コンピュータと教育　19－10
（１９９１．１１．１５）

ハイパーカードを使ったバグ診断システム

徳田　尚之（宇都宮大学）　安納　順一（富士通文教システム課）

本論文では,Van Lehn(1986)がその学習（修得）過程が'帰納仮説'で説明出来るとして挙げた66個のバグを階層的に分類し、原始的バグは単独でのみ現れ、しかもバグの移行は起きないという限定された条件下でのバグ診断システムを構築した。　階層構造は、１　借りを伴わないバグ　２　被減算に0を伴わないバグ　３　被減数に0を伴うバグと大分類しそれ以下は、各ノードに与えている問題群がバグを曖昧さなく区別出来るように選んだ。バグ探索は、最短で深さ２、最長で深さ７で完了する。このシステムは図形編集なども容易に出来るマッケントッシュのハイパーカード上で行った。複合バグ・バグ移行の取扱いについてはいま研究中である。

A Bug Diagnostic System by Hypercards

Naoyuki Tokuda and　Zyuniti　Annou

Faculty of General Education
Utsunomiya University
Utsunomiya,Japan 321

A hierarchical bug diagnostic system was built on Mac's Hypercards using 66 bugs examined in details by Van Lehn(1986) for establishing the induction hypothesis in the acquisition process of these bugs.The present system works efficiently,however,　only if primary bugs appear without compounding and bugs do not migrate. The bugs are grouped into three major partitions; 1　bugs not requiring borrowing , 2　bugs requiring borrowing with no 0's in top columns, 3　bugs requiring borrowing with 0's in top　columns .　An extension is now underway for dealig　with compound bugs and with bug migration.

1　いとぐち

知的CAIシステムの重要な構成要素の一つは診断システムで、具体的には学習者の犯す誤り（コンピュータ・プログラムからの比喩でよくバグと呼ばれる）を同定することにある。バグは色々な理由で生成されるし、その分析にも色々な方法がとられる。ITS(Intelligent Tutoring System)の分野では 学習者が修得すべき教材知識を手続きネットにより表現する方法が、最も古くから研究されて来ている。この理論ではバグは学習者が誤った手順を修得してしまうことにより生成され、その環境に直面すると学習者は常に全く同じ誤った手続きを実行してしまうと仮定する。多くの実例を分析して纏められたバグ群として有名なものとしては、生成の仕方、分類の仕方により30個から100個近くにのぼるとされる整数の減算でのバグ(Brown & Burton 1978 ,Brown & Van Lehn 1980,Van Lehn 1986,福田&徳田 1989)、68個の分数の加減算のバグ(Shaw et al. 1982),数十個の線形方程式のバグ (Sleeman 1984)などが有名である。ことに減算手続きについては、多くの初心学習者（おもに小学生）にとっては理解の困難な上位桁からの借りの演算を伴う代数的手続きを正確に修得する必要があり、バギーシステムとして最も詳しく研究されてきている。実際、何万人規模にのぼる小学生等の学習者を対象に行われた調査結果も蓄積されている。バグモデルの代名詞として使われている由縁でもある。実際に観察されるバグは、多岐にわたり、学習者が全くの気まぐれとか不注意で犯すバグ（引き算中他のことを考えていたとか、隣の生徒の答を移し違えるといった類のもの）を考慮の対象から外しても100個を越すバグ（他のバグと区別出来るという意味で原始バグという）が見つかっている（Brown & Burton 1978)。これらのバグはかならずしも原始的バグの形で現れるとは限らず複数個のバグが複合して現れたり（複合バグ compound bug という）、全く同じ問題を同じ学習者が解いていても時と場合によって違ったバグ（bug migration とよばれている）を犯すこともある。多数個のバグを持つバギーシステムのバグを同定する診断システムは、容易ではない。

Burton (1982)は、この難しいバギーシステムにバグ診断システム（Debuggy または IDebuggy System...Debuggyの会話型オンライン版... と呼ばれている）を実際に構築した最初の著者である。Burtonは、各原始的バグは正しい引き算手続きからの変形であり、それらのバグは学習者が与えられた問題を解く時単独または組合わさって出現するという前提条件をまず置いた。診断プログラムの目的は,要求されている引き算技能の修得レベルについてどの程度理解しているかといった通常のテストが目的とする修得レベルの評価ではなく,学習者の修得した（修得していない？）引き算という代数的手続き技能のどの副技能部分（すなわちバグ）を診断することにある。Burton (1982)のDebuggyシステムは大変野心的で、110個の原始バグと４個までの複合バグを許すというものである。まず、６０個近いバグを与えられた問題群に対して正誤に関して論理的等価性を定義してバグの格子構造（実質的にはグラフ構造）を作り上げた。

