記号処理 32－4
（1985. 6. 18）

PROLOGを用いた実用に向けてのエキスパートシステム：KRISP

石川格子　杉八合勲
（株式会社 東芝）

[1]まえがき

昨今、エキスパートシステムとして開発・発表されるものが数多くあり、何れもLISPで作成される場合が多いが、我々は"KRISP"というPROLOGによるエキスパートシステムを開発した。KRISPとは、Knowledge Representation and Inference System in Prologの事で、文字通りPROLOGを用いて知識表現し、PROLOGのバックトラック機構を多いに利用した推論を行なうシステムであり、従来の、知識をIF－THEN形式のルールに置き換えたプロダクションシステムなどでは処理が困難なアプリケーションに適用する事を目的に開発した。従って、KRISPはあらゆる分野に利用可能な汎用を目指すのではなく、特定の問題領域をターゲットとしたシステムである事を前提としているとも言える。KRISPを、実際のアプリケーションに適用した結果、満足な解答が得られ、PROLOGによるエキスパートシステムの有効性が立証された。本論文では、我々のシステム作成におけるPROLOGによる知識表現の基本的考え方を中心に、KRISPの概要、開発経緯を報告する。

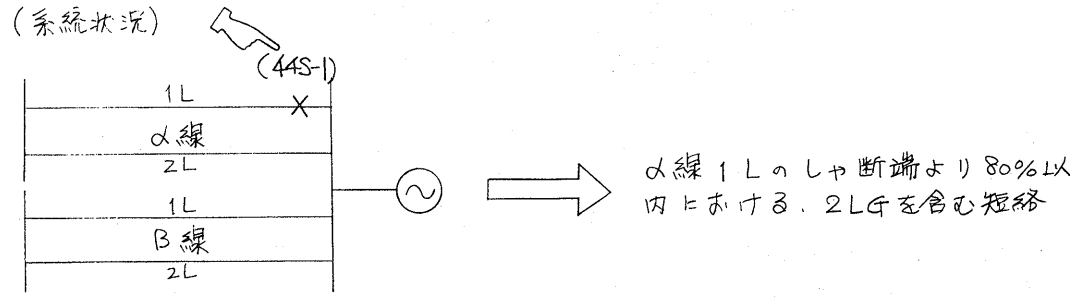
[2]開発の目的

主たる目的は、[1]まえがきでも述べた様に、従来のプロダクションシステムでは、充分に解決し得なかった問題領域に対して柔軟な解決策を与えることである。又、本システムを構築する手段としてPROLOGを用いた事により、PROLOGによる実用に耐え得るエキスパートシステムの可能性、さらに有効性、発展性を確かめ、PROLOGゆえの開発し易さ、開発サイクルの短かさ、プロトタイピングに適している面を探る。

[3]適用分野

KRISPは、IF－THEN形式で表わし得ない部分が生ずる様な問題領域をターゲットとしている訳であるが、今回は、電力系統における系統事故判定に適用した。電力系統において事故が発生すると、事故の波及を防ぐべく保護リレーが作動して開閉器を開放し事故区間を健全系統から切り離す。そして、その時点での電流情報等、いろいろな系統情報が検出される。この様な場合、現在は人間が過去の経験と知識により動作した保護リレー情報、オシロ情報などを基にして事故区間を判断している。しかし、事故時には、事故点で電気の供給がストップしてしまっている訳で、そこを早く回復しなくては、という緊迫した状況にあり、平常心で落ち着いて判断できるかは保障の限りではない。また、人によっては知識が足りない（経験不足）部分があり、本当は起こる可能性のある事故状況に考え及ばない可能性もある。さらに、複数の人間が同時に判定を下した場合、多分に判断基準が違う事が考えられ、異なった判定が下される事もあり得る。従って、冷静で、一定の判断基準に基づき、精度を持った判定を下すエキスパートシステムの必要性が生じてくる。（図－1）に事故判定例を示す。尚、本アプリケーションでは、瞬時に得られる情報（従来人間が

