小型化版ＣＨＩの資源管理方式

3U-5

藤崎琢巳＊　小長谷明彦＊＊　新　潟＊＊
＊　ビーコンシステム（株）　＊＊　日本電気（株）Ｃ＆Ｃシステム研究所

1．はじめに

小型化版ＣＨＩ[1](以降、ＣＨＩと記す)は、高性能プロセッサと大容量メモリを備え、論理型言語ＳＵＰＬＯＧを実行するバックエンド型推論マシンである。ＣＨＩ上にリモート入出力操作を用いた対話型のプログラミング環境を実現するために、ＣＨＩのオペレーティング・システム(以降、ＯＳと記す)は論理型言語を指向した多重プロセス環境を提供する。本稿では、このような多重プロセス環境における効果的な資源管理の方式について述べる。

2．設計思想

論理型言語では実行時に多くの'ゴミ'を発生する。このようなゴミとしては、'retractしたクローズ'、'ユニフィケーションでコピーした構造体'、'述語呼出の環境を保持するフレーム'等がある。処理系の実現方式として'スタック環境モデル'[2]を採用することにより、これらのゴミの多くはバックトラック時に回収可能である。しかし、このモデルでは'retract したクローズ'やＯＳがシステム制御のために使用したデータ領域は回収できない。

このようなメモリ領域の回収法の代表的な手段として、ガーベージ・コレクション(以降、ＧＣと記す)がある。ＧＣでは現在使用中のメモリ領域にマークを付け、マークがつかなかった領域を回収する。ＣＨＩのような多重プロセスをサポートするマシンでは、次の２通りのＧＣ方式が考えられる。

① 全プロセスを対象として一括してＧＣを行なう方法
② プロセス単位にＧＣを行なう方法

ＣＨＩでは次の２つの理由によって、②の方法を採用する。

- 全メモリ(１２８Ｍワード)の一括ＧＣは長時間の処理の中断を招く。
- ＣＨＩでは機械語の単位の割込みを許している。したがって、割込んだプロセスがＧＣを引き起こした場合には、割込まれたプロセスにおいてＧＣが可能であるという保証はない。

ＣＨＩのメモリ領域は、プロセスに局所的なデータを格納するプロセス空間と、各プロセスに共有されるシステム空間からなる。前者は論理型言語の実行のためのスタックなどに使用し、後者はプロセスに共有されるＯＳのシステム制御のためのコードとデータの格納領域に使用する。このようなメモリ管理方式におけるプロセス単位のＧＣの実現にあたっては次の２つの問題が発生する。

① ユーザが assert したクローズをプロセス空間とシステム空間のどちらに置くか？
② 不必要になったシステム空間の動的データの領域をどのようにして再利用するか？

ＣＨＩでは、クローズはプロセス空間に格納し、プロセス単位に管理する方式を採用した。その理由は、クローズ・データベースを大域的に扱うとプロセス間で競合が生ずること、ならびにクローズ・データベースをプロセス空間に置くことにより、プロセスの消滅によって格納領域を回収することができることによる。②の問題を解決するためには、次のアプローチを採用する。

- システム資源のオブジェクト指向的実現
- システム資源の明示的な解放と領域の回収

次に、これらのアプローチによる資源管理の利点について述べる。

3．資源

ＯＳはメイルボックス、ストリーム、仮想ファイルなどのシステム資源(以降、資源と記す)を提供する。ＯＳの資源の設計と実現はオブジェクト指向パラダイムを用いている。すなわち、個々の資源(インスタンス・オブジェクト)は、資源への操作を記述したクローズと、内部状態を保持する'スロット'から構成される。資源は'クラス'(資源の型)を指定して生成され、これを操作するメッセージを資源に伝えることでその機能を利用する。このようなパラダイムを用いることで、プログラマに対して資源の内部状態や操作の実現法を隠すことができる。さらに、資源単位に領域を確保することが可能になり

Resource Management of CHI Compact Version
Takumi FUJISAKI＊, Akihiko KONAGAYA＊＊, Atushi ATARASHI＊＊
＊ B-CON Systems Inc., ＊＊ NEC Corporation