この構造を最大限に使って膨大な仮設空間の枝刈に利用する。バグ構造がグラフ構造のため、効率の良いバグ探索にはアドホックな方法、巧みなヒューリスチック戦略が不可欠となる。

福田&徳田（1989）は、このバグ同定にもっと合理的な推論計算法として、Dempster-Shafer理論を使ったEvidential Reasoning計算法を使うことを提唱した。計算の対象にBrown & Van Lehn(1980)の修復理論が生成した階層構造を持つ約36個のバグを選んだ。もし複合バグを許さず階層構造を持ちしかも原始的バグは単独でのみしか現れないという限定された条件下のバグ探索には、ノード上の証拠の影響はそのノードの属する家族メンバー間に局所化されるという特徴を利用したShafer & Logan(1987)または Gordon & Shortliffe (1985)のアルゴリズムが非常に有効であることを示した。事実、バグの数 n についてDempster-Shafer理論なら指数関数的爆発をする計算量がこれらのアルゴリズムを使うとO(n)まで減少させられることを示した。実際36個程度のバグ数であれば、バグを階層的に構築出来れば、Dempster-Shaferの確率推論計算法でバグの同定が実用的に可能であることを示した。本論文では、Van Lehn (1986) がこのバグ獲得過程を含めた代数的手続きの学習過程が帰納仮設に密接に依存していることを明確にした66個のバグについてこれを階層構造に組み直し、前報の確率的推論計算を実際には使わなくともバグの同定が可能なシステムが構築できたので報告する。

2　学習理論とバグの階層構造

Brown & Burton(1978)のバギーシステム、その結果を基にBurton(1982)が構築した診断システムDebuggyまたはIDebuggyシステムでは、多くの学生に観測された代表的なバグを構造化する方法を提唱しているがどちらかというとアドホックに原始的バグとして列挙してきているにすぎない。例えばBurton(1982)は、与えられた問題群への答が正・誤のどちらを示すかによりバグの構造化を実行して格子構造を提唱しているが、出来たバグ格子の組織的構造は、複雑で理解しにくいという大きな欠点をもつ。　一方、Brown & Van Lehn(1980)は、バグの生成過程を説明しうる理論として修復理論を提唱した。これは、'正しい引き算の代数的手続き' を構築する6個のゴール・サブゴールの中に組み込まれた12個のルールのどれかを故意に除去し、演算手続きが続行不可能となり行き止まり状態(Impasseという)になったら、組み込んであるいくつかの'ヒューリスチック'を頼りに修復作業を行い手続き的には間違ったいわゆるバグを生成するという方法である。組み込むヒューリスチックにもよるが、出来るだけ一般性を持つ'ヒューリスチック'のみを組み込んだ Brown & Van Lehn(1980)のプログラムではやく40個程度のバグが生成出来たが、実際に観察されたバグはその内の14,5個に過ぎず、なかには人間の学習者では犯しえないバグも多い。我々にとってこの修復理論が明確にした最大の利点は、実はバグが階層構造を持ちうることを示唆してくれたことである（福田&徳田1989)。実際、除去

したルールにより、
1　借りの作業を伴わないバグ群
2　借りの作業を伴うが被減数（引かれる数）に0を伴はないバグ群
3　被減数が　0　からの借りを伴うバグ群
の三つの枝分かれをルート部で持つ階層構造とすると良いことを示した。

