1Q−7

エキスパートシステム構築支援ツール KBMS-3
− メモリ管理方式 −

村山 隆彦　　森原 一郎　　牛島 浩一
NTT 情報通信網研究所

1 はじめに

知識ベースに格納されているルールやフレームなどの知識は，一般にエキスパートシステム (ES) の開発時や推論実行時に頻繁に更新される．これらの知識は，柔軟な構造を持つため，領域の確保 / 開放が頻繁に実行されると，フラグメンテーションなどによるメモリ利用効率の悪化が問題となる．

本稿では，ES 構築支援ツール KBMS-3 (Knowledge Base Management System ver.3)[1] において，知識ベースの管理に必要なメモリを，知識の特性毎に分割管理することによって，効率的なメモリの確保 / 再利用を可能とする方式を提案する．

2 ES 構築支援ツール KBMS-3

2.1 知識とデータ

KBMS-3 は，Lisp 言語ベースの KBMS-2[2] をもとに C 言語ベースに拡張した ES 構築支援ツールである．

KBMS-3 では，ルール / フレームなどのいわゆる知識，及びその知識内の数値などのデータを処理する．知識及び知識内のデータは，以下のような特徴を持つ．

ルール: ルールは RETE ネットワークに展開される．ルールは，開発時には更新がかかるが，原則として，実行時に変更されることはない．しかし，RETE ネットワーク中には多くの作業用データ (メモリノードに蓄えられるトークン) が格納されており，実行時に頻繁に変更される．

フレーム: フレームは複数の上位 / 下位フレームを持つ階層構造を形成し，またそれ自身，複数のスロット・値などの組を持ち，それぞれの動的な変更を許している．よって，その実現にはリスト構造などの柔軟な構造が望ましい．

知識内のデータ: KBMS-3 は，C 言語ベースの ES 構築支援ツールであるため，スロット値などのデータも数値 (整数 / 実数)，文字列などの C 言語でサポートされるデータ型が基本である．更に，シンボル，多値データ (一次元リスト) のような Lisp 言語のデータ型を一部採り入れ，知識 (記号) 処理を容易にしている．なお，シンボルは大域データとしてシステム一意である．

2.2 知識のモジュール化

ES が大規模化してくると，知識の維持管理のために知識ベースを一元管理する必要が生じてくる．一方，知識ベースを利用する側からは必ずしもすべての知識ベースを占有する必要はない．このような要求に備えて，KBMS-3 では知識ベースを部分知識ベース (以下，KB と呼ぶ) に分割し，必要な KB を動的に統合しながら，効率的に推論処理を行うことができる [3].

3 KBMS-3 のメモリ管理方式

3.1 メモリ再利用の効率化

KBMS-3 は，C 言語をベースとすることにより，処理性能が高い，従来情報処理システムとの結合 / 組み込みが可能，などの利点が得られる反面，Lisp 言語で容易に利用できたリストなどの柔軟なデータ構造を実現しようとすると，メモリの確保 / 再利用などの管理を行わなければならない，という問題が生じる．特に，推論などのようにデータの作成 / 変更 / 削除が頻繁に起こる場合には，フラグメンテーションなどによるメモリ利用効率の悪化が問題となる．

そこで，KBMS-3 で使用する知識及び知識内のデータを特性 (種別) 毎に管理し，頻繁なメモリ確保 / 再利用に効率よく応えるようなメモリ管理を実現した．着目点は，以下の 2 点である．

1. メモリ管理を考える時，対象となるデータがモジュール化されていると容易である．そこで，メモリの確保 / 再利用は前述した KB 単位で行う．各 KB 内のデータは，シンボル (大域データ) を除いて，他の KB 内のデータと共有することはない．そこで，各 KB が使用状態になった時，その KB 内で使用するデータの領域を確保し，メモリにロードする．また，終了後，その KB の領域を一括回収することが可能である．
2. 知識，及び知識内のデータは C 言語の構造体で実現されるが，再利用の効率化を図るためには，構造体の大きさが等しいことが望ましい．そこで，各構造体毎に一定領域 (ルームと呼ぶ) を割り当て，そのルームに対してメモリの確保 / 再利用を行う．

3.2 メモリの確保 / 再利用

3.2.1 ルーム管理

各 KB 内では，システム (OS) から確保したメモリ領域を一定領域の大きさ (ルーム) 毎にまとめて管理する．

メモリ確保: ルーム管理が OS に対してメモリを要求する．確保された領域はルームとして KB 内で一意な番号が付与される．

メモリ再利用: 空になった (解放要求された) ルームを，ルーム管理テーブルのフリーリストに加える．OS への解放は行わない．

An Expert System Building Tool KBMS-3
− Memory Management Techniques −
Takahiko Murayama, Ichiro Morihara, Kouichi Ushijima
NTT Network Information Systems Laboratories