判定する際に用いた情報の内の一部）のみから事故判定を行なっている。

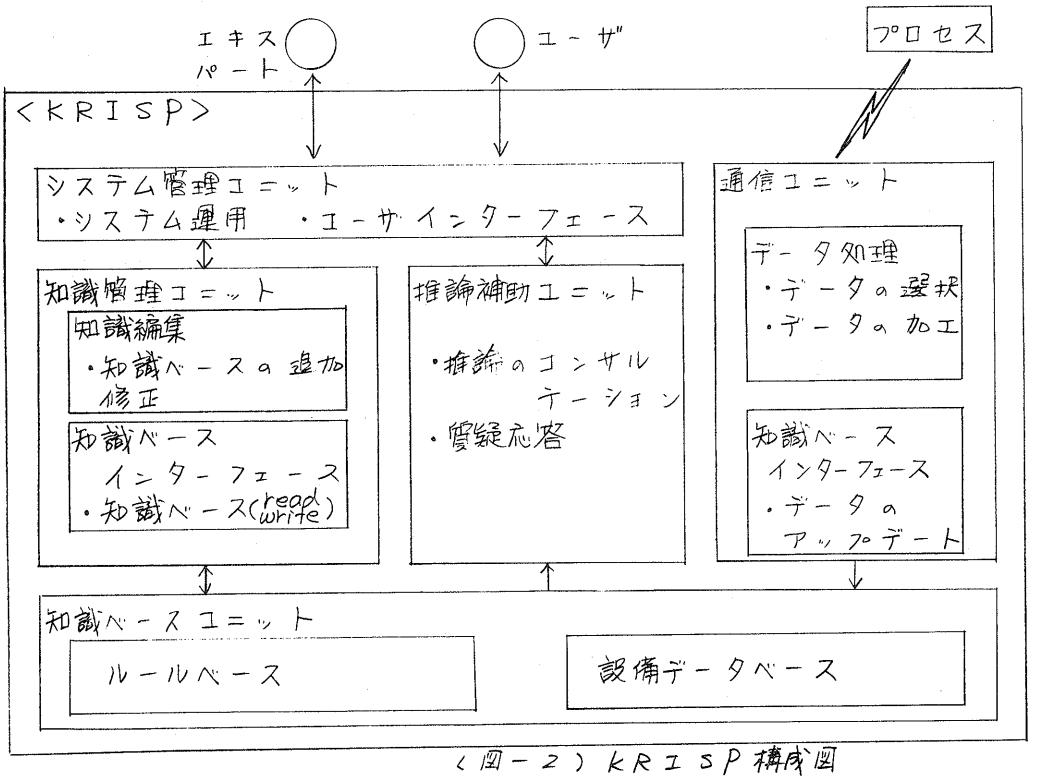


（図－1）事故例

[4] KRISPの構成

KRISPは、システム管理ユニット、知識管理ユニット、推論補助ユニット、通信ユニット、知識ベースユニットから成る。（図－2）

システム管理ユニットは、KRISPの全体の操作をコントロールする部分で、人間（ユーザ、エキスパート）とシステムの持つ機能とのインターフェースを司る。知識管理ユニットは、知識編集の為のユニットで、追加・修正機能と、そのためのユーザインターフェース機能を持つ。推論補助ユニットは、通常の推論（解答のみ得られる）とはいかない、推論の過程における知識採用の状況、変数

