計算機アーキテクチャ　83−34
（１９９０．７．１８）

並行プログラムのためのテスト・デバッグ環境「mimsy」に内蔵された推論エンジンの構造

山田　剛，小原　啓義
早稲田大学　理工学部

「mimsy」は、並行プログラム開発時のテスト・デバッグ作業を支援する統合環境であり、内蔵された推論エンジンにより、ターゲットプログラムに対するテスト・デバッグ作業の自動実行や、プログラマーへの助言機能などを実現している。この「mimsy」の推論エンジンは、ルールベースを基本とした構造になっているが、プログラマーの支援のためには、さらにデータベース機能とデータ依存ネットワークを表現する機能が重要となる。本論文では、これらの3つの機能の必要性と「mimsy」の推論エンジンへの統合方法について述べ、RETEアルゴリズムの変形について言及する。

The Structure of the Inference Engine Embedded in "mimsy", the Test Debugging Environment for Parallel Programs

Tsuyoshi YAMADA, Hiroyoshi OHARA
School of Science and Engineering, Waseda University
3-4-1 Ohkubo, Shinjuku, Tokyo, 169, JAPAN

"mimsy" is an integrated environment which supports debugging and testing phases in concurrent program development. It automatically runs testing and debugging procedures, and provides functions which aid the programmer, through an embedded inference engine. This engine of "mimsy" has a structure which is based on a rule-base system. However, in order to achieve the desired objectives of "mimsy", a database and capabilities which express data-dependency networks are also necessary.

In this paper, the necessity of these three functions and their integration to the inference engine is discussed, as well as the modification of the RETE algorithm.

1. はじめに

大規模な並行/並列プログラムの開発は、従来の逐次プログラムの開発に比して困難であり、生産性ならびに信頼性の向上には、高機能な開発支援システムの導入が不可欠である。

しかし、並行/並列プログラムの開発は、逐次プログラムの開発とは異なる面を持っており、プロセス間の相互作用や非決定性と言った固有の問題がある。しかも多くのプロセスからなる並行/並列プログラムでは構造・動作共に格段に複雑になるため、プログラムの並行/並列動作を扱うことができる能力と共に、非決定的なプログラムの挙動を十分に解析・検証する能力が開発支援システムに求められる。

このような並行/並列プログラムの開発を支援する方法としては、幾つかのアプローチが考えられる。しかし、プログラム合成や正当性検証など形式的証明に基礎をおく方法は、一般的に言って、プログラムの大規模化とともに急速に運用が困難になるという性質があり、全面的にこれらの方法に頼ることはできない。また、対照的に(学習なども含めた)ヒューリスティックな処理による開発支援の方法は、プログラムのセマンティクスをシステムに理解させるという問題があり、システムの規模も非常に大きくなる。

我々がこれまで開発を行ってきた「mimsy」[1]-[3]は、いわゆる「知的」な開発支援システムであり、推論エンジンを核として据えたテスト・デバッグ環境である。このmimsyの推論エンジンは、ルールベースに基づいており、プログラマーの開発作業を援助したり、限定された範囲であるがプログラマーに助言を与える機能を備えている。この意味で、mimsyは上記のヒューリスティックな開発支援システムに含めることができる。だが、一般の知的プログラミングシステムとは異なり、プログラムのセマンティクスを抽出するような高度に知的な処理は、推論エンジン中で行っていない。mimsyの開発目的は、開発からテスト・デバッグ段階に至るプログラマーの作業を効率化するための、一貫したプラットフォームを構築する事にある。推論エンジンは、プログラム開発の作業全体を統括制御するためと、開発中のプログラムに関する情報を蓄積・運用する機構として、導入されたものである。

本稿では、このような目的で導入されたmimsyの推論エンジンについて、入出力部と内部の構成を述べ、mimsy全体の構成の中で果たす役割について報告する。

2. mimsyの構成

mimsyは、並行/並列プログラムの開発を行うためのプラットフォームであり、開発工程の中でも特に、テスト・デバッグの工程を支援するよう考えられた開発支援システムである。

