1N-2 PROTONによるプログラム変換エキスパートシステム（Prolog to ESP変換の場合）

水谷 晃　　市川 幸二　　澤本 潤
（三菱電機株式会社）（株式会社メガロシステム）（ICOT）

1. はじめに

PSI上のエキスパート・システム構築支援ツールPROTONを用いて、PrologプログラムをESPプログラムに変換するエキスパート・システムを構築した。

最終的には、約300ルールの知識ベースとなった。

一部制限は加わるものの、複数ファイルから成るPrologプログラムを複数クラスのESPプログラムに変換できることが確認できた。

プログラム変換システムの構築に、PROTONを用いた利点、欠点について述べる。

2. PROTONの機能

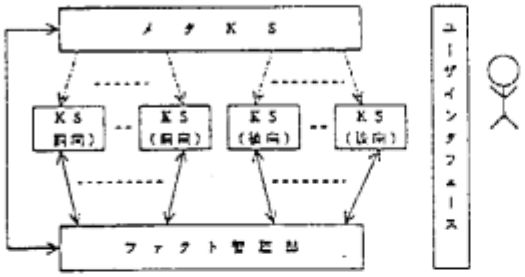
PROTONは次のような機能を持つ。

（1）複数の知識表現機能

問題領域の対象記述をおこなうフレーム表現、フレーム間の関係を表す関係表現、経験的な知識を記述するルール表現、戦略を記述するメタ・ルール表現

（2）柔軟な推論制御機能

ルール・セット（KS）による前向き／後ろ向きの組合せ、ルールによる推論とフレームによる操作の組合せ、メタ・ルールによるKSの選択・実行



PROTONの構成図

3. 変換システムの構成

本システムは、大きくは、推論機構（PROTON）と知識ベースに分かれ、知識ベースはさらに初期設定ルール，入力プログラム分解ルール，変換ルール，出力プログラム生成ルールの4種類のルールに分けられる。

4. PrologからESPへの変換方式

PrologとESPでは、基本的な制御機構はほぼ同じであるが、いくつかの点で違いがある。組込み述語の機能の違いは、一部はマクロで対処し、一部はシミュレート用クラスへのメソッド呼び出しという形に変換する。

最も大きな違いとしてESPがオブジェクト指向言語であり、クラス単位に分割（モジュール化）されるという点がある。

Prologプログラムをどのようにの対応付けたかを以下に述べる。

Prologプログラム1ファイルをESPプログラム1クラスとし、述語の呼び出し関係を解析することにより、クラスの継承関係を作り出す。呼び出し関係がループになっているときは、別途クラスを作り、そのクラスにすべてのクラスを継承させる。

基本的にはすべての述語はESPプログラムではローカル述語とし、public宣言されている述語は他のクラスから呼べるように、インスタンス・メソッドとする。また、定義だけされていて、そのファイル内の他のどの述語からも呼ばれていない述語は、他のファイルから呼ばれていると解釈し、インスタンス・メソッドとする。ファイル内で定義されていない述語は、他のファイルで定義されていると解釈し、メソッド呼び出しの形に変換する。

5. PROTONによるプログラム変換

Prolog／ESPプログラム変換システムの構築にPROTONを用いたことで、次のような利点があった。

（1）宣言的知識表現を用いることで、Prologプログラムを処理しやすい形に表現できた。

（2）関係表現を用いることでProlog述語のヘッド，ボディの関係を容易に表現できた。

（3）手続き的知識のモジュール化により、変換処理を段階ごとに分けて表現でき、管理しやすかった。

Program Translation Expert System on PROTON.
(A Case of Translation Prolog to ESP)
A. Mizutani[1], K. Ichikawa[2], J. Sawamoto[3] (1. Mitsubishi Electric Corporation, 2. MEGALO SYSTEM CO., LTD., 3. ICOT)