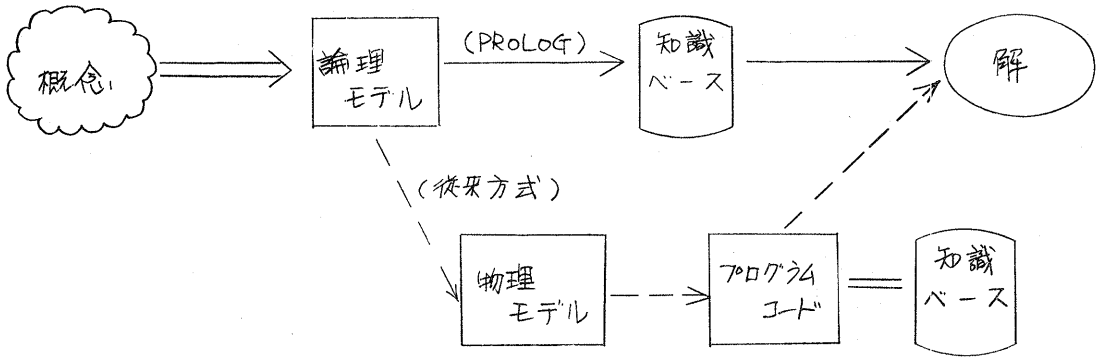


（図－2）KRISP構成図

のユニフィケーションの状況を知る為のものである。通信ユニットは、KRESPへオンライン情報を提供するもので、オンラインで入力された多くの情報の中から必要な情報のみを選択したり、知識ベースに合致した形式に修正するデータ処理、知識ベースアップデートの為のインターフェースが含まれる。知識ベースユニットは、ルールベースとデータベースから構成される。これらのシステムをVAX-11上のC-PROLOGで作成することにした。他の言語ではなくPROLOGを用いた理由は、PROLOGの持つバックトラック機構を、そのまま利用した推論を行なう為である。本アプリケーションでは、電力系統の設備データベースを作成する際に、総て設備間の関係を表わす述語により定義している。例えば、ある送電線：line1の電源側には母線1が、負荷側には母線2が接続しているとすると、"電源側(line1,母線1)．負荷側(line1,母線2)．"となる。この様な形式になっているゆえに、推論において系統の接続状態を追って調べてゆく場合に、論理的な関係と物理的な関係が一致して、述語表現とパターンマッチング機能がそのまま効率的に適用でき人間の思考経路と似た推論が可能になった。また、数あるPROLOGの中でC-PROLOGを選んだ理由は漢字が可能な環境である。データとして漢字が使えるのは勿論であるが、述語名として漢字が使えるというのは、プログラムを理解する上で大変役に立つ。知識を獲得してまとめる際に、箇条書き風にしておき、そのまま それを述語としてプログラミングした訳で、プログラム開発者以外がリストを見ても 内容がわかり易い。事実、複数の人達から、従来の言語に比べて、内容が見え易い との評価を受けた。

[5] PROLOGによる知識表現方法

人間の知識を計算機上に知識ベースとして実現する為には、無形の知識を、有形の知識ベースに変換(transform)する必要がある。人間の知識を、一足飛びに計算機の上への形に変換するのは難しく、何段階かの変換のステップを踏むのが通常であろう。各々の変換には何が為されるかというと、あるレベルの概念(concept)があり、それをまず分析し(identification, decomposition)、抽象化して(abstraction)、まとめる(integration)作業が順に行なわれると考える。言い換えれば、変換とは知識の具象化(realization)である。この変換を基本にして 知識ベースを作り上げてゆくが、知識ベース化の過程は、(図-3)の様に表わせる。PROLOGによるシステムでは、解空間と実世界を論理モデルのみで

