最適解を作り，生徒の解をそれと比べる（差分モデル）ことで，学習者の issues をさぐる．

WUNPUS アドバイザ (Goldstein and Carr) (WUSOR) は，確率的，論理的な考え方を養うゲームであるが，より進んだアドバイザのために，Goldstein は，Piaget の発達心理学にならって，各発達レベルでの分化の程度に応じた知識と発達の階梯を表現する Genetic Graph を提案している．

＊プロムナード＊

学習者の issue をさぐるために WEST が用いた差分モデルの考え方は，その後の学習者モデル研究の出発点になっている．

学習者がもつ分野知識を発達的に捉えるという WUSOR の視点は，QUEST に引き継がれている．

＊ ＊

BUGGY (Brown and Burton) は，減算の技能を教えるという課題について，その手続きの階層のさまざまな点において生徒がもつと仮定されるバグを，モデルのなかにもち，課題に対して，バグを含む手続きによる結果を提示できる．これは，教師の卵を訓練するのに用いられた．

DEBUGGY (Burton) は，BUGGY を核とした診断システムであり，複数の問題に対する学習者の解答を基に，複合したバグを，オフラインで同定できる．

＊プロムナード＊

Burton らは，BUGGY が，バグ診断のための問題集合のよさを評価する尺度に使えることを示唆している．故障診断における測定のよさの評価との類比がここにはみられる．

DEBUGGY で診断できない生徒が多くみられたことが，バグ生成の認知モデルの研究をスタートさせ，REPAIR，STEP theory が生まれ，また，診断法としての ACM が現れた．これらについては，本号「知的 CAI における学習者モデル」に詳しい．

＊　　＊

現実に遭遇する多くの問題は，算術計算のような手順が決まっている課題と異なり，可能な着手の組み合わせと発展のなかから解を発見するという非決定性のプロセスを欠かすことができない．LOGO は，そのような問題解決と発見の訓練を，早期に行うための環境として，提唱されている[6]が，SPADE (Miller) は，LOGO のプログラミングの学習を，汎用性のある階層的プランニングの方法の習得であると捉えて，このような方法の実行を誘導し，支援する環境を提供しようとするものである．

G. Polya[7] に従って，仕様からプログラムコードを決定する問題解決の過程を，問題の理解，解決の計画，解決の実行，解の吟味というフェーズに分け，システムは，それぞれのフェーズに対して，支援ルールをもっており，メニュー方式で作業を促す．

＊プロムナード＊

SPADE は，プランニングの誘導をともなうエディタにとどまったが，プログラミング言語の教育においては，仕様からコーディングまでの段階的詳細化の階梯を，初心者といえども，はっきり意識させるべきであるとする SPADE の考え方は，最初の失敗の後の MENO プロジェクトに受け継がれることになる．

＊　　＊

PSM/ACE (Sleeman) は，核磁気共鳴スペクトルのデータと分子式を基にして有機物の分子構造を推定することのできるガラス箱分野エキスパートと，このような課題を解く過程で学習者にアドバイスを与える指導エキスパートを接続したもので，分野エキスパートは，一種の制約伝搬を用いる．学習者に，自分の答についてなぜそう考えるかの説明を入力させ，それにたいしてコメントするために，Semantic Grammar と差分モデルを用いている．

＊プロムナード＊

分野独立な指導エキスパートを目指した点では，GUIDON とならぶ先駆的試みといえる．GUIDON と同じく，基本原理に遡った説明はできなかった．

3. STEAMER と QUEST

STEAMER[2),8)] (Hollan, Stevens, Hutchins. et al.) は，SOPHIE プロジェクトの影響下に，船舶の推進機関としての蒸気タービンプラントの運転要員を訓練するための学習環境としてつくられた．要員の教育は，決められた場合に対処できる手順だけを覚えさせればよいというわけにはゆかない．実際，それは，ただ覚えるには，あまりに多岐にわたっており，そのうえ，予測できない異常事態の場合には，独自の手順を編みださねばならないことがいくらでもあるからである．

そこで，エキスパートが，この複雑な物理系における物理過程の理解を可能にするのに用いているであろうメンタルモデルを，初心のユーザのなかに育てる必要があり，そのために，「interactive and inspectable simulation」をその，基本的メカニズムとして採用し，物理的構造よりも，概念レベルでの忠実性をねらいとした．

STEAMER は，内部的には，数学的定量モデルでシミュレーションを行う点で，SOPHIE-I と，同様であるが，イメージ上における任意変量の入力と動的な表示という direct manipulation interface[27)] をフルに利用して，定性的なメンタルモデルの形成を支援しており，また，学習者のイニシャティブで，任意の部分プロセスの抽象化されたシミュレーションができる—minilab という—ために，必然的に原理に戻った思考が促される．実際，システムの動作の，人間エキスパートによる説明は，必ず抽象されたデバイスモデルに還元し，抽象されたプロセスとして，階層的に原理に遡りながら，具体例でこれを支持する形をとることが観察されるという．