並行/並列プログラムのテスト・デバッグ工程には、プロセス間の同期や通信の観測・解析をはじめとして、ターゲット言語から独立したプログラミングモデルの不在・非決定性の取り扱い・分散環境における不透明性の問題など、並行/並列プログラムに固有の問題・要求が存在する[2]。mimsyの設計にあたっては、これらの問題点を解決するために、従来の開発支援システムとはかなり異なったFig.1のような構造を採用し、種々の並行/並列言語を取り扱うことのできる汎用的なシステム構造とした。

mimsyは、Fig.1でDIBと書かれているルールベース型の推論エンジンを中心として設計されており、この推論エンジンがmimsy全体の制御を行う構造となっている。テスト・デバッグ作業において、DIBは、図の

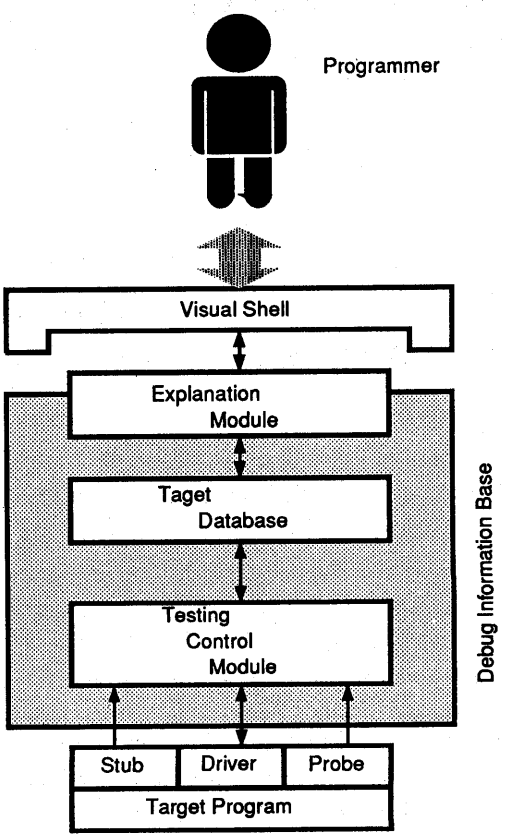

Fig.1 Schematic Structure of "mimsy"

Driverを介して開発対象であるターゲットプログラムの実行を制御し、Probe/stubによって実行履歴や各種情報を収集する。DIBは、また、プログラマーとの対話を制御する役割も担っており、ルールライブラリーを用いて、プログラマーに対してターゲットプログラムの挙動を解析・報告する機能を実現する。

3. マルチプロセスカーネルの構造とプローブ

mimsyに内蔵されたDIBは、ターゲットプログラムを駆動し情報を収集する。現在、ターゲットプログラムとしてはOCCAM言語で書かれたプログラムを取り扱うことができるようになっており、Fig.2のように、OCCAMプログラムをトランスレーターによってSCHEME言語に変換し、それを実行する形式をとっている。マルチプロセスのスケジューリングや同期・通信機構は、SCHEME言語で記述されCONTINUATION[5]を利用したマルチプロセスカーネルによって実現される。

我々のマルチプロセスカーネルは、Table.1のように共有メモリーとカウントセマフォ・CSPをサポートしており、プロセスは動的な生成・消滅が可能である。DIBは、このマルチプロセスカーネルからプロセスの生成・消滅及びプロセス間通信に関する情報を得ることができる。その他のターゲットプログラムに関する情報は、Fig.2にあるように、ターゲットプログラムに直接プローブを挿入し、収集する。

4. 内蔵推論エンジンに要求される機能

我々は、mimsyに用いる推論エンジンとして、C言語に埋め込まれたCOPS[4]と、SCHEMEによるインプリメント[1]を行ってきた。SCHEMEによるバージョンは、デバッグ作業に関するプログラマビリティーを確認するためのものであったが、COPSは、高速化を第一目標として、RETEアルゴリズムを忠実にインプリメントしたルールベースエンジンである。これらのインプリメントを通じて、mimsyに要求される推論エンジンが満たすべき機能と、RETEアルゴリズムの摘要限界が明かになってきた。この章では、先ず、内蔵推論エンジンが備えるべき機能について述べる。

