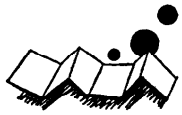
解　説

エキスパートシステムにおける知識獲得†

渡　辺　正　信††



1. はじめに

知識ベースと推論エンジンから構成されるエキスパートシステムのパフォーマンス（機能や速度）は，知識ベースの完成度で決まる．つまり，エキスパートシステムの問題解決能力は，対象ドメイン（領域）の専門知識を，知識ベースの中に十分に保持しているかどうかで決定される．しかし，対象ドメインの専門知識が明確でないこと，専門知識が時間と共に変化することなどにより，完全な知識ベースを最初から構築するのは非常に困難である．逆に，これが非常に困難であるから知識工学的アプローチを取ったわけである．最初から完全な知識ベースを構築できるのであれば知識工学的アプローチを取る必要はない．したがって，エキスパートシステムの開発・発展において，段階的に知識ベースを充実させていかなければならない．ここに知識獲得の問題が発生する．

エキスパートシステムにおける知識獲得とは，エキスパートシステムの開発・改良・拡張において，対象ドメインの知識を抽出・分析すること，次に実行可能でかつ汎用な形式にその知識を変換すること，さらに，獲得された知識と既存の知識ベースとの一貫性を保持・管理することを言う．この知識獲得作業は，通常，特定ドメインの専門家と知識工学者との共同作業の形をとり，多大な時間を要する非常に困難な手作業的色彩の強いものである．この作業は，たとえ現在のエキスパートシステム構築ツール（AI シェル）を利用したとしても大幅に工数を削減できるものではない．むしろ，知識表現や推論機構を提供してくれる AI シェルの普及にともない，知識獲得の工数比の増大が顕在化し，重要さが強く認識されるに至った観がある．必然的に，知識獲得の半自動化，または，自動化を目指す研究開発が活発化し，知識工学における最大のボトルネックであるという認識が定着してきている．

本稿では，エキスパートシステムにおける知識獲得の現状と今後の方向について解説する．以下，第2章において，知識獲得の問題とアプローチ例を示す．そこでは，エキスパートシステム開発フェーズからみた問題と，知識を具体化していく知識獲得段階での基本問題（特に，知識の抽出・変換・管理という基本問題）を整理し，各問題に対する代表的アプローチ例を紹介し現状の技術レベルをみる．第3章において，知識獲得での現状の課題と最近の研究動向を踏まえて，知識獲得の今後の方向を述べる．

† Knowledge Acquisition for Expert Systems by Masanobu WATANABE (Computer System Research Laboratory, C&C Systems Research Laboratories, NEC Corporation).

†† 日本電気(株) C&C システム研究所コンピュータシステム研究部

2. 知識獲得の問題とアプローチ例

ここでは，エキスパートシステムにおける知識獲得の問題を明確化し，それに対する代表的アプローチ例を紹介する．最初に，エキスパートシステム開発における知識ベースの初期化と詳細化という大局的観点でこの問題を整理し，実例を大別する．次に，知識獲得段階を，問題の確認・概念化・定式化・実現・テストの5つに分けて各段階での基本問題とアプローチ例を明らかにする．その後，知識の抽出・変換・管理という三つの基本問題と具体例に関して詳述する．

2.1 エキスパートシステム開発フェーズからみて

エキスパートシステムの開発では，知識ベースを段階的に詳細化するため，デモシステム，プロトタイプシステム，実用システムを逐次的に開発していくことが必要である．これらおのおののシステムを開発する上での知識獲得の問題は，図-1 に示したように知識ベースの初期化フェーズと，詳細化フェーズに分けることができる．以下，おのおののフェーズでの問題とアプローチ例を紹介する．

2.1.1 初期化フェーズ

知識ベースの初期化とは，専門家からラフな知識ベースを抽出してエキスパートシステムの原型を形成することである．ラフな知識ベースという意味は，この知識ベースを使った性能が専門家のそれに到達していないということである．例えば，初期化されたデモシステムでは，限定された例題に対する解でも専門家のそれには必ずしも及ばない．その詳細化により，デモシステムは限定された範囲で専門家の域に達するこ

<table>
<tr><td rowspan="2">知識獲得問題</td><td colspan="4">初期化フェーズでの問題</td><td colspan="2">詳細化フェーズでの問題</td></tr>
<tr><td rowspan="2">知識獲得段階</td><td rowspan="2">問題の確認</td><td rowspan="2">概念化</td><td rowspan="2">定式化</td><td rowspan="2">実現</td><td colspan="2">テスト</td></tr>
<tr><td>知識構造のテスト
(静的テスト)</td><td>推論過程のテスト
(動的テスト)</td></tr>
<tr><td rowspan="3">知識獲得システム</td><td rowspan="3" colspan="3"></td><td colspan="2">ETS</td><td>TEIRESIAS
EMYCIN
ONCOCIN
TMS/ATMS</td><td>SEEK/SEEK2
LEAP
LAS
VILLA
CONSTELLATION
ACES</td></tr>
<tr><td colspan="2">MORE
SALT
ROGET</td><td colspan="2"></td></tr>
<tr><td colspan="4">MOLE</td><td colspan="2"></td></tr>
</table>

図-1 知識獲得問題と知識獲得システム

とになる．なお，初期化フェーズは，2.2 で詳述する4つの知識獲得段階に分かれる（図-1 参照）．この初期化フェーズでの問題点は，エキスパートシステムでの知識表現や推論方式を習熟していない特定ドメインの専門家の知識を整理し，エキスパートシステムのヒナ型をいかに効果的効率的に形成できるかという点にある．この問題に対するアプローチ例として，ETS[1]，MORE[2]，SALT[3]，ROGET[4]，MOLE[5] などがある．