学習者は，実機関の制御入力のほかに任意変量の入力や，故障の設定を行うことができ，また，大局的な動作状態とサブシステムの動作状態の間をシステム構成の表示をポイントすることで，階層的に上下できる．さらに，ダイナミックなゲージパネルの表示，実機関には存在しないが，理解を助けるための視点における表示（パイプ内フローレートなど）が設けられ，変量/変化率の動的なイメージ表示が行われる．時間を止めながら，多変量の過渡状況を観察することもできる．

オーサリングについても，オブジェクト指向で実現されたグラフィックエディタにより，プログラマではない専門家が，システムの多くの部分をつくることを可能にした．ツールとしては，Flavor が用いられ，多くのグラフィック要素のおのおのが，objects として，それを扱う demon とともに登録されている．ユーザは，必要な objects の icon にポイントし，しかるべき位置にこれをもってきてパラメタを指定すると，これらを組み合わせた new Flavor がつくられる．"tapping demon" によって，component objects 相互の接続を行うとともに，数学モデルの変数との結合を達成する．

＊プロムナード＊

STEAMER の最終目標は，具体的な物理過程を，一般化されたコンポーネント，過程，原理の，構造的，機能的，トポロジ的結合のインスタンスとして，理解させる（原理に基づいた説明）仕組を組み込むことにある．Forbus の QP (Qualitative Process) 理論[9]は，その基礎固めの一環である．これは，atomic な部品の状態の間の相互作用の結果としてプロセスがあるとする envisioning とは相補的に，atomic なプロセスの組合せが状態の変化をひきおこすとみる見方にたっている．

＊ ＊

QUEST[2),10),11)] (White and Frederiksen) は，STEAMER が実現した「interactive and inspectable simulation」をうけつぎ，かつ，それがインタフェースで実現した概念レベルでの忠実性を，WHY が目指した「原理に基づいた説明」を可能とする内部表現にまで推し進めた学習環境を作るため，de Kleer, Brown 流の定性的推論に，WUSOR が目指したような発達認知的な視点に立った多層モデルを導入しようとする試みである．

電気回路の原理的な理解が教育目標であり，そのために，次のような軸において，発達的な視点から，階層的な複数のモデルが考えられた．

（1） 変化の扱い方のレベル：電圧，電流があるかないかのレベル，電圧の変化が，どんな電流の変化をもたらすかのレベル，変化の変化率のレベルなど．

（2） 複雑さのレベル：構造的情報，部品の記述から回路の動作を推論し説明する機能，因果推論の制御(時間順序的因果と，拘束伝搬的なそれの折り混ぜ)．

多層モデルに課せられる条件は，上位両立性，すなわち，より下位のモデルにおける reasoning が，より上位のモデルでの reasoning において，矛盾を生じないことである．

学習環境として，つぎのようなモードが提供された．

（i） problem-driven mode：問題を与えられて，help が受けられる．

（ii） example-driven mode：デモに続いて，演習をうける．

（iii） student-directed mode：学習者が，モデルに結びつけて問題を選択して，自分のコースをつくる．

（iv） open-ended exploration mode：学習者が，部品を使って回路を組み立てて，動作を試し，それについて，定性的な説明を受けられる．

学習者に与える問題としては，部品の振舞の予測，ある振舞をするような回路の設計，故障箇所の発見など．問題は，対応した水準のモデルに結び付けられて，そのなかでも，典型例，反例，バグ生成，バグ矯正などに分類されており，常に，モデルを走らせることによって，学習者を支援する．

ところで，多くのアナログ回路では，基本要素部品の状態の間の相互拘束が強いので，要素部品間でのデバイス独立な envisioning は，もともと不可能である．QUEST は，この点を，部品の状態の相互拘束を，デバイストポロジの探索によって見出して，結果として得られるグローバルな状態を接続する形で，困難を解決している．これは，envisioning に対する本質的な改良である．

また，発達的な視点から，階層的なモデルに基づいて，学習者モデルをつくり，学習者が到達している水準のモデルに隣接するモデルを必要とする問題を与える．支援の説明も，学生の到達水準モデルと新しいモデルとの差に焦点をあてる．

中等教育の生徒に使わせた結果は，電気回路に対して，かなりの間違った捉えかたをしていた7人の生徒の全員が，1日1時間，5日間の独習で，回路のどこ

がオープンか，絡地しているかについて，誤りのない診断ができるようになったという．

＊プロムナード＊

QUEST の envisioning は，結局，状態の接続としてプロセスが記述できる大きさの視野で状態を記述する一つの方法を提示しており，この理論への本質的な寄与といえよう．