一方、Van Lehn(1986)は、代数手続きを修得する学習過程を
1　発見による学習過程
2　例題の観察等からの帰納仮説にもとづく学習過程
3　類似仮説にもとずく学習過程
4　口述仮説にもとずく学習過程
の四通りの仮説にもとづくと仮定して、Buggyモデルの66個のバグを各々検討した結果、これらのバグを合理的に説明できるのは2の帰納仮説のみであることを確認した。観察されたバグの85%を帰納仮説により説明できるとした。発見または口述によりこれらの代数手続きを修得するというのは、全く例外的であり、ほとんどの学習者は、教師などが黒板で例示する幾つもの解答例から帰納的に引き算用の代数手続きを修得し学習を進めることになる。バグの修得もまた帰納的過程によるが、その帰納過程で
1　加重な簡素化（Oversimplification）
2　過重な一般化(Overgeneralization)
3　過重な特殊化(Overspecialization)
4　視覚的数値の対処を許す緩和した帰納仮説
等のためバグが生成されるという理論である。Van Lehn(1986)は、バギーモデルのバグを修復理論の基礎となった'正しい代数手続き'中に組み込め、しかもそれらのバグが観察されたバグの85%にも及ぶことを示した。これは、観察されるほとんどのバグが階層構造に組み込めることを示唆しており、バグ探索・バグ診断上大きな利点となる。

我々は、本論文で 階層構造を持つVan Lehnの66個のバグについて、
1　原始的バグは、常に単独でしか現れない
2　複合バグさらにバグの移行は起きない
と仮定して、ハイパーカード上でバグ診断システムを作成した。

3　　階層構造とバグ診断システム

図1に、我々が構築したバグの枝分かれノードを含む階層図を示す。各ノードには、いくつかのそのノードにふさわしい問題群がしまってあり、その問題の答の正・誤またはノードによってはその答の種類によってどの子供ノードに行くか決定される。丸のマークは、終端ノード（葉ノードと呼ぶ）を示しマークの中の数字は、Van Lehn(1986)によるバグ番号である。Van Lehnの分類はアルファベット順で、そのバグ番号自体は階層構造・格子構造とは関係な

い。例えば、ノードN3の下にある36,38,40,48の番号を持つバグは、N3ノードの問題を解けば、バグはユニークに決まることを示す。

バグの大分類は、修復理論の例により（福田&徳田1989）

① 借りを要しないバグ群（N2,N3に属するバグ）

② 被減算に0を伴わないバグ群(N13,N14,N15,N16,N17,N18,N19に属するバグ群)

③ 被減数に0を伴うバグ群(N7,N9,N10,N11に属するバグ群)

と大きく分類した。

なお、③はさらに

③a 被減数に0が１個のみ現れるバグ群（N10）

③b 被減数に0が２個以上現れるバグ群（N9,N11）

と分岐させることができる。

なお、小分類の方法としては、帰納仮説による学習過程でバグ生成に寄与している

① 視覚的対処を許す緩和した帰納仮説

② 過重な簡素化

③ 過重な一般化

④ 過重な特殊化

によってバグの階層化を進めることもできる。

ここでは、Burton(1982)がとったのとよく似たアドホックな方法、すなわち、'各ノードの問題で出来るだけ多くのバグが判別できる'というガイドラインに従って選んだ。ここで示した階層構造はその構築法から勿論一通りには決まらないし、もっと効率の良い階層構造が存在する可能性もある。

我々が各ノードで使った問題群は表１に示してある。最短で借りを伴わないN2のバグ群30,41,50,52,53,61は２問で、最長で被減数に0を伴うN11のバグ群11,31,39,44は７問でバグの同定ができることが分かる。

4 検討

この論文では、Burton(1982)のDebuggyシステムの簡略化したモデルと考えて良い階層構造を持つ原始的バグが単独でのみ出現し、しかもバグの移行は起きないという限定された条件下でのバグ診断プログラムを構築した。階層構造が構築出来、各ノードに適した問題群を作成出来ればDempster-Shafer理論などの確率的推論計算の手間を掛けなくとも正しくバグ同定ができることが分かった。木構造の深さは一番短いところで２、長いところで７で、６６個もあるバグ探索としては非常に効率が良いシステムと思われる。

ハイパーカードは、図形エディターさらにMacPaintのような図形編集プログラムがインタラクティブに使えるなどこの種の診断システムを作るのに便利な機能も多いだけでなく、ハイパーカード上どこにいてもシステムメッセージをカードのボタンのようないわゆるオブジェクトに送り込むことで実行出

来るなど多岐にわたる機能を使うことが出来る。大変視覚的にも興味深いシステムを作ることが出来た。ただ、ハイパートークでプログラムを作っているので実行速度はかなり遅いという欠点は免れない。