ETS は，専門家に対する系統的なインタビュ方法（特定ドメインに独立な方法）を使って初期のルールベースを自動生成する．系統的なインタビュとは，分類問題で結論となる項目（通常は複数）を最初に質問し，その後，これらの項目間の差とか類似性を示すパラメータを抽出し，その重みづけなどを行っていくものである（2.3.1 参照）．MORE，SALT，ROGET，MOLE などは，専門家へのインタビュにより，まず，対象ドメイン（MORE，ROGET，MOLE は診断型問題，SALT は設計型問題）に対するモデル（ドメインモデルと呼ぶ）を生成する．その後，そのドメインモデルから実行可能な形式のルールを生成して，ルールベース（知識ベース）を初期化する（2.3.2 参照）．この他，知識ベースの初期化を行うものとして，因果関係モデルから診断ルールを生成する知識コンパイラ[6]や，データベース中のデータや専門家が入力したデータから，知識抽出用ルールを使って診断ルールを生成するもの[7]などがある．いずれも，初期化フェーズにおいて，専門家に代わって初期のルール集合を作成することを目指す．

2.1.2 詳細化フェーズ

既存の知識ベースを段階的に改良，拡張してエキスパートシステムの性能を向上させることを知識ベースの詳細化という．詳細化フェーズは，知識構造を評価・改良する静的テストと推論過程を補強する動的テストからなる．ここでの問題には，新たに追加しようとする単位知識（ルールやフレームなど）と既存の知識ベースとをいかに整合させるかとか，知識ベース全体の一貫性をいかに保持するかとか，問題解決能力を向上させる上での知識ベースの弱点をいかに抽出し補強するかという問題がある．換言すると，知識構造を詳細化する問題と推論過程を詳細化する問題に分かれる．

まず，知識構造の詳細化問題に対するアプローチ例としては，TEIRESIAS[8]，EMYCIN[9]，ONCOCIN[10]，TMS[11]，ATMS[12]，MORE，SALT，ROGET，MOLE などがある．TEIRESIAS，EMYCIN は，ルールモデルを使って，専門家が入力したルールの条件部に不足しているパラメータなどを補足することを対話的に支援する機能を持つ．ここでルールモデルとは，あるパラメータを結論づけるルールの条件部に使われるパラメータ集合のテンプレートであり，既存の知識ベースのルール集合から生成される．また，ルールモデルは新しいルールの入力により必要に応じて更新される．ONCOCIN は，ルールの条件部に記述されたパラメータの特定の値に対して結論づけるルールの欠如（ミッシングルール）を抽出し，知識ベースの完全化を図る．TMS，ATMS は，推論途中で知識が変更されるごとに知識ベースの論理的一貫性を動的に管理維持する方式を提案している．TMS/ATMS では，特に，知識獲得に関して言及していないが，獲得された知識と既存の知識ベースの管理・検証という意味で重要な役割を果たす．MORE，SALT，ROGET，MOLE は，知識ベースの静的な構造を解析してその弱点（例えば，徴候が明示されていない原因仮説など）を引き出し，専

門家へ知識ベースの詳細化を促す．

推論過程の詳細化問題に対するアプローチ例としては，MOLE, SEEK[13], SEEK2[14], LEAP[15], LAS[16], VILLA[17], CONSTELLATION[18], ACES[19] などがある．MOLE は，自分の誤診例とそれに対する専門家の正しい診断例を比較し，動的な診断プロセスにおけるそれらのギャップを対話的に分析し埋めていき知識ベースのバグの補正を支援する．MOLE が，一つの誤診例を使ったのに対して，SEEK では，複数の診断例（ただし，同じ結論に至るべき正解例と誤診例）を基に利用されたルール集合の中からバグルールを統計的分析によって自動的に局所化する(**2.3.1** 参照)．さらに，SEEK2 では，バグルールの修正を一部自動化した．一方，LEAP, LAS, VILLA, CONSTELLATION では，専門家の問題解決ステップをモニタすることにより新しいルールを獲得する(**2.3.2** 参照)．このモニタ方式は，知識ベースの初期化に利用可能であるが，既存のルールを専門家が否定した場合などに，新たな方式(ルール)を獲得して知識ベースを拡充することに力点がおかれている．ACES は，診断での原因仮説の誤りをモニタ部が自動的に検出し，なぜ自分が誤ったかをデバイスモデル（一種の因果関係モデル）を基に判断し，その誤った仮説を生成するルールの条件部を補正する．

次に，初期化と詳細化フェーズをもう少し細かな段階に分けてそこでの知識獲得の基本問題とアプローチ例を明らかにしていく．

2.2 知識獲得段階からみて

知識獲得段階とは，デモシステムから実用システムのそれぞれを開発していく上での基本段階ということで，図-1 と図-2 に示した5つの段階（問題の確認，概念化，定式化，実現，テスト）からなる[20]．以下，各段階での基本問題を示す．なお，図-2 に示したように，知識ベースの初期化フェーズは，問題の確認段階から実現段階の問題として，詳細化フェーズは，テスト段階及びそこからのフィードバックの問題（つまり，初期化フェーズの問題を含む）としてとらえることができる．

2.2.1 問題の確認段階

この段階は，対象とする問題の入出力を明確化し，従来のアプローチではなく知識工学的アプローチをなぜとるべきかを判断する段階である．特に，問題解決に重要な知識を誰が保持するのか，その人々（専門家）の協力を得ることが可能かなどの確認が重要となる．つまり，ここでの問題は，上記の判断基準は何か，確認事項は何かを明確化するということにある．例えば，「常識的知識が大きく関与しないような狭い専門領域に対象を絞る」といった格言の提示により判断基準の明確化が試みられている[20]．しかし，この段階を支援するシステムの提案には至っていない．