White と Frederiksen は，QUEST のようなシステムによって生徒が獲得するメンタルモデルの限界は，システムのもつモデルの限界そのものであり，トータルな教育システムとしては，誤った概念を診断して矯正するという方略よりも，システムにもたせる段階的なメンタルモデルを，確かで精密なものにすることのほうが，重要であると考えている．

4. LRDC プロジェクト

LRDC (Learning Research and Development Center at the University of Pittsburg) のチームは，WUSOR の genetic graph が目指したような発達的な知識表現に基づき，課題に対する学習者の応答から，指導すべき issue を探り，ダイナミックに，カリキュラムを生成してゆくための汎用性のある手続きを見出すことを目指している[2),12)]．システムは，学習者が自由に分野知識を探訪するのを許すモード，学習者の問題解決に際し，必要に応じてコーチするモード，システムが強力に指導するモードの三つの機能を提供する．

bite-sized tutoring architechture と称されるシステムの基本構成は，おのおのが issue に相当する分野知識のユニット—bite—からなるネットワークを，オブジェクト指向の考え方で構成したものであり，さらにその上に，カリキュラムユニットの指導ゴール/サブゴールネットワークからなる層—知識層を refer するポインタをもつ—，最上層にメタレベルの能力を表現する層を置くことが提案されている．

知識層とカリキュラム層の分離は，同じ指導目標を，多様な主題で構成することを可能にする．

各 bite オブジェクトには，他の bite との概念階層的な関係のほか，前提知識と確認知識，ミニアチュア学習者モデル，当該知識についての診断機能，問題生成機能，指導コメントの生成機能などが記述される．

経済を教えるマイクロワールドシミュレーションに，bite-sized tutoring を組合せたもの，また，電気に関する原理を教えるシステムにおいて，学習者の知識状態の変化に関して最大の情報が得られるような課題を生成する機構を試みている．

＊プロムナード＊

学習者の思考の，深層における構造を，その組合せとして診断できる程度まで，任意の分野の知識のユニットを “bite” してゆくことは，現在の知識処理技術では，不可能である．したがって，汎用のシステムを目指すならば，LRDC プロジェクトのように，知識の “biting” を，カリキュラムが柔軟に組み替えられるような構成単位というマクロな水準に留め，診断も，そうした知識単位に閉じて行うのが現実的であろう．

5. MENO プロジェクト

MENO[2)] は，初心者に Pascal を教える知的 CAI プロジェクトであり，その目標は，学習者のプログラムにおいて，構文的なレベルでない間違いを診断し，その原因となっている間違った構想を探り，その構想を矯正するような指導をすることにあった．

最初のシステム MENO-II[13)] (Soloway. et al.) は，繰り返しブロックに焦点を当て，繰り返しブロックのカテゴリ対プログラムパターンの知識と，バグライブラリとから，学習者のプログラムをパースし，その結果を，間違った構想のネットワーク対バグの対応に関する知識と照合することによって，間違った構想を指摘する．

このシステムは，プログラムの段階的作成過程とデータの流れを視点に入れていなかったために，学習者のおかす間違いの多くを正しく診断できなかった．

＊プロムナード＊

スタッフは，研究の方法論についての反省にたって，実際に学習者が犯すバグを収集し，他方で，プログラミングのための知識の体系をつくる研究を行った[14)]．対象は，繰り返しブロックに絞った．

とくに，初心者が，自然言語により，すなわち人間が人間に対して，手順を説明をするときの表現と，プログラミング言語によって手順を表現しようとする過程を比較観察した．その結果，二つの言語は，機能的には同様のタスクを表現することができても，表層のレベルで似ている表現が，機能のセマンティクスとしては，ほとんど重なっていない，という点が，初心者のつくるバグプログラムの多くを説明することが分かった．実際，初心者は，プログラミングに関する自分の知識が不足しているとき，自然言語の表現の外挿

で，これを補おう (repair) とするのであった．

MENO からは，BRIDGE，PROUST，MENO-TUTOR の三つのプロジェクトが派生した．

* *

MENO スタッフが見出した上記の結論は，学習者において，プログラムのプランニングに固有のメンタルモデルを成長させる必要を意味する．BRIDGE[2] (Bonar) は，自然言語を使って，ゴール記述，データ単位でのゴール記述，ステップの記述，プログラミング言語レベルの仕様，という4つの段階を追ったプランニングを行わせることによって，自然言語と，プログラミング言語との間の谷間に橋をかけようというものである．システムは，学習者に対して，使える語句のメニューを示し，学習者は，このメニューから選んだ語句の組み合わせで，プランニングを記述する．このことにより，システムは，汎用の自然言語処理を行うことなく，学習者の意図と，どの段階でプランニングしているか，を把握することができる．BRIDGE は，これに基づいて，段階によって異なるコーチを行う．

