情報処理学会第37回(昭和63年後期)全国大会

エキスパートシステム構築支援ツールＫＤＬ（２）

5G-10

推論機構

高橋　和男　　平岡　丈介　　招　行正　　平松　辰夫

（株式会社　明電舎）

１．はじめに

ＫＤＬ（Knowledge Description Language）は、診断・設計・計画など様々な分野の、実用的なエキスパートシステムを構築することを目的としたエキスパートシステム構築支援ツールで、プロダクションシステムとフレームシステムをオブジェクト指向の概念の下に統合している。

当社ではＫＤＬを用いて設計型、計画型など各種エキスパートシステム[1][2]を開発しておりそのうちの幾つかは実用段階となっている。

本稿では特にＫＤＬのプロダクションルールによる推論について報告する。

２．推論サブシステム

ＫＤＬではルールセットを複数定義して推論に使用する。推論を開始すると、まずワールドと呼ぶ推論環境が生成され、そこに複数のルールセット間で推論情報を交換するためのワールドメモリが生成される。同時にワールド中にコンテキストと呼ぶルールセット単位の推論環境が生成される。コンテキスト中にはそこに所属するルールセットだけから参照することのできるローカルメモリが生成される。グローバルメモリはフレームと同様な長期記憶で、推論中の複数のワールド間あるいはルールセット間でデータを交換したり、推論結果を受け渡したりするために使用する。しかしフレームのような一定のデータ構造を持たず、単純な要素の並びとして構成される。

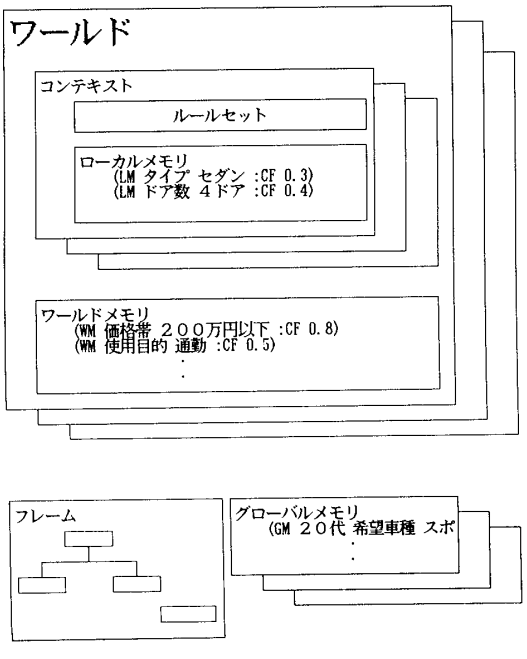

図1　KDLの推論環境

ＫＤＬでは、推論中のワールドから別のワールドを生成し、複数のワールドを用いた推論を行うこともできる。この多重ワールド機構によりスケジューリングなどの問題も効率よく解決することができる。

３．推論の制御

ワールド中のルールセット（コンテキスト）の状態は推論の進行に従って変化する。またメタルールを用いて推論を制御することも可能である。ルールセットの状態変化の流れを図２に示す。

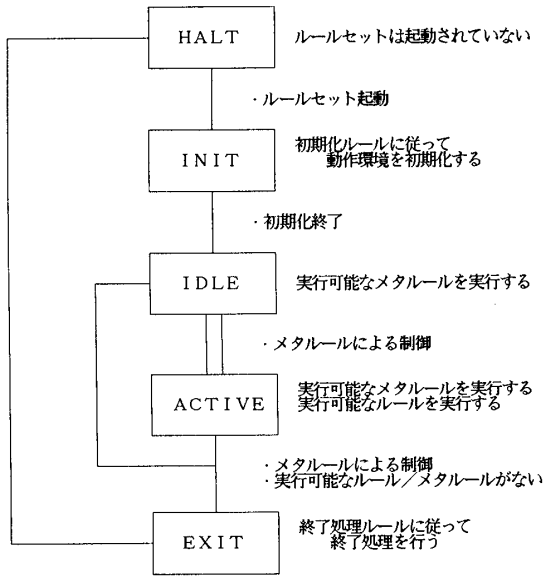

図2　ルールセットの状態遷移

ルールセットは起動された直後にＩＮＩＴ状態になる。ＩＮＩＴ状態のルールセットでは初期化ルールが実行され、推論環境（ローカルメモリ、ワールドメモリなど）の初期化が行われる。推論の初期化は通常の推論と異なり手続的な処理が要求される場合が多い。従って、初期

Expert System Building Tool KDL (2)
Kazuo Takahashi　Jyousuke Hiraoka　Yukimasa Maneki　Tatuo Hiramatu
MEIDENSHA CORPORATION

化ルールは通常の認知実行サイクルには従わず、実行可能なルールを記述順序に従って実行する。

初期化ルールの実行が終了すると、ルールセットはＩＤＬＥ状態へ遷移する。ＩＤＬＥ状態のルールセットでは推論サイクル毎に実行可能なメタルールの集合（コンフリクトセット）が生成され、全てが実行される。ＩＤＬＥ状態のルールセットではメタルール以外のルールは実行されない。