2.2.2 概念化段階

概念化とは，問題を明らかにする上で重要な概念や概念間の関係を明確化することである．特に，ドメインの専門家による問題解決プロセス／推論プロセスを明示化する専門家モデルの同定と，エキスパートシステム利用時のマンマシンインタフェースを構築する上でのユーザモデルの同定が問題となる．例えば，専門家モデルにおける概念をドメイン方程式として明確化するツール SUPE-SPOONS[21]や，具体的マンマシンインタフェースをベースにユーザモデルを同定していきエキスパートシステムのラピッドプロトタイピングを支援するツール[22]が提案されている．

2.2.3 定式化段階

この段階では，概念化の段階で明確化されたキー概念，概念間関係，部分問題などがエキスパートシステム構築ツールの枠組によって形式的表現に表される．こ

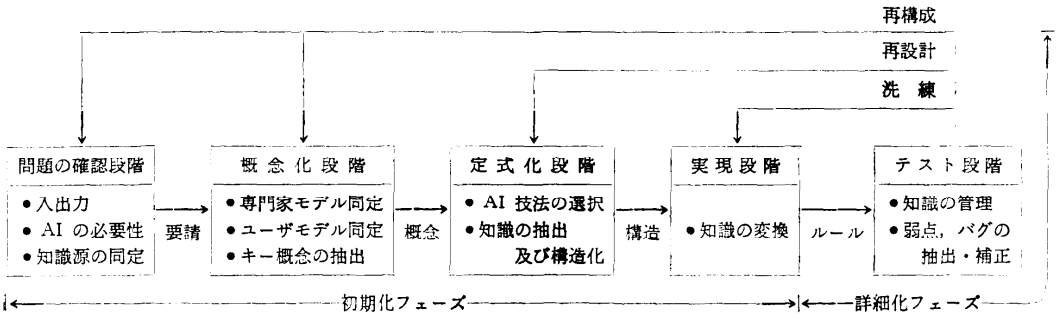


図-2 知識獲得段階での基本問題

こでの問題点は，モデルの性質に応じて最適なAI技法(知識表現や推論方式)を選択する手順と，選択された知識表現の枠組内で知識を抽出し，最適な構造化を行う方法にある．文献23)は，対象問題／モデルの性質に応じてどのようなAI技法を選択すべきかの5つの基準を示している．例えば，**図-3**は，問題に出現する変数の値の決定手順の性質により，制約伝播手法と前向き推論ルールの利用のどちらを選択すべきかの基準を示している[23]．MORE/MOLEなどは，診断問題に固有の知識を，まず徴候や原因仮説をノードとするネットワーク表現として抽出し，そのネットワークを段階的に詳細化，構造化することを支援する(図-1参照)．なお，知識の抽出については，**2.3.1**で詳述する．

2.2.4 実現段階

ここでの基本問題は，前段階で定式化された知識を，採用したツールの知識表現の枠組に従ってコンピュータ上で推論実行可能な形式にいかにして変換するかということである．例えば，ETS, MOREなどは，ネットワークで定式化された知識を推論実行用のルールに自動または半自動変換する．なお，知識の変換に関しては，**2.3.2**で詳述する．

2.2.5 テスト段階

この段階では，知識ベースの構造的整合性テスト(静的テスト)と，推論過程のテスト(動的テスト)が行われる．つまり，知識ベースの詳細化問題が発生する．特に，知識の管理と，知識ベースの弱点抽出・補正といった基本問題がある(図-1, 図-2参照)．なお，知識の抽出に関しては**2.3.1**，知識の管理に関しては**2.3.3**で詳述する．

2.3 知識の抽出・変換・管理

知識獲得での主要な三つの基本問題について述べる．それは，①エキスパートシステムの能力を向上させるために必要な知識を抽出する問題と，②抽出された知識を推論実行可能な表現形式に変換する問題と，③入力した知識と既存の知識ベースの論理的一貫性を管理する問題の三つである．

2.3.1 知識の抽出

知識の抽出という基本問題は，初期化フェーズでの定式化段階と，詳細化フェーズでのテスト段階で発生する．その抽出形態は，環境からの知識の取り込み方法とエキスパートシステムでの推論実行結果からのフィードバックの方法により，次のようなレベルに分類される．そこでは，エキスパートシステムの性能を向上する上で必要な知識をいかに効果的に効率良く抽出するかが問題となる．

(1) 人間のアドバイスによる抽出

人間が新しい知識(例えば，ルールなど)を意図的に(直接または間接的に)与える形態である．システムは必要に応じて，その新しい知識を知識ベースの表現形式に変換する．ここでの問題は，人間にとって最適なアドバイスの形態(知識を提供する形態)は何かということである．例えば，推論実行可能な形式のルールを人間が直接与えるものとしてTEIRESIASやEMYCINの方式がある．また，ルールを直接与えるのではなく，ルールを生成するのに必要な情報を系統立てて質問するものとしてETS, MOREなどがある．そこでは，入手した情報をルールに変換するルール生成部を持つ．

図-4は，ETSを使ってデータベースシステム(CREATABASE, EASYTRIEVE, SIR, ADABASなど)の選択に関するコンサルテーションシステムを構築するためのインタビュ例の一部を示す[1]．そこでは，まず，結論となる項目で注目するものを3個(CREATABASEなど)質問し，次に，三つの項目で二つには共通で，他の一つにはない特性(ここでは，TEXT RETRIEVAL)を求める，さらに，この特性に対する反対の特性(ここでは，NOT TEXT RETRIEVAL)を確認する．その後，三つの項目に対してその特性からみた重みづけを行う．ETSは，この特性と反対の特性ペア(Constructと呼ぶ)を基に分類に関する知識を整理する方式を，心理学でのPersonal Construct理論を拡張(Constructに対する重みづけを付加することなど)して実現した．