第4段階まできた学習者には，得られたプランのユニットを，グラフィックな表示—ここでは，すでにシミュレーションが可能—を用いて接続させ，ここで始めて，プログラミング言語への変換を支援する．

＊プロムナード＊

BRIDGE は，SPADE と同様，階層的プランニングというシステムの依ってたつ理論を全面にたてて，学習者とのコミュニケーションの主役としているが，階層の方向は，プログラミングの知識の形成過程にそっている．

プログラミング学習支援の研究は，自動プログラミングの研究と重要な接点をもつ．

プログラミングは，基準化された表現の組合せによって，最初の自然言語レベルの仕様が記述できる程度に抽象化された分野であって，支援システムにおける自然言語による対話処理も比較的楽である．

* *

PROUST[2),15)] (Johnson and Soloway) は，学習者のつくったプログラムについて，そのプログラム設計のプロセスを，再構成することによって，診断を行うシステムである．

プログラムコードだけから，学習者のプランニング過程を推定することはむずかしい．ちょっとしたコードの間違いや，部分同士の絡み合いによって，判断を誤るからである．

そこで，PROUST は，プログラミングの過程に，ゴールとサブゴールの設定—意図の表現＝問題記述—，プランの選択，コードによる具体化の3段階を区別し，各段階での仕様がもたらす拘束を用いた analysis by synthesis によって，学習者のプログラムを解釈し，これに基づいて，ロバストで意味のある診断メッセージをつくりだす．

システムは，ゴールカテゴリと，それを達成するためのプランのリスト，そして，プランをコードで表現するためのテンプレート，マッチングを調整するための変換ルール，バグルールといった知識をもっている．

課題そのものと，システムが理解できる形の問題記述は，システム管理側が入力しておく．

学習者が，システムの提示する課題に対して自分がつくったプログラムを入力すると，システムは，問題記述のゴールから，上述の知識ベースの縦型探索によって，プランを検索し，テンプレートを求め，必要ならバグルールを援用して，それにマッチするコードをプログラムのなかに探す．この際，グローバルな制御プランを優先する．部分的にマッチングがとれると，一般に，プランのなかにあるサブゴールが起動され，再帰的に進行する．一般には，複数の候補が求められ，そのうち，より少ない個数のバグを仮定して，よりよいマッチングがえられたものを，学習者の構想と結論し，「学習者の意図に添った形での」正しいプログラミング過程との比較により，各レベルでの誤りを指摘することができる．ただし，マッチングの度合いがよくないときは，その旨を告げる．

PROUST の評価試験は，206 名の学生のプログラム (数を読んで平均を求める)—89% がバグを含んでいた—の 81% を，確信をもって解釈し，解釈できなかったのは 4% であった．バグ (795 個) のうちの 78% に対し，誤りを正しく指摘した．もっと複雑なプログラムでは，50% を解釈し，64% のバグを正しく検出した．

PROUST の診断をベースにした指導システムが現在，Yale 大学で開発されている．

＊プロムナード＊

PROUST は，トップダウンなやり方の採用により，複合したゴールに対する取り扱いを可能にした意義が大きい．しかし，完全にトップダウンなパーシングは，ゴールに対して実現法の自由度が大きいプログ

ラムにおいては，実行不可能になる．かといって，ボトムアップは，制約が少なすぎてこれまた爆発する．そこで，Johnson は，学生に，システムに意図を伝える対話を行わせながら，プログラミングを進めさせる方法をとろうとしている．

* *

MENO-TUTOR[16] (Woolf and MacDonald) は，検出されたバグに基づいて，それを矯正するためのソクラテス的対話を生成する機構である．分野に依存しない対話管理ネットワークが，分野依存の文生成機構を制御するという構成をとっているが，後者の自然言語処理機構については，まだ簡単なものに留っている．対話管理ネットワークは，一種の ATN であり，教育目標的，戦略的，戦術的の順に詳細化される対話の焦点を状態として表すノードと，その間のデフォルト推移を指定するアークが設定されている．さらに，学習者モデル，談話モデル，分野モデルに依存した条件で起動され，デフォルトにかかわらず焦点を移行するためのメタルール集合がある．

システムは，たとえば，基本事項の理解を確認した上で，バグの背景を探り，学習者が正しく把握している視点を強調したうえで，誤った考え方を指摘する，というふうに対話を進行させる．分野としては，WHY のテーマであった降雨の因果関係と，MENO のテーマである PASCAL 言語について試みられた．