Table.1 Facilities of Multi-process Kernel

make-process	make-semaphore
start-process	wait-semaphore
wait-process-list	make-channel
get-process-staus	send-message
schedule	receive-message
fork-thread	alt-select
wait-thread-list	init-csp-scheduler
get-thread-status	

4.1 ルールベース機能

mimsyが現在内蔵している推論エンジンであるDIBは、OPS系のルールベース機能を持っており、アサーションチェックやブレークポイント、その他のデバッグ機能などを実現する。また、プログラマーがDIB自信をプログラムすることによって、独自の仕様検証ツールや動作解析ツールを組み込むことを許している。

このルールベース機能の導入によって、mimsyを構

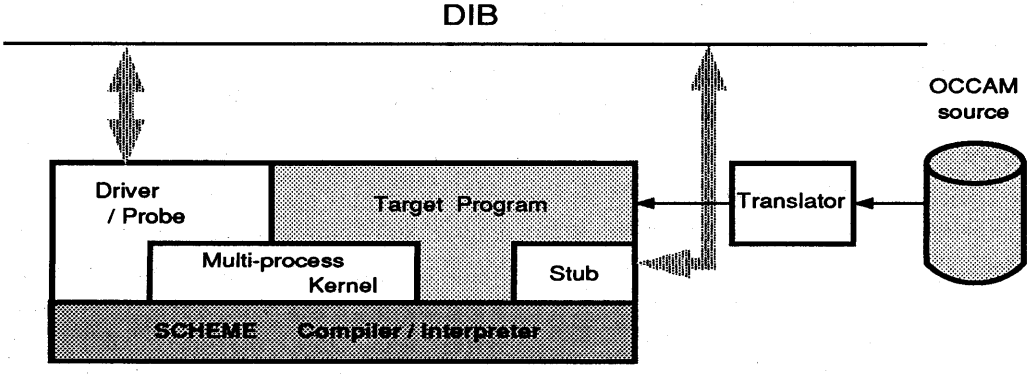

Fig.2 Run-time Environment of Target Program

成する各部の独立性が高くなり、ターゲットプログラムの記述言語に非依存な構造を実現する事ができた[2]。

4.2 データベース機能

mimsyは、最初データベースを基本として設計を開始したが、機能の向上と汎用性の実現のために、ルールベースを核とする構造に再設計された。しかし、ターゲットプログラムの実行から得られる各種情報を検索したり解析するためのデータベース機能は、依然として必要である。

mimsyのような開発支援システムでは、データベース機能は、扱うデータの形式と検索方法が前もって定まっておらず対話的に使用されるため、関係型のデータベースが適している。このデータベース機能を我々のmimsyの推論エンジンでも実現したい訳であるが、特に関係データベースの機構を導入しなくとも、ルールベースで代替が可能である。関係型データベースの処理は、関係演算を基本としているが、この演算はほとんど1対1にルールを用いて書き下すことができる。但し、データベースに対して検索を指示することは、ルールベースに新たにルールを加え、結果が得られた後、そのルールをシステムから除去すると言う操作として実現されるため、対話的にルールを編集することのできるルールベースが必要となる。

4.3 データ依存ネットワーク機能

ターゲットプログラムに関して、推論エンジンが保持すべき情報は、大きく分けて2種類になる。一つは、現在のターゲットプログラムの状態であり、もう一つはターゲットプログラムの挙動に関する履歴情報である。この履歴情報は、時間順に並んだデータの列[1]であるが、種々のデータ依存関係を表すアークと共にデータ依存ネットワーク[6]として組織化される必要がある。

また、mimsyでは、DIBによりVisual Shellの画面表示が制御されるが、例えば、ターゲットプログラムの現在状況をルールにより解釈し画面上に表示するような場合(Fig.3)、その表示を行う理由の一つでも消滅した場合、自動的に画面上の表示が消去されると言う制御を行いたい。このような機能を通常のルールの列で記述しようとすると、表示の消去を行う記述が面倒になる。DIB中のデータ間にTMS[7]で行われているような因果関係のネットワークを張り、そのネットワークをメンテナンスするという機構があれば、このような動作は簡単に表現できる。