(2) 例による抽出

あるクラスに属する例や反例を入手し，そのクラスを特徴づける記述(ルールなど)を，汎化／特殊化過

数値またはブール値をとる多数の変数の値を見つけることが必要ですか？
and
既知の変数値を使って他の変数の値を決定することができる制約条件が変数間に存在しますか？
and
値が最初に与えられる変数は問題ごとに異なりますか？
Yes, 問題ごとに異なります → 制約伝播
No, 最初に同じ変数に対して値が与えられる → 前向き推論ルール

図-3 最適なAI技法の決定例

```
What is the name of a database you'd like to consider?
ETS**CREATABASE
What is the name of another database to consider?
ETS**EASYTRIEVE
What is the name of a third database to consider?
ETS**SIR
Think of an important characteristic that two of CREATABASE, EASYTRIEVE, and SIR share, but that
the other one does not. What is that characteristic?
ETS**TEXT RETRIEVAL
What is that characteristic's opposite as it applies in this case?
ETS**NOT TEXT RETRIEVAL
Please rate these things on a scale of 5 to 1, where 5 means more like
TEXT RETRIEVAL and 1 means more like NOT TEXT RETRIEVAL.
      (CREATABASE)**5
      (EASYTRIEVE)**5
      (SIR)**1
What is the name of another database which you feel is different from SIR in some important attribute?
ETS**ADABAS
What is that attribute?
ETS**INVERTED
.....
```

図-4 ETS でのインタビュ例

程を通して形成する．ここで，複数の例を入手して汎化／特殊化を行う帰納的学習方式[24]と，単一例のみを入手して汎化／特殊化を行う演繹的学習方式[25),26)]がある．ここでの問題は，有効な例をどのような時点でどのような形式で抽出できるかということにある．例えば，LEAP, VILLA などは，人間の設計過程をモニタし，人間が示した新しい解法例を抽出した後，その解法を一般化して推論実行用の汎用ルールに変換する．

（3） 問題解決を通した抽出

エキスパートシステムが問題解決をする中で，以後の問題解決に有効と考えられる新しい知識（ヒューリスティックルールなど）を抽出する．ポイントは，問題解決に有効な知識をどのようにして抽出するかにある．問題生成，解決，一般化機構をシステム内に保持するもの[27]，探索を行って解いた部分問題での前提条件と解をルールとして保存し（これをチャンキングと呼ぶ），以後このルールを利用することで，探索を削減し問題解決の効率向上を目指すもの[28]，複数の問題解決例を使ってルールのバグを局所化[13]し，さらに，バグを補正するもの[14]，人間が示した解決法をモニタし必要な知識を抽出するもの[15)-18)]，システム自身が推論結果の誤りを検出し，その原因となったバグルールを自動修正するもの[19]などがある．ここでは，問題解決を通して有効な知識を抽出する SEEK での方式例を紹介する．

TEIRESIAS が，一つの診断例を基にルール連鎖上の虫取りを，専門家との対話を通して支援したのに対して，SEEK では，複数の診断例（ただし，同じ結論に至るもの）を基に，利用されたルール集合の中から，バグルールを統計的分析によって自動的に局所化する．

まず，一つの誤診のケースを考える．次に，誤診を導いたルール連鎖を考えて，各ルールを汎化または特殊化することにより誤診を避けられるかどうかを検査する．ここで，ルールの汎化とは，例えば，ルールの条件部のあるパラメータ項を削除することであり，特殊化とは，パラメータ項を追加することである．この検査結果は，各ルールのポイントとして加算される．すなわち，あるルールの汎化により誤診が避けられるとすれば，そのルールの汎化に関するポイントがプラス 1 される．この手続きを他の誤診のケースについても実行し，各ルールのポイントを加算する．図-5 は，この検査結果を示したものである[13]．ここで，Certainty は確信度で三つのレベル (Definite, Probable, Possible) に分かれている．汎化／特殊化の検討では，Certainty はできるだけ直接取り扱わないようにしている．この表からルール 56 を汎化することによって

Mixed connective tissue disease

Rule	Certainty	Generalization	Specialization
54	Possible	2	0
55	Possible	7	0
56	Possible	8	0
57	Probable	2	0
58	Probable	2	0

図-5 MCTD ルール (Mixed connective tissue disease を結論づけるルール) 検査サマリ

Rule 56:
3 or more Majors for MCTD
→Possible mixed connective tissue disease

Generalization of Rule 56:
2 or more Majors for MCTD
→Possible mixed connective tissue disease

ここで，Majors とは，Major criteria のこと

Major criteria	Minor criteria
1. Swollen hands	1. Myositis, mild
2. Scterodactyly	2. Anemia
3. Raynaud's phenomenon or esophageal hypomotility	3. Pericarditis
4. Myositis, severe	4. Arthritis < = 6 wks
5. CO diff capacity, nl : < 70	5. Pleuritis
	6. Alopecia

図-6 ルール56の修正（汎化）例

	Before	False positives	After	False positives
MCTD	9/ 33(27%)	0	17/ 33(52%)	0
Others	80/ 88(91%)		80/ 88(91%)	
Total	89/121(74%)		97/121(80%)	
Others				
RA	42/ 42(100%)	9	42/ 42(100%)	8
SLE	12/ 18(67%)	4	12/ 18(67%)	3
PSS	22/ 23(96%)	5	22/ 23(96%)	3
PM	4/ 5(80%)	1	4/ 5(80%)	1