＊プロムナード＊

このシステムの目的は，対話の焦点を表すノードの集りを，対話構成のプリミティブとし，メタルールをさまざまに替えて，どのような対話制御がよいのかを探求するための道具として役立つことである．

6. GUIDON プロジェクト

MYCIN[17] は，ガラス箱エキスパートであるプロダクションシステムにより，患者に関するデータをもとに，感染症の診断と処方の提案をすることのできるエキスパートシステムである．MYCIN における，分野独立な推論機構 EMYCIN は，まるで無関係な他のいくつかの分野で利用することができた．このことは，分野エキスパートとも推論機構とも独立な指導エキスパートモジュールをつくることにより，分野エキスパートさえ入れ替えれば，任意の分野の知的 CAI を構成できる可能性を示唆する．

GUIDON[18] (Clancey) は，このような試みの最初のものであった．指導エキスパートもプロダクションシステムで構成しており，推論の完全なトレースをとるという機構を MYCIN に付加し，症例について得られる case solution tree がもつさまざまな情報に基づいて臨床医の卵と対話を行うことにより，現実的な問題解決の文脈において，診断知識を伝えようとする (issues and examples, ソクラテス的対話) ものであった．

学習指導モジュールには，コミュニケーションモデルがあり，指導ルールは，コミュニケーションモデルの与える対話の焦点と学習者モデル（オーバレイ）に基づいて，採りあげるべき分野ルールを選び，対話のパターンを決定し，学生による入力を受付け，コミュニケーションモデルの更新を行う．

対話のパターンとしては，学習者側が，診断仮説を提出し，あるいは，判断に必要と思うデータを要求する．システムは，学習者モデルを参照して，学習者が，そのような提案や要求をするに至るのに用いたルールを推定し，差分モデルにより issue を探り，それに基づいて，批評し質問する．

＊プロムナード＊

GUIDON に，知識が欠如しているという指摘をされても，自分の知識のなかに MYCIN がもつ専門家レベルの知識へのつながりをもたない学習者において，その指摘は，なんの意味ももちえないのであるが，MYCIN には，診断のためのマクロな方略や，ルールのよってたつ根拠が，対話を可能にするような形では表されていない．ルールの構造も，フラットなプロダクションシステムの制約のもとで診断のパフォーマンスがあがるように，拘束や制御的な意味合いの条件を合わせもたせたものになっており，学習者にとっては，理解しにくく，また記憶もできない知識になっていることが分かった．さらに，推論が，病原仮説から後ろ向きに行われることを前提にルールができているため，人間が行う，前向き後ろ向きを組合せた推論とは，咬み合わない場合が少なくないことが明らかにされた．

こうして，GUIDON の問題点は，その分野エキスパートの MYCIN の知識表現にあることが明らかとなったが，Clancey らは，医学教育の現場における名講義の観察により，分野の知識を，分野に依存しない汎用の診断方略の体系の枠組みのもとに編成し，そのような方略の体系を陽な形で訓練する対話を構成する必要があることを見出した．そのため，結局，NEOMYCIN と名付けた新しい分野エキスパートを

つくることになった[19].

MYCIN の知識ベースは，いわば，水面上に見えるコンパイルされた知識だけをもっており，これが，知識の伝達に使えるものになるためには，それらを成り立たせている水面下の構造まで知識を「デコンパイル」する必要があるのであった.

NEOMYCIN は，この水面下の構造の表現として，医療診断一般に通用する汎用の診断方略—診断思考法—を階層的に表現した「メタストラテジ」エキスパートをもち，それによって，診断仮説の空間の管理が行われ，すべての推論が，分野知識の自律的な接続ではなく，こうしたメタストラテジルールによる管理のもとに，分野知識がケースに適用されることによって進行する.

活性化されている仮説集合は，differential というデータ構造に置かれ，その記述に基づいて仮説集合の評価に有効なデータのタイプが決められ，あるいは，データに基づいて活性化された仮説がそこに入れられる．differential による制約を利用することによって，焦点を絞った前向きの推論も可能になり，人間の行う推論に近づいた.

実際，この推論方式により，MYCIN が扱った範囲をこえて，関連疾病との弁別を行うことができるようになった．また，differential に基づいて，対立仮説の評価というロジカルな視点から焦点の推移を行えるので，ユーザの思考に添ったプロセスが実現できる．さらに，診断過程の説明が，特定の仮説に対する診断方略のゴールとタスクという視点で行えるようになった.