このように、mimsyのDIB中では、データ依存ネットワークによるデータ表現が必要である。データ依存ネットワークは、ノードと属性を持ったアークにより表現され、アークはポインターによって実現することができる。しかし、メモリー中でこのような構造を採用すると、ポインターによってデータ依存ネットワークを辿ることは高速にできるが、逆に、ノードの集合を一括して取り扱うというような処理が難しくなる。このような集合操作は4.2節で述べたデータベース機能に重要であり、データ依存ネットワークの実現とは相反する要求となる。

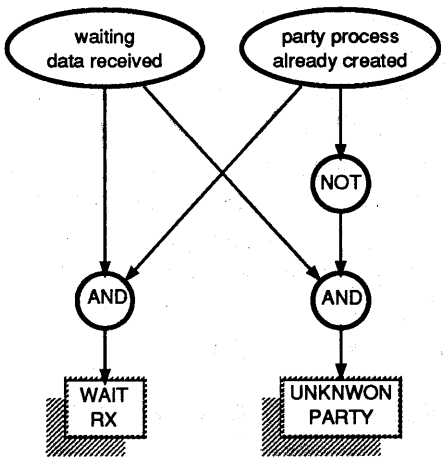

Fig.3 Displaying Alert Box
using Data Dpendency Network

4.4 動的なルールベースによる統合

我々は、mimsyという開発支援環境の推論エンジンに、以上の3種類の機能を必要としている。第1のルールベース機能と第2のデータベース機能は、動的なルールの追加・削除を可能とするようRETEアルゴリズムに修正を加えることによって、ルールベースの枠組に統合可能である。しかし、第3のデータ依存ネットワークの記述能力については、通常のルールベースシステムとデータ表現法が異なるため、統合には工夫が必要である。

ルールベースも関係データベースも、個々のデータは、フラットな形でメモリー中に格納されている。すなわち、データはそれぞれレコード(ルールベースでは、WMEと呼ばれる)として格納されており、それらの間に階層構造やネットワーク構造は存在しない。そのため、検索には時間を必要とするが、階層構造にもネットワーク構造にも束縛されずに、任意のレコードの集合を抽出することができるという利点がある。我々のDIBでも、4.2節で述べたように検索の柔軟性が必要であるため、DIB中のデータ構造としては、このようにフラットなデータ構造をとることが望ましい。

mimsyのDIBには、以上の理由からフラットなデータ構造を採用することとした。これによって、データ依存ネットワークは、すべてWMEを用いて表現することになる。データ依存ネットワークのノードは、WMEを用いて表し、アークは、アークの属性と始点・終点を価として持ったWMEとして表す。

このようなデータ構造によってデータ依存ネットワークを表現すると、データの部分パターンとマッチングをとるルールを記述できるなど、統合による新たな利点が生まれる。また、データ依存ネットワークを辿る際のオーバーヘッドは明かに増加するが、これは将来的に、アークを表すWMEの検索にインバーテッドインデクストゥリーを導入することによって、軽減をはかる予定である。

5. RETEアルゴリズムに対する修正

これまで我々が行ったmimsyの推論エンジンのインプリメントは、RETEアルゴリズムに準拠しており、任意の時点でルールを追加したり修正を行う機能を備えていなかった。mimsyは、対話的なシステムであるので、これは明らかな短所である。ルールの追加・削除・修正を動的に可能にするためには、RETEアルゴリズムを修正して導入しなければならないが、この修正の如何は他の問題とも係わってくる。

RETEアルゴリズムの修正に係わる問題とは、RETEアルゴリズム自信の効率に関する問題である。RETEアルゴリズムは、ルールベースシステムに一般的に用いられるが、万能ではない。特に、リアルタイムシステムにおいて、効率の低下が見られる場合が多い[8]。

5.1 βメモリーの効率

RETEアルゴリズムでは、ルールを予めネットワークの形にコンパイルする。このネットワークの中で、候補として選択されたWMEをマージしてゆく働きを持つのが、Fig.4のTwo-inputノードである。また、Two-inputノード中で、候補WMEの組み合わせを保持するのがβメモリーである。