図-7 ルール56の修正前と修正後の性能

誤診を避けられるケースが8あって最高ポイントとなっていることがわかる．ここまでは，SEEK が自動的に行う．この後，専門家がルール 56 を汎化するとする．**図-6** は，ルール56の汎化例である．ここでは，もとの条件では，MCTD に対する Major Criteria が3個以上となっていたのを，1個減らして2個以上に修正している．この修正によっていかに性能向上したかを**図-7** で示している．すなわち，MCTD に対する33のケースのうち修正前は，9ケースしか正しく診断されなかったものが，修正後，17のケースについて正しく診断されるようになったわけである．さらに，この副作用として他のケースへの悪影響がなかったと同時に，誤診 (False positives : 例えば，RA でないものを RA と診断すること) をも減らすこととなった．注目すべきことは，ルールの条件部のみの変更（汎化または特殊化）によりシステムの性能向上が望めたことと，修正すべきルールを複数の例を使って自動的に局所化したことである．

2.3.2 知識の変換

知識の変換とは，知識抽出により自然言語などで定式化された知識をコンピュータ上で推論実行可能な形式に表現することである．なお，知識の変換という基本問題は，初期化フェーズでの実現段階と，詳細化フェーズでのテスト段階でおこる．特定の AI シェルを利用してエキスパートシステムを構築する場合は，そのシェルで推論実行できる表現形式に表すことになる．つまり，ルールベースで推論実行する場合は，抽出された知識をその AI シェルのルール言語で正確に表現しなければならない．このとき，専門家から抽出した知識の形式によって次の3種類の変換方式がある．

（1） ルール形式からルールへの変換

プロダクションルール形式で専門家から抽出し，そのシンタックス／セマンティックスのチェックを行い，実行可能なルールに変換する．ここでの問題は，入力されたルール形式を内部形式に正確に変換することだけでなく，専門家の意図が十分に反映されたルール形式であるかを確認することにある．例えば，TEIRESIAS などでは，入力されたルールを自然言語の説明文に変換して，ルールが意味するものを専門家に確認させる．これはルール言語による記述が必ずしも専門家にとってわかりやすいものでないため必要となる．今後，自然言語や図形を使ってルール形式を入力できるようになったとき，この問題が重要となる．

（2） モデル形式からルールへの変換

ルールベースシステムにおいて，定式化段階で抽出された知識構造がルール形式ではなく，別の表現形式（例えば，ネットワークとかフレーム表現）の場合は，この表現形式（モデル形式）からルールに変換することが必要になる．ここでのポイントは，ルールを生成するために必要な情報を，どのような表現形式を基に専門家から抽出するのが最適であるかということと，いかにルール形式に変換するかということにある．ETS, MORE, SALT, MOLE などは，モデル形式の情報から実行可能形式のルールを自動または半自動生成する．ただし，モデル形式は，対象とする問題が分類型，診断型，設計型により異なる．例えば，診断型に対するモデルは，徴候とか故障原因仮説をノード，それらの因果関係をリンクとするネットワークで表現される．つまり，専門家にとっては，直接ルールを入力するより，このような因果関係ネットワークを形成することが容易であろうと考えるわけである．

ここで，MORE の例を使って，対象ドメイン（診断問題）のモデル化とモデルの詳細化，及び，モデルからルールの生成について紹介する．**図-8** は，MORE でのネットワーク表現されたドメインモデルを示している．ここで，ノードは原因仮説(H)と徴候(S)を示し，ノードの間の実線はノード間の因果関係を示すリ

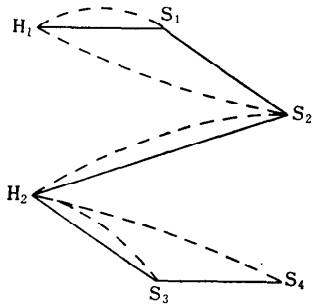


図-8 MORE でのドメインモデル例

IF S1 と S2があれば <徴候>
THEN H1 を(4)で信じる <原因仮説>

図-9 MORE で生成されたルール

ンクである．なお，ノード間の点線はパスと呼ばれ，原因仮説(H)と徴候(S)を直接関係づける特殊なリンクである．MORE は，まず，ユーザ（ドメインの専門家）が，このドメインモデルを構築することを対話的に支援する．特に，ネットワークの形状を解析することでドメインモデルのバグの修正を支援する．例えば，原因仮説(H)にリンクされた徴候(S)が一つもないときは，このことをユーザに知らせ，その徴候の入力を要求する．このような詳細化支援戦略を8つ持つ．ドメインモデルが作成された後，MORE は，診断ルールを専門家の支援の下で作成する．**図-9** は，図-8 で示したドメインモデルから生成されるルールの例である．ここで，原因仮説の中での(4)は，この H1 を信じる度合を示すサポート値であり，専門家により入力されたものである．また，徴候間に関する生起依存性がある場合，ルールを生成する方法が複雑になる．例えば，S1 と S3 は同時に起こり得ないとわかっていれば，この二つをルールの条件部に記述する必要がない．この情報は，ドメインモデルのネットワークに陽に記述されないのでルール生成時に注意が必要となる．なお，MORE の発展システム MOLE では，サポート値をシステムが診断経験を通して自動修正する．

特に，機械系のシステムなどのように故障原因の因果関係が比較的明確に記述できる場合は，専門家に診断ルールを直接求めるよりも，因果関係モデルから一部の診断ルールを自動生成することが可能である．専門家にとっても知識ベースの初期化には，この方式が望まれるであろう．この因果関係モデルからの診断ルールの生成は，深い知識をコンパイルして浅い知識（ルールに対応する）を生成すると呼ばれている[6]．つまり，診断を実行する場合は，深い知識（因果関係モデル）を使うより浅い知識（コンパイルされた知識）を使って診断する方が効率がよくなる．