このシステムを実用的にするには、複合バグさらにはバグの移行を考慮する必要がある。今これを実現するため仮説生成・検証機能を取り入れたシステム作りに取り掛かっているところである。トレース機能・グラッフィク機能などに特徴を持つ新しいオブジェクト指向システムADS上で作る予定である。

参考文献

1 Brown ,J.S. & Burton,R.B. (1978) Diagnostic models for procedural bugs in basic mathmatical skills. Cognitive Science 5,pp.379-426

2 Brown,J.S.& Van Lehn,K. (1980) Repair theory. A generative theory of bugs in basic mathematical skills. Cognitive Science 4,pp.379-426

3 Burton,R.B.(1982) Diagnosing bugs in a simple procedural skill. in Sleeman,D & Brown,J.S(Eds) Intelligent Tutoring Systems, Academic Press

4 Gordon,J. & Shortliffe,E.H.(1985) A method for managing evidential reasoning in a hierarchical hypothesis space. Artificial Intelligence 26,pp.324-357

5 福田章夫 & 徳田尚之(1989) バギーモデルと効率の良いエラー同定法 「教育におけるコンピュータ利用の新しい方法」シンポシウム情報処理学会

6 Shafer,G.& Logan,R. (1987) Implementing Dempster's rule for hierarchical evidence. Artificial Intelligence 33,pp.271-298

7 Shaw,D.J.,Standiford,S.N.,Klein,M.F.& Tatuoka,K.K.(1982) Error analysis of fraction arithematics(Tech.Rept 82-2-NIE),University of Illinois,Computer-Based Education Research Laboratory

8 Sleeman,D.H.(1984) Basic algebra revised:A study with 14-year olds. International Journal of Man-Machine Studies

9 Van Lehn,K.(1986) Arithmetic procedures are induced from examples. in Conceptual and Procedural Knowledge:The case of mathematics. Edited by J.Hiebert,Hillsdale,New Jersey