分野知識のほうも書き換えられた．まず，MYCIN のルールに混在していた戦術拘束的情報が除去され，代わりに，メタストラテジが，分野知識を検索するときに，メタルールをインスタンス化する（ふるいわける）のに使われる抽象化された視点情報が付加された．すなわち，知識は，一般原理，日常的事実，定義，分類，因果関係，ヒューリスティックス，などのカテゴリに分類され，さらに，因果的，病原学的，病歴的といったグループに集約され，こうした多様なウィンドウを通して，メタルールは，分野知識の部分集合にその焦点を当て，仮説の生成，起動，検証，削除などの仕事を，効率よく行うことができる.

* *

GUIDON 2 は，まだ，部分的な構想の試作を重ねている段階であるが，GUIDON に代わって，NEOMYCIN に実現されたような知識エキスパートの透明さを前提に，汎用の知的 CAI の枠組みを再構成することを目指している．Clancey は，その目標を「与えられたデータに対して，ちょうど診断におけるように，なんらかの意味で対応したクラスを見出すという「発見的分類」を行うという問題解決の能力を教育する汎用の generic なシステム」と考えている[20].

扱っているのは MYCIN の分野であるが，構想は十分に汎用性がある.

HERACLES[20] (Clancey) は，その基本となる「発見的分類」を行う分野独立なシステムであって，ちょうど，MYCIN に対する EMYCIN に相当し，NEOMYCIN から，メタストラテジの部分を取り出して拡張したものである．その構成は，分野独立な問題解決方略と，対象の間の抽象化された関係を記述する言語—対象分野を，メタレベルの推論ができる形式に組織化するために必要—とからなる．EMYCIN が，プロダクションルールによる後ろ向き推論というインプリメンテーションの枠組みにおいてのみ汎用性をもっていたのに対し，HERACLES の汎用性は，抽象度が一段高い.

IMAGE[21] (London and Clancey) は，ちょうど，PROUST におけるように，学習者の解答を解釈して，彼が用いたであろうプランニングを再構成するシステムである．ただし，ゆくゆくは，指導エキスパートと組合せることを目指しており，問題解決の過程において，オンラインで，学生の入力を見ては，部分的プランと，学習者の振舞の解釈を更新してゆく．解釈は，HERACLES の提供する問題解決方略生成機構を利用する．まずは，トップダウンに，当面の仮説から学習者の振舞を予測する．これが学習者の振舞いを説明できなかったとき，はじめて，ボトムアップの探索を起動する．後者においては，探索を絞るためのヒューリスティックを用い，また，HERACLES の提供するランクづけした予測を利用して，爆発を防ぐ．このランクづけされた予測は，学習者の振舞いを評価する，あるいは，つぎのステップへのアドバイスをつくりだすのにも使える．最近のバージョンでは，プラン上のバグモデルも入れている.

学習者の振舞いの，複数の視点から解釈を提供することが，IMAGE の特徴であって，これを利用して，複数の指導方略を示唆する能力をもっている.

ODYSSEUS[22] (Wilkins. et al.) は，もともと，machine learning の手法を用いて，NEOMYCIN の

分野知識を改定するためのプログラムであり，エキスパートの入力に対し，自分に欠けている知識，違っている関係を，HERACLES による拘束を利用して検出する「learning apprentice」であった．

同じプログラムを，学習者モデルをつくるのに，利用できる．すなわち，HERACLES の戦略は共有しているものとし，学習者の振舞いを，エキスパートの知識を用いてボトムアップに複数の「推論の道筋」を立てて解釈しつつ，解釈できなくなったら解釈できるまで，学習者のもつ知識に関する仮説を更新する．IMAGE に比べて，data-driven なので，自由度が大きいため，HERACLES のシミュレーションや優先度のヒューリスティック，学習者の戦略スタイル，解釈の整合性などによる枝刈りを必要とする．

実は，ODYSSEUS 自体，学習者の振舞いを解釈するという「発見的分類」を行っている．したがって，インタプリタを HERACLES で書くことができる．

以上の学習者の振舞いを評価するシステムとともに，HERACLES を規範にして「発見的分類」能力を学習させるためのモジュールが，いくつかの視点から作られている．

GUIDON-WATCH[23](Richer and Clancey) は，アニメーションとウィンドウを利用して，HERACLES の推論戦略を，学習者に見られるものにするシステムである．

GUIDON-MANAGE[2] (Clancey) は，学習者自身が，問題解決オペレータを操作し，結果をグラフィックな表示によってみながら，問題解決の筋道を成長させてゆくことができる支援環境である．

GUIDON-DEBUG[2] (Clancey. et al.) は，学習者が，システムによる問題解決過程を評価することによって，システムの知識ベースを改定することができるもので，ODYSSEUS をインタフェースにしている．故意に知識のバグを設定することにより，学習者にそのバグを発見させるなどの使い方が試みられたが，指導プログラムなしでは難しいようである．