(3) 例からルールへの変換

具体的問題（状態）とその解が例として与えられた場合，その問題パターンを条件部に，解を帰結部に持つルールを生成すれば，以後の問題解決に利用できる．ここでのポイントは，そのルールをできるだけ一般化して，その適用範囲を拡げておくことである．つまり，例からルールへの変換では，一般化手法が重要となる．従来は，この一般化は，複数の例に基づく帰納的手法が主流であったが，近年，単一例から一般化する演繹的手法(説明ベース学習とも呼ばれる)が注目されている．**図-10**, **図-11** は，VILLA で演繹的手法を使って例から汎用的ルールを生成したものを示す．そこでは，図-10 で示した設計の要求仕様 (Function to be implemented～Where Input Signals Satisfy) とそれに対する設計者の回路合成例 (User's Solution) を例として，図-11 の汎用的ルールを生成している．ここで，図-11 の Function 部における〈bool-fn1〉と〈bool-fn2〉は任意のブール関数，〈any1〉と〈any2〉は任意の値を示す変数である．つまり，図-10 の Function の特殊な部分が変数に置き換えられることによって一般化されたわけである．その一般化手法は，まず，ユーザの解（例）がもとの要求仕様をなぜ満足するかをドメインの知識（ここでは，ブール式の変換定理など）を使って検証（説明）する．この検証ステップでは，与えられた例の中で何が重要な項目であるかが抽出される．次の一般化ステップでは，検証に利用されたドメイン知識の前提条件を基に重要な項目を一般化する．

2.3.3 知識の管理

知識ベース管理とは，知識ベースの一貫性・完全性を保持することである．特に，新しい知識が獲得されて知識ベースに対する更新が発生したときに知識ベースの管理が問題となる．この知識ベース管理の問題は，詳細化フェーズでのテスト段階で発生するわけであるが，静的な管理と動的な管理の二つの問題に分けることができる．

静的な管理の問題とは，知識ベースの静的な状態(知識構造)に対する一貫性・完全性をいかに保持する

かということである．例えば，ONCOCINでは，所与のルール集合でカバーできてない条件部の値に関するルールの欠如（ミッシングルール）を指摘することでルールベースの完全性を保持することを支援する．ただし，この支援は，推論実行中に行われるものでない．

一方，動的な管理の問題とは，推論実行中に知識ベースの更新が発生するごとに，知識ベースの一貫性をいかにして保持できるかという問題である．例えば，TMS／ATMSは，非単調推論をベースに知識ベースの論理的一貫性を動的に管理する機能を提供する．そこでは，Aという事実の成立が，Bという事実の成立でのみ保証されている場合，Bが偽になると自動的にAを偽とする．つまり，問題解決部では，必要時にBを偽とすることだけを指示すれば，Aも偽とするような指示を与える必要はない．TMS／ATMSは問題解決や推論を実行するシステムというより，事実間の支持ネットワークを使って知識ベースの一貫性を保持するシステムである．また，DIFF[29]では，問題解決部で実行されるルール自身にAS-LONG-AS部を付加することによりTMSと同様な機能を持たせる方式を提案している．

以上述べたように知識の管理という問題は，獲得された知識を知識ベース側に完全に吸収し，消化させるときに発生する問題である．

3. 現状の課題と今後の方向

ここでは，前述した知識獲得での三つの基本問題（知識の抽出，変換，管理）をベースに，現状の課題を整理し，知識獲得システムの今後の方向を示す．

3.1 知識の抽出

ここでの問題は，エキスパートシステム構築における初期化フェーズと詳細化フェーズにおいて，システムのパフォーマンスを向上させるための知識を，いつ，どのようにして，どんな形式で抽出するのかということであった．初期化フェーズ及び詳細化フェーズの静的テスト段階でのこの問題に対しては，まず，TEIRESIASにおいて，専門家から推論用ルールを直接入手する場合の支援方式が提案された．その後，ETS, MOREなどが，推論用ルールに替わって，ルー

Function to be implemented:
Inputs: Input1, Input2, Input3
Outputs: Output
Function: (Equals (Value Output (i))
(If (And (Equals (Value Input1 (i)) High)
(Not (Equals (Value Input2 (i))
(Value Input3 (i)))))
Then Low
Else Tristated))

Where Input Signals Satisfy:
(Equals (Voltagelevel Input1 (i)) Switchable-High-Level)
(Equals (Voltagelevel Input2 (i)) Restoring-Logic-Level)
(Equals (Voltagelevel Input3 (i)) Restoring-Logic-Level)

User's Solution:

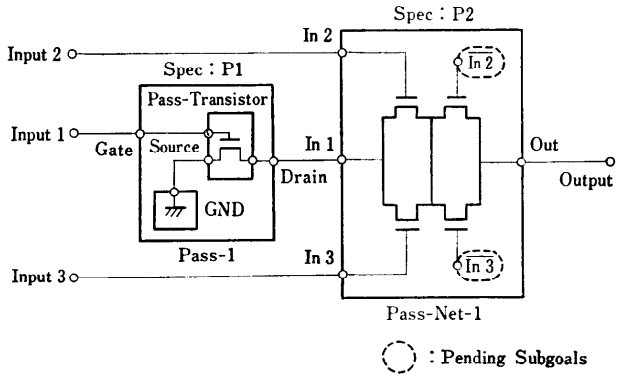
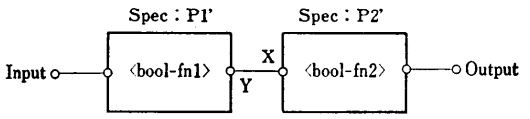