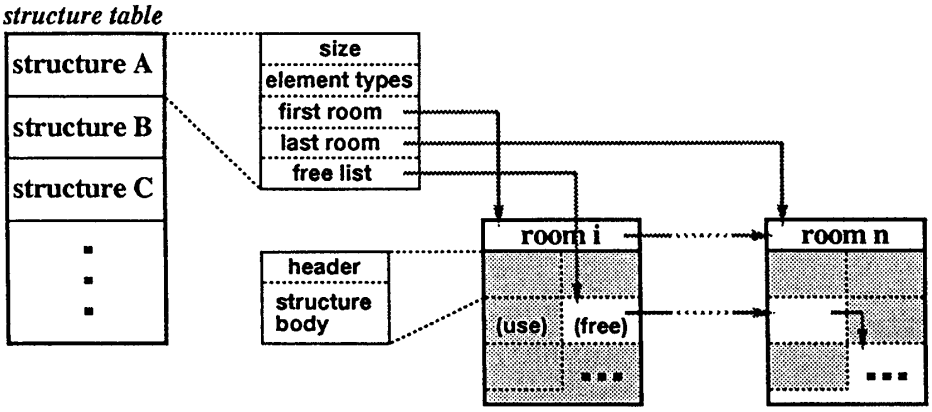

図 1: ルームによる構造体の管理

3.2.2 構造体のメモリ確保 / 再利用

各 KB で利用する構造体に関する情報を格納する構造体テーブルを用意し，構造体の種別，サイズ，及び各メンバの型を格納する．構造体の情報を得る場合には構造体の識別番号をキーとしてこのテーブルを参照する (図 1)．

構造体定義インタフェース: KB 毎に利用する構造体が異なるため，構造体テーブルへの構造体に関する情報の登録インタフェースを設けた．これにより，構造体に関する情報さえ登録すれば，メモリ確保 / 管理などは共通のメモリ管理が行い，構造体を容易に管理することができる．

メモリ確保: 各構造体のメモリ確保要求に対し，構造体テーブルの指定された識別番号からその構造体用のルームに接近する．ルーム内にフリーリストがあればそれを優先的に使用し，フリーリストがなければルーム内のフリー領域の先頭アドレスからその構造体 *(structure body)* + α *(header)* の領域を確保し，そのポインタを返却する．ルーム内に空き領域がなければ，ルーム管理から新たなルームを確保し，その中から領域を確保する．

メモリ再利用: 各構造体の不要になった領域が解放されると，その構造体用のフリーリストに付け加えられる．領域解放によりあるルームが空になった場合は，ルーム管理に解放要求する．

3.3 メモリ管理とファイル操作

KBMS-3 では，メモリ利用効率向上のために，各 KB が使用状態になった時，その KB をメモリにロードする．その際，KB ソースファイルからパージングしていては遅いため，予めパージングした KB のメモリイメージをバイナリ形式でファイルにセーブし，そのファイルをロードしてメモリに展開するバイナリセーブ / バイナリロードの機能を実現した．

前述したように，各 KB 毎に構造体がルーム管理されているため，ポインタなどのデータ (物理的アドレス) もルーム番号とオフセットの組 (論理的アドレス) で表すことができる．さらに，KB 単位 / 構造体単位にルーム管理しているため，KB と構造体の種別が判れば，その領域に格納されているデータの型が判る．以下に概略を示す．

3.3.1 バイナリセーブ

構造体中のポインタを指している部分を以下のように書き換え，メモリイメージを指定のファイルにセーブする．

シンボル (大域データ): ファイルのシンボルデータ域にシンボル名文字列を書き込み，シンボル (へのポインタ) が格納されていた領域はそのシンボル番号に書き換える．

ポインタ (KB 内): ポインタ先の構造体領域の前の領域に格納されているルーム番号とオフセットに書き換える．

3.3.2 バイナリロード

メモリ上に領域を確保し，指定のファイルをロードする．構造体中のポインタを指している部分 (ルーム番号とオフセット) を以下のように書き換える．

シンボル (大域データ): シンボル番号からシンボル名の文字列に接近し，シンボル名の文字列をインターンし，そのポインタに書き換える．

ポインタ (KB 内): ルーム番号とオフセットからポインタに書き換える．

この方式により，KB ソースファイルからパージングする場合と比較して，KB のロード時間が約 1/5 に減少した．

4 おわりに

本稿では，KBMS-3 におけるメモリ管理方式として，知識の特性毎，すなわち，KB 単位 / 構造体単位に管理する方式を示した．ルームという概念を導入することにより，領域の確保 / 開放を容易にし，メモリ利用効率を向上させた．これにより，C 言語ベースでありながら，メモリ管理に煩わされない知識ベースの構築が可能となった．

参考文献

[1] 服部他：エキスパートシステム構築支援ツール KBMS-3 − 構成方式 −，情処 42 全大, 1992.

[2] 森原他： 推論制御機能を強化した ES 構築支援ツール: KBMS, 人工知能学会研究会資料, 1988.

[3] 島崎他：KBMS におけるマルチ KB 管理方式, 63 信学春季全大, 1988.