C は正しい場合の分岐
W は間違えた場合の分岐
a, b, c . . . は C, W 内での分岐、及び C, W に分けられない場合の分岐です。

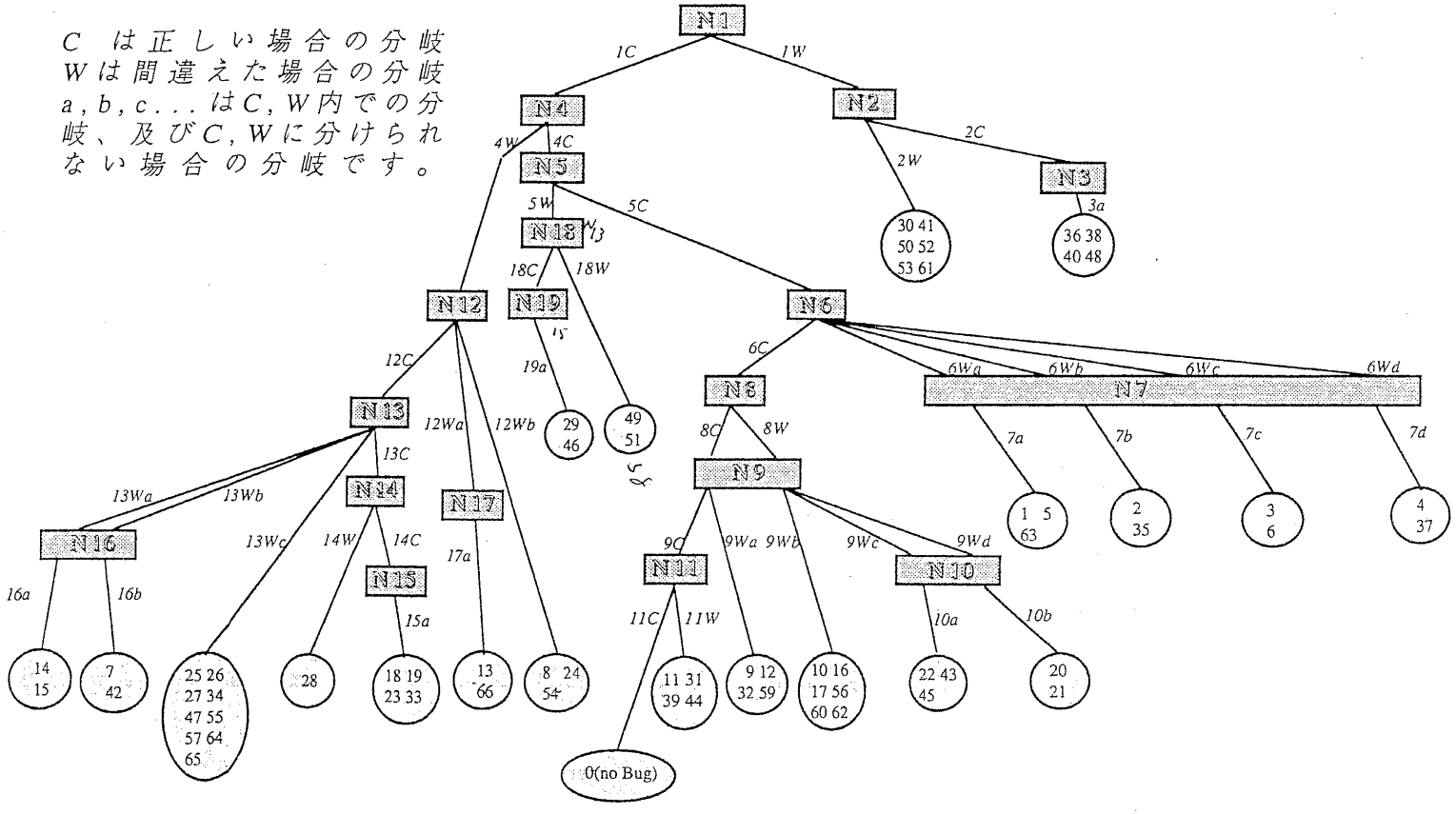

図1　分岐ノードを含むバグ階層図（Van Lehn(1966)によるの６６個のバグ）

表1　各ノードで使う問題群

(N1)	(N2)	(N3)	(N4)	(N5)
1645 40	223 13	20 3	1231 538	1151 1056
1102 80	544 34	1350 229	1851 959	31171 21077
1966 60	336 16	180 65	1661 768	41191 31095
1513 10	689 79	160 28	1461 965	61131 41034
1225 20	432 22	1180 76	1571 873	1181 1083
1373 70	785 75	130 29	1741 848	1121 1029
1448 40	248 38	170 31	1381 487	1131 1038
1827 20	843 33	150 43	1631 933	51141 31047
1214 10	552 42	1880 738	1871 976	1161 1067
1193 90	764 54	1470 362	1351 554	21181 1085

(N6)	(N7)	(N8)	(N9)	(N10)
2510 317	115 49	203 19	3004 207	105 8
1610 814	218 99	306 28	2003 109	107 8
310 17	1513 247	408 69	5001 1805	104 7
510 16	2311 166	1605 938	4004 2906	2108 9
1610 418	1214 838	2504 1936	8002 3603	1034 53
110 13	316 78	507 48	6005 408	3107 9
210 18	1415 258	803 27	7008 2709	103 5
1410 216	2217 167	706 49	4006 907	106 9
2810 917	418 79	606 37	5003 808	1028 96
1310 715	113 65	405 57	3001 1903	1076 88

(N11)	(N12)	(N13)	(N14)	(N15)
4002 2	355 183	27453 8545	546 48	716 48
7000 6	387 195	26872 7963	362 67	4136 97
5000 4	33 14	25762 6855	453 59	5122 83
10004 52	361 171	24523 5619	865 68	6127 98
20005 81	3578 1934	28633 9724	732 35	7138 49
2000 9	32 19	26332 7429	562 68	211 9
10009 36	36 18	23753 4848	345 47	3125 86
3000 7	346 152	28536 9627	843 49	4111 72
10003 71	322 191	24425 5518	756 58	5116 87
40006 82	3246 1824	23632 4824	684 86	6127 39

(N16)	(N17)
5132 239	50 26
4346 448	750 327
8760 869	370 135
2524 725	407 82
3763 964	600 349
6243 349	20 13
5457 559	430 218
7657 758	250 137
3635 836	306 93
4874 975	509 238