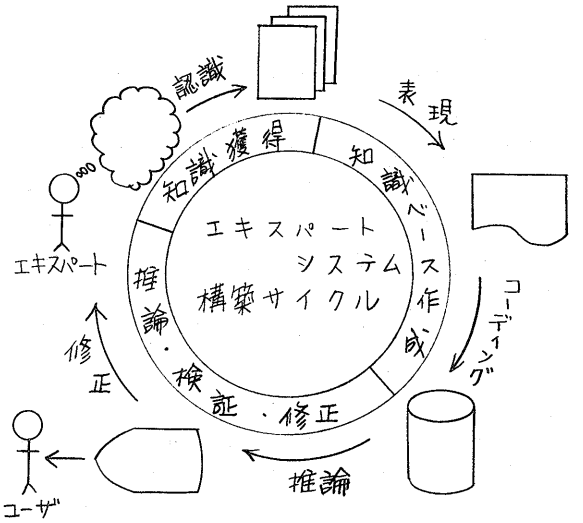


(図-3)知識ベース作成

結びつける事が可能であるが、従来方式であると、一担 論理モデルから物理モデルを形成する必要が生じる。ここに、従来方式とPROLOGとの大きな違いがあり、従来方式による物理モデル形成が、知識の自然な変換を妨げる要因となっている。また、構造化された知識を表現する際、PROLOGでは 1つのまとまった知識をホーン節で書き表わす事が可能であるので、構造化を素直に記述する事ができる。これは、PROLOGによる知識の変換が容易な理由の1つである。

(b) KRISPの開発

一般的な、エキスパートシステムを構築するステップは(図-4)の様に表わされ、KRISPも同様のステップを踏んだが、PROLOGを用いている為、コーディング以降のステップは比較的短期間で可能である。



(図-4) エキスパートシステム構築サイクル

(i) 知識獲得

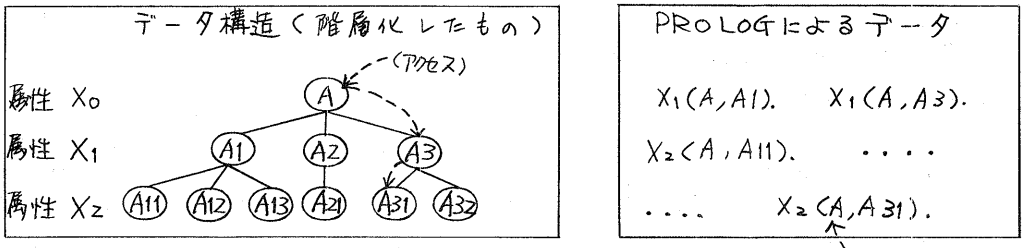
• エキスパート選定 我々は、エキスパートシステム構築に際しての一番のキーポイントは、知識獲得におけるエキスパートの選定にあると思っている。問題領域に精通しているのは当然であるが、広い視野で客観的に分析できる人が望ましい。今回のアプリケーションは、当社に関係の深い領域であり、関係領域のエキスパートは 身近に数多く存在した。その中で、本アプリケーション開発に適任の幅広い視野を持ったエキスパートに巡り会う事ができ、エキスパート選定は成功したと確信している。

• 知識獲得へ認識 エキスパートも決まり、いよいよ知識獲得を行なう訳であるが、エキスパートはシステム構築に関して、反対に システムを開発するエンジニア(Knowledge engineer:以下 KE)は問題領域に関して 素人であるのが普通である。従って、どちらかが他方の領域を、ある程度理解しなければ 知識獲得の作業が円滑に進まない。従って 最初は KEが問題領域を理解する為、エキスパートより説明を受けた。KEの方で 概略(知識のアウトライン)が理解できた時点で、逆に、知識をシステムに移植する事を前提とした形態に基づいて KEからエキスパートにインタビューを行ない、知識の充実、体系化を行なった。最終的には、エキスパート、KEといった役割による分離はなくなり、俗な言い方ではあるが、"二人三脚"の形で知識獲得を完了した。

(ii) 知識ベース作成

• 表現 獲得した知識は、表形式にまとめられるもの(例えば、原理、法則)もあるが、経験的な知識などの単純にIF-THEN形式に置き換えられないものもある。そこで、知識を、IF-THENに拘らずに、内容のわかる様な箇条書き風(条件、操作、結果)にまとめた。まとめる際には、知識のクラス分け、入力情報、出力情報を考慮した。

・コーディング　前述の如くまとめた知識を、PROLOGによりコーディングするが、非常に容易である。カットオペレータ(!)によるバックトラックの制御、リスト操作の手法程度の事を弁えていれば、それらを適切に使う事により、簡単にコーディング可能である。基本的には、知識の表題とパラメータとを頭部とし条件、操作、結果は順に箇条書きの文をそのまま述語名にして'コンマ'で区切って体部に書く。以上は、ルールの話しであり、データの方は、形態が異なる。設備データの持つ属性毎に述語を作成し、意識的にバラバラにした。その際　属性の述語の引数に　データの検索をする際必要となるキーワードも設定してあるので、検索したい属性値に直接アクセスでき、知識(ルール)記述においても分かり易い様になっている。(図－5)