リアルタイムシステムにルールベースを適用すると、このβメモリーの更新が頻繁に行われる場合があり、RETEアルゴリズムの効率を低下させる結果となる。また、単純にβメモリーの内容が大きくなり過ぎ、オーバーフローを起こす場合もある。

我々のmimsyで、テスト・デバッグのためのルールのサンプルを書き下ろし評価を試みたが、βメモリーの更新に関するオーバーヘッドは、あまり重大な性能低下を引き起こさない。むしろ、重大なのは、βメモリーの容量の問題である。前述のように、mimsyではルールをデータベース検索用としても用いるため、βメモリーが極めて大きくなることがある。そのため、βメモリーを利用しないモードを設けないと、実用性が損なわれることが判ってきた。

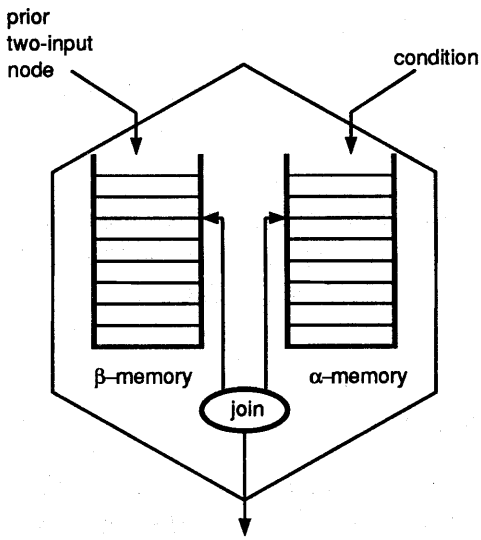

Fig.4 Two-Input Node

5.2 ネットワークの最適化

RETEアルゴリズムでは、ネットワーク中の同じ

ノードをひとつに併合し、処理時間の短縮化をはかる。また、ネットワークの動作をシーケンシャルにシミュレートするためにリニアライズという操作を行い、通常のシーケンシャルプログラムに翻訳するという最適化が行われる。

これらの最適化手法の中で、リニアライズは4.2節で述べたルールの動的な編集を妨げるため、断念せざるを得ない。もう一つの最適化手法であるノードの併合は、処理速度の向上よりもむしろα・β両メモリーの節約のために行うべきである。ノードの併合は、ルールの動的な編集を難しくするが、リファレンスカウントの導入により、これを行う予定である。

6. おわりに

現在、以上のような考察に基づいて、mimsyの推論エンジンの修正と実験を進めている。しかし、mimsyの構造は、リアルタイムな対象に対してルールベースシステムを適用し、しかも対話性が重視されるという特殊なシステムであるため、今後実システムを完動させ、さらなる評価・修正が必要であると考える。

また、今回の報告では取り上げなかったルールベースとVisual Shellの連動についても、以前とは構造が異なっているので、あらためて報告をしたいと考える。

謝辞

「mimsy」の各部の作成に携わっている小原研究室の中村伸一郎・石黒淳・羽鳥和重・平田勝裕・武田小夜里並びに石田秋也の各氏に感謝します。

参考文献

[1] 羽鳥他、“並行プログラム用デバッガにおけるビジュアルインターフェース”、情報処理学会マイクロコンピュータとワークステーション研究会 58-2、Dec.1989

[2] 山田他,“並行プログラムデバッガmimsyのユーザインタフェース”、電子通信学会技術研究報告、CPSY88-40,pp.13-18、Aug.1988

[3] 小松他、“並行プログラム用デバッガmimsyのユーザーインターフェース”、第29回プログラミングシンポジウム、pp.1-11, Jan.1988

[4] 石田他、“プロダクションシステム記述言語COPSによるプロトコルの記述”、アドバンスドネットワーク研究会、Feb.1990

[5] R.Kent Dybvig,"The SCHEME Programming Language", Prentice-Hall,Inc. , 1987

[6] Eugene Charniak,et al.,"ARTIFICIAL INTELLIGENCE PROGRAMMING",sec.11-16,Lawrence Erlbaum Associates,Inc. ,1980

[7] 太原育夫、“人工知能の基礎知識”、第9章、近代科学社、1988

[8] 石田他、“プロダクションシステムの高速化技術”、情報処理 pp467-477、May.1988