後述する資源管理の実現が容易になる。

資源のデータ領域の管理は、分身システム(buddy system)[3]を用いて実現する。ユーザが資源を生成すると、資源のスロットのためのデータ領域を分身システムから切り出す。資源が不要になった時、そのデータ領域を再利用するために分身システムに戻す。これを資源の'消去'と呼ぶ。この方法はオブジェクト・テーブル法[4]よりも資源の管理が安価である。また、明示的に資源を消去することにより、システム全体のＧＣを回避することが可能となる。

4. 資源の所有と解放

資源の明示的な消去において、複数のプロセスが資源を共有している場合には、資源を消去するタイミングが問題となる。このタイミングを決定するために、所有資源リストと所有者リストを用いた資源の所有と解放の機構を用意する。

- 所有資源リスト ： 各プロセス毎に該プロセスが'所有'している資源を保持する（図１）。
- 所有者リスト ： 各資源毎に当該資源を'所有'しているプロセスを保持する（図２）。

あるプロセスにおいて資源が不要になった時は、対象とする資源を所有資源リストから外し、当該プロセスをその資源の所有者リストから外す。これを資源の'解放'と呼ぶ。資源を解放した結果、資源の所有者リストが空になると、その資源を消去する。また、プロセスが消滅した場合には、その所有している資源の全てを解放する。以上の所有と解放の機構によって、複数プロセス間で共有している資源のデータ領域についても回収することができる。

5. プロセス間共有資源とジョブ

複数のプロセスで１つの問題を解く協調型問題解決システムを構築する場合、それらのプロセスが同一の資源を共有することが考えられる。このようなプロセス間共有資源の管理を容易にするために、これらのプロセスに共有される資源を問題解決の単位であるジョブが所有できるようにする。これにより、個々のプロセスがその資源を所有する必要がなくなり、所有資源リストや所有者リストの登録／削除を少なくてきる。さらに、ジョブが所有している資源の解放は、ジョブに'所属'するプロセスが全て消滅した時に行なう。このような機構を用意することにより、プロセス間の共有資源の解放を効率よく行なうことができる。

6. まとめ

ＣＨＩのＯＳの資源管理方式について述べた。その特徴を以下にまとめる。

- 資源のデータ領域は分身システムから切り出され、すべてのプロセスが必要でなくなった時、消去される（分身システムに戻される）。
- 資源はプロセスが所有している資源を表す所有資源リストと資源を所有しているプロセスを表す所有者リストで管理する。
- ジョブとプロセスは資源を所有し、その資源が必要がなくなると解放する。
- 解放された資源は、その所有者リストが空になると消去する。
- ジョブが所有する資源は、そのジョブに所属するすべてのプロセスが消滅した時に解放される。

以上の方式を取ることによって、不必要になったシステム空間の動的データである資源の領域の再利用を効率的に行なうことができる。そして、プロセス単位のＧＣを実現し、長時間のＧＣによる実行の中断を回避し、割込みプロセスでのＧＣが可能になる。今後、この資源管理の方式を使用して協調プロセスによる知識情報処理システムおよびマルチプロッセサのシミュレータなどを構築して行く予定である。

参考文献

[1] 中嶋良成 他：逐次型推論マシンＣＨＩ小型化版の設計思想、情報処理学会第３３回全国大会論文集 １９８６

[2] Warren.D.H.: An Abstract Prolog Instruction Set.TR309.Artifical Intelligence Center.SRI International.1983

[3] knuth.D.E.: Fundamental Algorithms.Addison-Wesley.1973

[4] Goldberg.A..et al.: Smalltalk-80 The Language and its Implementation.Addison-Wesley.1983

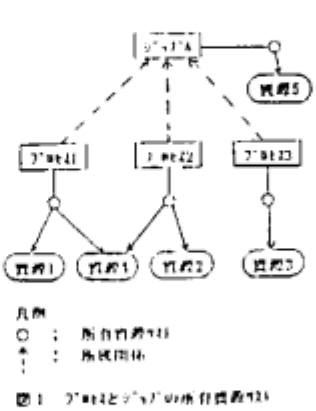

図１

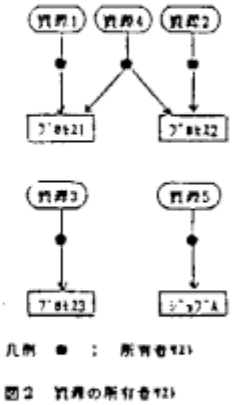

図２