図-10 VILLAに与えられた例

If the function to be implemented is of the form:
Inputs: *any*
Outputs: Output
Function: (Equals (Value Output (i))
(If (And ⟨bool-fn1⟩ ⟨bool-fn2⟩)
Then ⟨any1⟩ Else ⟨any2⟩))

Where Input Signals Satisfy:
(Equals (Voltagelevel *any*)
(If (Equals (Voltagelevel ⟨any1⟩)
(Voltagelevel ⟨any2⟩)
Passing-Low-Level)
Then Switchable-High-Level
Else Restoring-Logic-Level))

THEN one possible implementation is:



With Specifications of the two modules as follows:
P1': (Equals (Value Y (i))
(If ⟨bool-fn1⟩ Then ⟨any1⟩ Else ⟨any2⟩))
P2': (Equals (Value Output (i))
(If ⟨bool-fn2⟩ Then (Value X (i)) Else ⟨any2⟩))

図-11 VILLAで生成されたルール

ルを生成するために必要な情報を，ドメインモデルに従って専門家から入手する方法を示した．後者では，専門家にとって親しみにくい推論実行用ルールを直接入手するよりは，もっと専門家にとってわかりやすい形式で知識を入手する方が望ましいと考えたからである．しかし，このためには，ドメインの問題構造が分析されてそこでの概念モデルが確立されていなければならない．

現状では，例えば，ETS は分類型問題，MORE/MOLE は診断型問題というように限定された一部の問題（ドメイン）に対してのみそのドメインモデルが明らかになってきたにすぎない．今後は，その他の種種の問題（例えば，設計，計画，分析など）に対するドメインモデルの確立が必要である．同時に，ドメインの専門家から，ドメインモデルを形成するために必要な情報を，どのようにして得るかが重要な課題である．この観点から，自然言語テキストからドメインモデルを生成する知識獲得方式（例えば文献 30）など）が注目される．さらに，ドメイン固有の推論制御用知識（ヒューリスティックス）を抽出するためのドメインモデルの構築が必要となる．

詳細化フェーズの動的テストでの知識抽出の問題に対しては，①推論実行中にいかにして知識ベースのバグまたは弱点を抽出するかという点と，②一度解いた部分問題の解を利用して以後の問題解決の効率化をいかに図るかという点と，③人間との対話的問題解決過程において人間の持つ新しい知識をいかに取り込む仕掛を用意するかという三つの重要な課題がある．第1点に対しては，ACES で提案されているような自分自身の推論実行での失敗をモニタしてバグや弱点を抽出する方法，第2点に対しては R1-SOAR[28] で見るように一度解いた部分問題をチャンクとしてキャシュし，以後このチャンクを利用して問題解決の効率化を図る方法，第3点に対しては，LEAP, VILLA などのように，問題解決ステップに対するエディタを人間に提供し，人間の問題解決プロセスをモニタし，有効な知識を抽出する方法などがそれぞれの課題に対する今後の方向を示唆している．

3.2 知識の変換

まず，ルール形式からルールへの変換においては，変換されたルールの検証が課題となる．例えば，複雑なパターンマッチング記述が専門家の意図を正確に反映しているかの確認などは，大きな問題となってくる．

次に，ドメインモデル形式からルールへの変換においては，そのドメインモデルを使って必要十分なルールが完全に生成できるかという課題がある．簡単なルールでもこの変換過程または変換後に人間の助けを必要とするのが現状である．この問題に対する今後の方向は，ドメインモデルの緻密化にある．この意味で定性的な因果関係モデルを使った推論方式の確立が必要となる．

最後に，例からルールへの変換での課題は，一般化の問題である．特に，単一例から一般化を行う説明ベース学習方式の発展及び，その手法と従来の帰納的学習方式のそれぞれの長所を生かす統合化方式の確立などが重要となる．

3.3 知識の管理

ここでは，効率の良い動的な管理方式の確立が大きな課題である．例えば，TMS/ATMS では，動的管理における無駄なバックトラックをいかになくすか，最小とするかを目指して検討されてきた．今後は，さらに，知識ベースの動的な管理に要する処理を低減し，問題解決プロセスでの負荷をいかに少なくするかが大きな問題となる．つまり，知識ベース全体の完全な一貫性を常に保持するのは非常なオーバヘッドになるわけで，問題解決に必要最小限の管理を行うのが望まれる．逆に，問題解決に有効となるような知識ベースの一貫性を管理すれば十分である．

以上，知識獲得での三つの基本問題ごとに現状の課題と今後の方向を示唆したが，最後に知識獲得システム全体としての今後の方向について述べる．

3.4 知識獲得システム

知識獲得システムは，エキスパートシステムに対するメタエキスパートシステムである．つまり，知識獲得という問題を解決するためのエキスパートシステムが知識獲得システムとなる．今後は，この観点で知識獲得システムをとらえたアプローチの強化が必要となる．換言すると，知識ベース型知識獲得システムの方向がますます重要となる．そこでは，ドメインに独立した知識獲得アーキテクチャと，ドメインに依存した知識獲得用ヒューリスティックの抽出がポイントとなる．

4. おわりに

エキスパートシステム開発での最大のボトルネックといわれる知識獲得の現状と今後の方向について解説した．そこでは，まず，エキスパートシステムにおける知識獲得の問題と代表的なアプローチ例を，開発フェーズ，獲得段階，基本問題に関して分析し整理し