＊プロムナード＊

Clancey の主張するように，汎用の問題解決方略を pivot にして分野知識を扱うというパラダイムは，学習というプロセスの本質に大きな示唆を与えるものといえよう．しかし，分野知識によるインスタンスが，汎用の問題解決方略の獲得につながるプロセスの解明がなされなければ，問題解決方略自体を教育するための真に有効な指導方略をたてることは難しいだろう．

7. ACTP プロジェクト

心理学者 J. R. Anderson は，知的 CAI を，認知心理学の検証の場と考えており，ACTP[2] は，彼の認知学習理論である ACT* (Adaptive Control of Thought) 理論[24]に基づいて，CMU における彼とそのグループが進めている知的 CAI プロジェクトである．

ACT* によれば，問題解決とは，解決過程の状態と，ゴールの両方に依存して，適用可能性が決められるようなプロダクションルールの集合から，適切なルールを選んで適用してゆく過程である．そして，学習とは，宣言的に与えられる知識を，問題解決行動をとおして，プロダクションルールに手続化し，かつ，頻繁に現れるルールの組合せをコンパイル（チャンク）して，マクロなルールをつくってゆく過程である．

ACTP は，学習者が，このような知識の手続化とマクロ化を，問題の解決をとおして達成するのを支援する．ACTP のもつルールは，MYCIN のそれとは違って，かなりプリミティブなレベルのルールであり，正しいルールと合わせて，バグルールももっている．

支援は，常に，ゴールスタックと，学習者が行った問題解決のトレースを提示していることによって，学習者の短期記憶容量の不足を補う．また，問題解決の各ステップにおいて，正しいルールで説明できない，あるいは，バグルールで説明できる道を，学習者がたどりはじめたら，介入して，そこにおける最良の汎用の知識を提示する．システムは，次のステップをメニューから選ばせることで，学習者の解決行動を知ることができる．

特徴は，ルールが強いゴール指向のものであることである．これは，問題の解決を，ゴールのサブゴールへの分解という形に表現し，常にその表現に関係づけて，なぜという質問に答えられる点で利点がある．

これまでに，初等幾何学の証明と，LISP によるプログラミングを対象としたシステム[25),26)]がつくられ，現場での評価も行われたが，できない生徒について効果が大きかった．ゴールと解決過程が陽に提示されることが，学習者の思考を助けているらしい．これらは，市販もされている．

進んだ生徒は，このシステムを押し付けがましいと感じるようで，指導を柔軟にすべく改良が行われてい

るが，このことによって，探索空間が大きくなり，解の探索に時間がかかりすぎる．そこで，可能な解の道筋をオフラインで生成しておくという手段が使われる．

＊プロムナード＊

プリミティブなルールの組合せは，任意水準の学習者の振舞いを説明できるはずだが，ACTP のように，プリミティブなルールをフラットにつないだ形で，高度な知識を表現し切ることは，困難であり，たとえ書けたとしても，探索空間が大きくなりすぎる．結局，学習者の水準に合わせたルールセットを，手作りで書き，各水準では，その水準のルールのみを用いている．したがって，知識のチャンキングという学習モデルは実現できていない．これができれば，学習者がチャンキングを達成するにつれて，プリミティブを高度化することができ，探索空間を増やすことなく，診断ができるようになるはずである．

ところで，GUIDON の問題点は，MYCIN の知識がコンパイルされているために原理的な水準で話ができないことにあり，知識を階層的にデコンパイルする必要があった．確かに，ルーチン化した（部分）問題の解決を効率的に行うには，コンパイルされた形で知識を使うのがよいが，知識を，それをもたない者に伝達するためには，新しい（部分）問題を解く場合と同様，プリミティブな知識の新しい組合せをつくりだすメタレベルの知識―つまり深層における類比―といった階層構造にデコンパイルされた形がいる．ACTP には，メタレベルのルールがなく，その狭い問題カテゴリに特化された方略知識は，型にはまった思考を教えるおそれなしとしない．

8. むすび

教育は，学習者という対話の相手に関するモデルを成長させながら，これに基づき，相手のなかに，問題解決方略を含むできるかぎり汎用の知識を成長させることを目的として，対話を構成してゆくという，最も深い意味での「知識の伝達」[2]である．知的 CAI の研究は，コンピュータを介して，真の意味での知識の伝達を支援するシステムをつくることにあるが，ここで紹介した諸例から知られるように，その実現のために，学習現場での試行を通じて，知識伝達の計算モデルをつくることが，この研究の大きなサブゴールとなっている．これは，発達認知的な視点を必要とするために，認知科学との密接な共同作業が必要となる．