(図－5) PROLOGによるデータ構造

(iii)推論・検証・修正

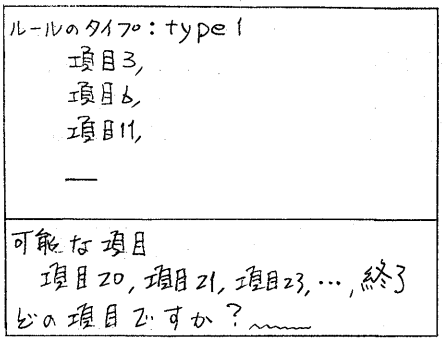
・推論　推論は、PROLOG自身の持つバックトラック機構に依存しており、特に推論機構を作成してはいない。PROLOGは　プログラム上、上から順に検索し、trueになった時点で検索が終了するという性質を弁え、それを利用する事により推論を行なわせる。しかし、どうしてもコントロールし切れない場合には　メタ知識を用いて知識コンパイル時に調整を図る。

・検証　推論の検証用に　各知識の中へ検証用トレース文が埋め込まれており、採用された(trueになった)知識についてのトレースが分かる様になっている。トレース文は、後日、誰が見ても理解できる様に　文章の形を用いた。C-PROLOGの持つデバッグ機能の出力の如く羅列形式では、計算機の素人が見て　分からない場合が多いと思う。それでは、推論の道筋が正しいか否かを検証することは難しいので、手間は掛かったが、文章の形にした。実際に、検証用トレースの結果は　一つの問題に対する解答レポートの様な形になっている。

・修正　推論、検証の結果、知識が間違っているとなれば、修正が必要である。修正には、知識エディタを用いる。設備データ用とルール用とでは、形が異なり、設備データの方は、fill-in-the-blank方式を採る。一方、ルール用であるが、ルールのタイプにより表現形態が異なる為、それぞれのタイプ用のフォーマットが用意されており、ユーザがどのタイプか選択する。ルールは、条件・操作部分と結論部分に分けて表示され、新しくルールを追加する場合には、タイプにより条件、操作、結果として、どの様な順で、どんな項目が可能か予めわかっているので、基本的な項目を用意しておき、ルール作成時に、その時点で使用可能な項目のみを示し、その中からユーザに選択させる方法をとる。この様に、ルール内での条件、操作、結果の出現順をもガイドする事により、ルール作成時に、暗黙にルールの矛盾チェック(100%ではないが)を行なう事となる。設備データ

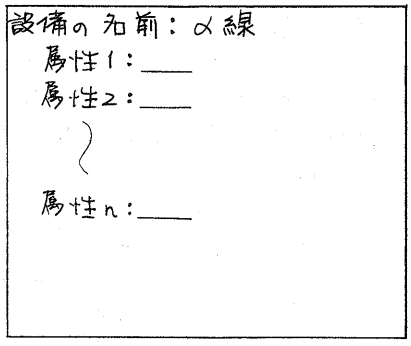
では、設備の実際の名前（例えば"α線"，"β母線"）がキーコードとなっており、名前から 設備の構造に従ったフォーマットを表示する。ユーザは、表示された属性に対して、値を入力する形を取る。（図－6）

・ルールエディタ



ルールのタイプに"type1"と入力した事により、条件節の項目が、順にガイドされ、終了を入力するまで続けられる。

・データエディタ



名前に"α線"と入力した事により、送電線に関する属性1～属性nが表示され、それらについての固有の値を入力する。（α線は、送電線であるとする）

（図－6）知識エディタ

4.評価

(i) 適用したアプリケーションの要求に対する評価

電力系統事故判定に適用したが、実事故例を10件以上（簡単なケースから複雑なケースまでいろいろ）推論した結果、判定結果で、一番確度の高い事故と実際の事故とが統て一致し、正答率が100%であった。時間的にも、本アプリケーションに対し ユーザが許容できる範囲内に入っており、数秒から数十秒で 判定を行なう。