た．次に，現状の課題と今後の方向に関して，知識の抽出・変換・管理という観点から述べた．最後に，知識ベース型知識獲得システムの重要性を強調した．

参 考 文 献

1) Boose, J. H.: Personal Construct Theory and the Transfer of Human Expertise, Proc. of AAAI '84, pp. 27-33 (1984).
2) Kahn, G., Nowlan, S. and McDermott, J.: Strategies for Knowledge Acquisition, IEEE Trans. PA and MI, Vol. PAMI-7, No. 5, pp. 511-522 (Sept. 1985).
3) Marcus, S., McDermott, J. and Wang, T.: Knowledge Acquisition for Constructive Systems, Proc. of IJCAI '85, pp. 637-639 (1985).
4) Bennett, J. S.: ROGET: A Knowledge-Based System for Acquiring the Conceptual Structure of a Diagnostic Expert System, Journal of Automated Reasoning, Vol. 1, pp. 49-74 (1985).
5) Eshelman, L. and McDermott, J.: MOLE: A Knowledge Acquisition Tool That Uses Its Head, Proc. of AAAI'86, pp. 950-955 (1986).
6) 小高，野村，田岡，山口，溝口，角所：知識コンパイラの構成とその応用，情報処理学会，知識工学と人工知能研究会資料，48-2 (1986).
7) 柳，志村：故障診断用エキスパートシステム，情報処理学会，知識工学と人工知能研究会資料，46-5 (1986).
8) Davis, R.: Interactive Transfer of Expertise: Acquisition of New Inference Rule, Artif. Intell., Vol. 12, pp. 121-157 (1979).
9) Van Melle, W. J.: System Aids in Constructing Consultation Programs, UMI Research Press (1980).
10) Suwa, M., Scott, A. C. and Shortliffe, E. H.: An Approach to Verifying Completeness and Consistency in a Rule-Based Expert System, the AI Magazine, Fall, pp. 16-21 (1982).
11) Doyle, J.: A Truth Maintenance System, Artif. Intell. Vol. 24, pp. 231-272 (1979).
12) deKleer, J.: An Assumption-Based Truth Maintenance System, Artif. Intell., Vol. 28, pp. 127-162 (1986).
13) Politakis, P. and Weiss, S. M.: Using Empirical Analysis to Refine Expert System Knowledge Bases, Artif. Intell., Vol. 22, pp. 23-48 (1984).
14) Ginsberg, A., Weiss, S. M. and Politakis, P.: SEEK2: A Generalized Approach to Automatic Knowledge Base Refinement, Proc. of IJCAI '85, pp. 367-374 (1985).
15) Mitchell, T. M., Mahadevan, S. and Steinberg, L. I.: LEAP: A Learning Apprentice for VLSI Design, Proc. of IJCAI '85, pp. 573-580 (1985).
16) Smith, R. G., Winston, H. A., Mitchell, T. M. and Buchanan, B. G.: Representation and Use of Explicit Justifications for Knowledge Base Refinement, Proc. of IJCAI '85, pp. 673-680 (1985).
17) 渡辺，岩本，山之内，松田：VILLA: VLSI 設計知識獲得システム，信学会，人工知能と知識処理研究会資料，AI86-1 (1986).
18) Lathrop, R. H. and Kirk, R. S.: A System Which Uses Examples to Learn VLSI Structure Manipulations, Proc. of AAAI '86, pp. 1024-1028 (1986).
19) Pazzani, M. J.: Refining the Knowledge Base of a Diagnostic Expert System: An Application of Failure-Driven Learning, Proc. of AAAI '86, pp. 1029-1035 (1986).
20) Hayes-Roth, F., Waterman, D. A. and Lenat, D. B. (eds.): Building Expert Systems, Addison-Wesley Publishing Company, INC. (1983).
21) Alexander, J. H., Freiling, M. J., Shulman, S. J., Staley, J. L., Rehfuss, S. and Messick, S. L.: Knowledge Level Engineering: Ontological Analysis, Proc. of AAAI '86, pp. 963-968 (1986).
22) Friedman, J. Y. and Jain, A.: Framework for Prototyping Expert Systems for Financial Applications, Proc. of AAAI '86, pp. 969-975 (1986).
23) Kline, P. J. and Dolins, S. B.: Problem Features That Influence the Design of Expert Systems, Proc. of AAAI '86, pp. 956-962 (1986).
24) Michalski, R. S.: A Theory and Methodology of Inductive Learning, in Machine Learning(ed. by Carbonell, J., Michalski, R. S. and Mitchell, T. M.), pp. 83-134 (1983).
25) Mitchell, T. M., Keller, R. and Kedar-Cabelli, S.: Explanation-Based Generalization: A Unifying View, Machine Learning, Vol. 1, pp. 47-80 (1986).
26) Dejong, G. and Mooney, R.: Explanation-Based Learning: An Alternative View, Machine Learning, Vol. 1, pp. 145-176 (1986).
27) Mitchell, T. M., Utgoff, P. E. and Banerji, R.: Learning by Experimentation: Acquring and Refining Problem-Solving Heuristics, in Machine Learning (ed. by Carbonell, J., Michalski, R. S. and Mitchell, T. M.), pp. 163-190 (1983).
28) Rosenbloom, P. S., Laird, J. E., McDermott, J. and Newell, A.: R1-SOAR: An Experiment in Knowledge-Intensive Programming in a Problem-Solving Architecture, IEEE Trans. PA and MI, Vol. PAMI-7, No. 5, pp. 561-569 (Sept. 1985).
29) Schaefer, P. R., Bozma, I. H. and Beer, R. D.: Knowledge-Based Validity Maintenance for Production Systems, Proc. of AAAI '86, pp. 918-922 (1986).
30) 西田，堂下：LSI の動作記述からの知識獲得について，情報処理学会，論文誌，Vol. 26, No. 6, pp. 1057-1068 (Nov. 1985).

(昭和 61 年 10 月 24 日受付)