一方、次のような不満な点もあった。

（1）ルールを逐次処理的に実行する戦略も取りたい。

（2）複数の同様な静的知識表現を１ルールで一括処理したい。

しかしこれらは、汎用のツールに組み込むべき機能かどうか、検討が必要である。

6．　ツールによるプログラム変換

本プログラム変換エキスパート・システムの構築手順を以下に示す。

（1）ＰＲＯＴＯＮ上でのＰｒｏｌｏｇプログラムのフレーム表現，関係表現等の設計を行なう。

（2）ＰｒｏｌｏｇプログラムをＰＲＯＴＯＮ内に取り込む部分（初期設定ルール，入力プログラム分解ルール）を作成する。

（3）変換結果を出力する部分（出力プログラム生成ルール）を作成する。

（4）変換ルールをいくつか作り、簡単なプログラムの変換を行なう。

（5）変換ルールを追加し、変換できる範囲を広げる。

（6）（5）を繰り返す。

上記のように、ツールを用いたことにより、プロトタイプを早い時期に作り（上記（4））、ルールの追加（上記（5））を行なうことで、動かしながら（動作確認を行ないながら）機能の拡張を行なうことができた。

初めから仕様を決めてしまうのではなく、今回のように作っていく過程で機能を次々と拡張していくようなシステム構築に対してツールの利用は有効であることが確認できた。

7．　おわりに

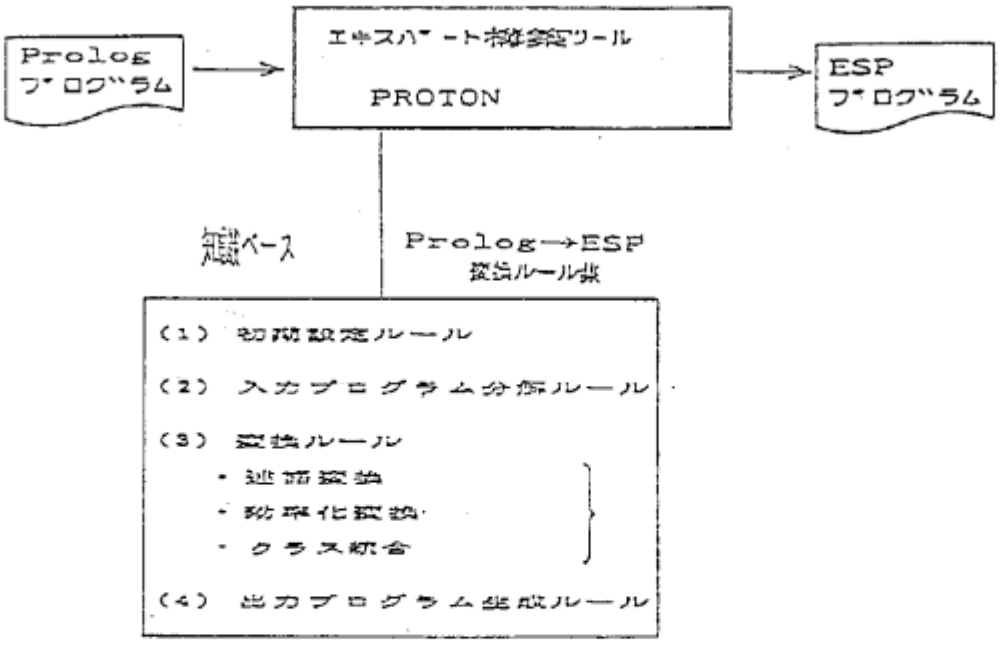
汎用エキスパート・システム構築支援ツールＰＲＯＴＯＮを用いてプログラム変換エキスパート・システムを構築した。

ＰＲＯＴＯＮの評価を目的として開発を開始したため、Ｐｒｏｌｏｇプログラムの変換率等はあまり追求せず、ルール数を多くすることや、ＰＲＯＴＯＮの各機能をできるだけ使いこなすことに重点を置いていた。

しかし、プロトタイプを動かし、ルールを追加していくことで、変換率が高くなり、一部制限のもとで、複数ファイルから成るＰｒｏｌｏｇプログラムを複数クラスのＥＳＰプログラムに変換できるようになった。

今回は、プログラム変換という立場からＰＲＯＴＯＮを利用し、評価した。今後は、他の問題領域の知識ベース構築を行ない、汎用エキスパート・システム構築支援ツールの持つべき機能について検討していく。

<参考文献>　本大会予稿集　１Ｎ－１．



Prolog → ESP 変換機能構成図