一方，人間生活に影響の大きいこれからの人工知能システムは，人間と機械の共同作業によってのみ，その真価を発揮するであろう．そのために，システムと人間は，互いに自分がもつ知識を伝達し合えなければならない．したがって，知識伝達の研究は，人工知能研究において，中心的なテーマの一つとなるだろう．

参考文献

1) Sleeman, D. H. and Brown, J. S. (eds.): Intelligent Tutoring Systems, Academic Press (1982).
2) Wenger, E.: Artificial Intelligence and Tutoring Systems, Morgan Kaufmann (1987).
3) Bobrow, D. and Collins, A. (eds.): Representation and Understanding: Studies in Cognitive Science, Academic Press (1975).
4) de Kleer, J. and Brown, J. S.: A Physics Based on Confluences, Artif. Intell., Vol. 24, pp. 7-83 (1984).
5) de Kleer, J.: An Assumption-Based Truth-Maintenance System, Artif. Intell., Vol. 28, pp. 127-162 (1986).
6) Papert, S.: Mindstorms: Children, Computers and Powerful Ideas, Basic Books (1980).
7) Polya, G.: Mathematical Discovery, Vol. I, II, John Wiley & Sons (1962).
8) Hollan, J. D., Hutchins, E. L. and Weitzman, L.: STEAMER: An Interactive Inspectable Simulation-Based Training System, AI Magazine, Vol. 5, No. 2, pp. 15-27 (1984); in Kearsley, G. (ed.): Artificial Intelligence and Instruction, pp. 49-67, Addison-Wesley (1987).
9) Forbus, K.: Qualitative Process Theory, Artif. Intell., Vol. 24, pp. 85-168 (1984).
10) White, B. Y. and Frederiksen, J. R.: QUEST: Qualitative Understanding of Electrical System Trouble Shooting, ACM SIGART Newsletter, No. 93, pp. 34-37 (1985).
11) White, B. Y. and Frederiksen, J. R.: Intelligent Tutoring Systems Based upon Qualitative Model Evolution, Proc. of the National Confernce on AI, Philadelphia, pp. 313-319 (1986).
12) Bonar, J. G. et al.: An Object-Oriented Architecture for Intelligent Tutoring, Proc. of ACM Conference on Object-Oriented Programming Systems (1986).
13) Soloway, E. M. et al.: MENO-II: An Inntelligent Tutoring System for Novice Programmers, Proc. the 7 th IJCAI, pp. 975-977 (1981).
14) Soloway, E. M. and Ehrich, K.: Empirical Investigations of Programming Knowledge, IEEE Trans. Software Eng., Vol. 10, pp. 595-609 (1984).

15) Johnson, W. L. and Soloway, E.: PROUST: An Automatic Debugger for PASCAL Programs, In Kearsley, G. (ed.): Artificial Intelligence and Instruction, pp. 49-67, Addison-Wesley (1987).
16) Woolf, B. P. and McDonald, D. D.: Building a Computer Tutor: Design Issues, IEEE Computer, Vol. 17, No. 9, pp. 61-73 (1984).
17) Shortliffe, E. H.: Computer-Based Medical Consultations: MYCIN, American Elsevier (1976).
18) Clancey, W. J.: Knowledge-Based Tutoring: The GUIDON Program, MIT Press (1987).
19) Clancey, W. J.: The Epistemology of a Rule-Based Expert System: A Framework for Explanation, Artif. Intell., Vol. 20, No. 3, pp. 215-212 (1983).
20) Clancey, W. J.: Heuristic Classification, Artif. Intell., Vol. 27, No. 3, pp. 289-350 (1985).
21) London, R. and Clancy, W. J.: Plan Recognition Strategies in Student Modeling: Prediction and Description, Proc. National Conf. on Artif. Intell., Pittsburgh, pp. 335-338 (1982).
22) Wilkins, D. C., Clancey, W. J. and Buchanan, B. G.: An Overview of the ODYSSEUS Learning Apprentice, in Mitchell, T., Carbonell, M. and Michalski, R. S. (eds.), Machine Learning, Academic Press (1986).
23) Richer, M. H. and Clancey, W. J.: GUIDON-WATCH: A Graphic Interface for Viewing a Knowledge-Based System, IEEE Computer Graphics and Applications, Vol. 5, No. 11, pp. 51-64 (1985).
24) Anderson, J. R.: The Architecture of Cognition, Harvard University Press (1983).
25) Anderson, J. R. and Reiser, B. J.: The LISP Tutor, Byte, Vol. 10, No. 4, pp. 159-175 (1985).
26) Anderson, J. R. et al.: The Geometry Tutor, Proc. IJCAI, Los Angeles, pp. 1-7 (1985).
27) Norman, D. A. and Draper, S. W. (eds.): User Centered System Design, Lawrence Erlbaum (1986).

(昭和 63 年 9 月 22 日受付)