ＩＤＬＥ状態のルールセットはメタルールによって制御され、ＡＣＴＩＶＥ状態へ移行する。ＡＣＴＩＶＥ状態のルールセットでは、推論サイクル毎にメタルールとルールの両方のコンフリクトセットが作られ、競合解消戦略に従い実行可能なルールが実行される。ＩＤＬＥ状態とＡＣＴＩＶＥ状態の間の状態遷移はメタルールを用いて自由に制御することができる。

ワールド中の全てのルールセットに実行可能なメタルールとルールがなくなるか、あるいはメタルールによって推論が終了させられるとルールセットはＥＸＩＴ状態となる。ＥＸＩＴ状態となったルールセットは終了処理ルールを実行した後ＨＡＬＴ状態へ遷移し、そのルールセットが属するコンテキストをワールド中から削除する。ワールド中のコンテキストが全て削除されると一連の推論が終了する。終了処理ルールも初期化ルールと同様にルールの記述順序に従って実行される。これは推論結果を加工するなど比較的定型的な処理が多いことによる。

以上のように、ＫＤＬではメタルールを用いてルールセットの状態をきめ細かく制御し、かつ推論の前処理、後処理を活用することにより複雑な推論を効率よく進めることができる。

４．ルールの記述

ルールセットは次の様に定義する。

```
(defruleset ruleset_name .....
  (rule-1 "exmaple-1"
      (GM    gm_name  家族  ?per1)
      (FRAME ?per1    年齢  (<= ? 20))
      (LM    ?per1    は    ?per2  の  兄です)
      -->
      (assert  (WM  ?per2  は  未成年))))
```

これは比較的単純なパターンマチング機能を使用した例である。この他、ＫＤＬでは以下に示すような条件節を記述することができる。

（１）　(VAR　{list　<条件要素>}*)

これはlistの各要素を並列的に取り出し、条件要素とのテストを試みる。

（２）　(ALL <ﾊﾟﾀｰﾝ> <変数1> <変数2> <条件要素>)

＜パターン＞にマッチするパターンをすべて捜し出して１つのｌｉｓｔを作り、＜変数１＞にバインドする。＜変数２＞にはマッチした＜パターン＞の確信度リストがバインドされる。＜条件要素＞はマッチしたパターンの個数に関するテストを行う。

（３）　(TEST <Ｓ式>　{<条件要素>}*)

Ｓ式を評価し、得られた値と条件要素とでテストを行う。Ｓ式中ではルール変数を参照することができ、また多値を返すこともできる。

（４）ユーザ定義パターン

複数のルールに同じ条件節の並びがある場合に、それらをユーザ定義パターンとして定義することができる。

```
(defpatter pat_1 (nation x y)
    (frame nation cat ?city)
    (frame ?city population (=<< ? x y)))
```

（５）ユーザ定義術語

ユーザが自由に述語を定義することができる。

```
(defpredicate << (x y z) (and (< x y) (< y z)))
```

（６）推論制御

メタルールの条件部ではルールセットの状態などを参照することができる。

(HALT　[ruleset_name])　ＨＡＬＴ状態で真
(IDLE　[ruleset_name])　ＩＤＬＥ状態で真
(ACTIVE [ruleset_name])　ＡＣＴＩＶＥ状態で真
(NULL　[ruleset_name])　コンフリクトセットが空で真

また、実行部にはルールセットの状態を移行させるための述語を記述する。

START　：指定されたルールセットをＩＤＬＥ状態へ
FIRE　：現在のワールドから別のワールドを生成する
TRIGGER：ＩＤＬＥからＡＣＴＩＶＥへ遷移させる
WAIT　：ＡＣＴＩＶＥからＩＤＬＥへ遷移させる
FINISH：ＥＸＩＴ状態へ遷移させる
ABORT　：ＨＡＬＴ状態へ遷移させる

５．おわりに

従来より提唱されている黒板モデルを拡張した。これによりルール型知識をより適切に構造化し、管理することが容易になったと考える。更に、ルールセットにいくつかの状態を与えることにより推論の制御が柔軟になったと考える。

参考文献

[1] 秋田ら：ディジタルリレーソフトウェア設計支援システムの開発、情報処理学会第３４回全国大会、7J-1、高橋ら：ディジタルリレーソフトウェア設計支援システムの開発－２、情報処理学会第３５回全国大会、4L-3、山田ら：ディジタルリレーソフトウェア設計支援システムの開発－３、情報処理学会第３６回全国大会、3Q-5
[2] 後藤ら：知識工学の変電所操作手順自動生成への適用－その１、情報処理学会第３５回全国大会、3P-8、後藤ら：知識工学の変電所操作手順自動生成への適用－その２、情報処理学会第３６回全国大会、7Q-4
[3] 松木ら：エキスパートシステム構築支援ツールＫＤＬ－概要－、本講演論文集（１９８８）