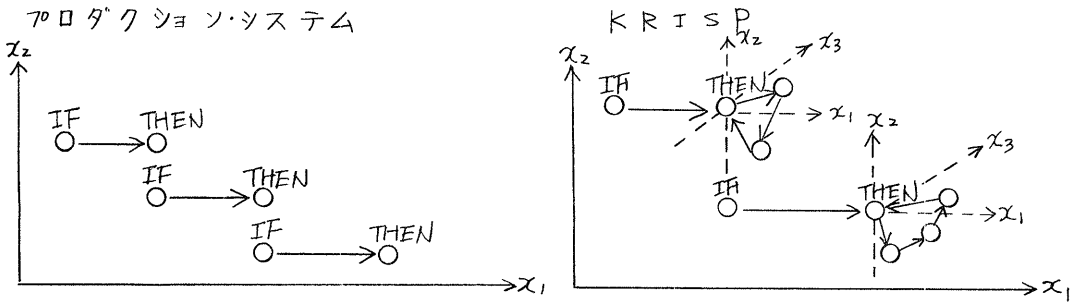
(ii) PROLOGによるシステム構築に対する評価

本アプリケーションの場合、判定結果の重要性と同じ程度に 推論過程も重要である。どの様な順序で、どの知識を採用し、それが採用された要因は何か、という事が 人間の判断の補助となり、知識の確認ともなる。本システムが、実際の現場に設置され、事故の際に判定結果を出力しても、果たして、人間が100%それを信用するかどうかは疑わしい。結果のみ出力されても納得できず、なぜその結果が導かれたかという過程が重要になってくる訳である。今回、知識ベースの中に 埋め込みの形でトレース文を入れ 推論過程を表示する様にしたが、そのトレースと本システムの知識提供者（エキスパート）の推論方法が、ほぼ同じである事が確認された。従って、エキスパートから獲得した知識を、PROLOGにより、そのまま 知識ベースに移植できた事、さらにそれが、推論に素直に反映された事が証明された。(i)で述べた様に、結果は満足できるものであり、PROLOGによる実用に耐え得るエキスパートシステム（即ち、数学の問題の解法 如くではなく、大きな問題領域におけるエキスパート システム）が可能であり、有用で

ある事が分かった。また、獲得した知識を認識してまとめたものを、そのまま書けば、PROLOGのプログラムに近い形になる、という面は他の言語を使用した場合のプログラムとはイメージが違う。使用したC-PROLOGは漢字が使える環境にあった為、尚更知識をまとめた仕様書そのままがプログラムになっている、という感がある。コーディングのステップが、PROLOGの場合短くて済み、プログラムの内容が見て分かり易く今回、日本語が使えた、という要素は大きいが)、という点で、PROLOGによる開発は従来手法より容易と言って過言でない。

(iii)総合評価

適用したアプリケーションの要求に対して、質的に充たす事ができたし、時間的にも、ユーザの許容範囲に入っている。従って、商速化の為他の低レベルの言語を使う必要もなく、逆に、今後コンパイラの普及、ハードウェアの向上が予想される為、今まで指摘されていた速度的な面は、否定材料としての比重が軽くなると思われる。加えて、PROLOGによる推論の質は従来のプロダクションシステムとは次元が異なるものである。(図-6)



(図-7) 推論のイメージ

[8]あとがき

今後、ユーザインターフェースの充実を図り、使い易いシステムを目指して改良、拡張を行なう予定であり、他の問題領域に適用してゆく事により、システムの柔軟性、実用性を増すつもりである。

[9]参考文献

1. W.F.Clocksin, C.S.Mellish: Programming in Prolog, Springer-Verlag, 1981
2. 中島秀之：知識表現とProlog/KR, 産業図書, 1985