

制約とマルチコンテキストに基づく並列協調問題解決

横山孝典[†] 小野昌之^{††}
和田正寛^{††} 大崎宏^{†††}

本論文では制約とマルチコンテキストを利用した並列協調問題解決モデルを提案し、その実現方式について述べる。このモデルは部分問題間の依存性が大きい設計問題を対象に考案したもので、制約充足機能を有するオブジェクトで表現した設計対象モデルを共有する複数のエージェントが並列かつ協調して問題解決を進める。そして制約充足機能により、設計解の整合性を維持するとともに、制約伝播を利用した協調動作を実現する。また、スロットの組み合わせを扱うノード網を用いたマルチコンテキスト機能をオブジェクトに持たせることにより、複数の解候補を並列に処理可能とする。そして、この方式に基づいた並列協調設計システムのプロトタイプを並列論理型言語 KL1 を用いて、分散メモリ型のマルチプロセッサ計算機 Multi-PSI 上に試作し、その有効性を確認した。

Constraint-Based Parallel Cooperative Problem Solving Using Multiple-Context Objects

TAKANORI YOKOYAMA,[†] MASAYUKI ONO,^{††} MASAHIRO WADA^{††}
and HIROSHI OHSAKI^{†††}

This paper presents a parallel cooperative problem solving model for design problems. This model consists of several agents and a design object model, and constraints on the design object model are used for coordination among agents. A design object model is composed of multiple-context objects which have functions to keep their states satisfying given constraints. The multiple-context management mechanism using a node network, by which combinations of multiple values of slots are processed, is developed to exploit parallelism in the design data. The model is suitable for design problems in which there are heavy dependencies among the subproblems. A prototype of a cooperative design system based on the model presented here has been developed on a multiprocessor machine Multi-PSI using a concurrent logic programming language KL1.

1. はじめに

知識処理における複雑な問題への対応のひとつとして、協調問題解決が注目されている。協調問題解決は問題を定式化が可能な部分問題に分割し、複数のエージェントが並行に部分問題を処理し、協力して全体の解を求める方式である。人工知能には分散 AI あるいは分散協調問題解決と呼ばれる研究分野があり¹⁾、黒板モデル²⁾、契約ネットモデル³⁾、オブジェクト指向モデル⁴⁾などの協調問題解決モデルが提案されている。また、最近並列プロダクションシステム⁵⁾など、並列

問題解決方式の研究が行われている。

我々は設計問題を対象に並列協調問題解決の研究を行っている。設計における協調問題解決とは、部分問題間で中間解などの情報を互いに交換しながら並行に処理を進め、全体としてひとつの要求仕様を満足する解を生成する方式である⁶⁾。ここで中間解とは設計途上で生成される部分的な解候補のことである。

現実の設計は複雑で、複数の部分問題に分割して設計せざるを得ない⁷⁾。例えばひとつのマイクロプロセッサを動作設計、構造設計、タイミング設計などの観点で設計する。しかし、設計対象の共有から生じる部分問題間の依存関係のため、完全に独立な部分問題に分割することはできない。また、解全体の整合性が必要なうえ、問題によっては部分問題ごとの最適解がトレードオフの関係になり、単純に合成して全体の解を得ることはできない。このため、複数の部分問題間で協調して設計解を生成する必要がある。

[†] (株)日立製作所日立研究所
Hitachi Research Laboratory, Hitachi, Ltd.

^{††} (財)新世代コンピュータ技術開発機構
Institute for New Generation Computer Technology

^{†††} (財)日本情報処理開発協会
Japan Information Processing Development Center

これまでに提案された協調問題解決モデルのうち、現実の設計過程を素直にモデル化できるのは、複数のエージェント（知識源とも呼ぶ）が協調して黒板上に設計解を生成する黒板モデルである。このため黒板モデルに基づいた設計システムがこれまでにいくつか発表されている^{8),9)}。

しかしほとんどの黒板システムは本質的に並列動作に適したものとは言えない。最大の問題は逐次処理を前提に、次にどのエージェントの処理を実行するかを決定するスケジューリング機構を用いて協調動作を実現していることである。並列計算機上での実用も試みられている^{10)~12)}が、実用的な手法が確立されるには至っていない。

我々の目的は、ひとつの設計対象を複数の異なる視点で眺め、性質の異なる部分問題に分割して解く協調問題解決システムを、並列計算機上に実現することである。本論文では制約とマルチコンテキストに基づく新しい並列協調問題解決モデルと、その実現に必要な知識表現と問題解決の枠組を提案する。以下ではまず、設計問題における並列協調問題解決の課題について考察する。次に、その課題を実現するため、制約充足機能とマルチコンテキスト機能を持つオブジェクトで表現した設計対象モデルを中心に、複数のエージェントが並列かつ協調的に動作する問題解決モデルを提案する。そして本方式の技術上のポイントである整合性維持と協調動作のための制約充足処理と、複数の解候補を並列に処理するためのマルチコンテキスト管理方式について説明する。最後に本方式を用いた簡単な設計システムのプロトタイプを紹介する。

2. 並列協調問題解決の課題

我々は現実の設計過程との対応がよく、並列計算機上での実現も試みられている黒板モデルをベースに本研究の課題について考察したので、以下に述べる。

(1) 対象知識の表現

並列協調問題解決に有効な知識を自然に記述し、利用できる知識表現の枠組みが必要である。しかも、システム構築を容易にするため、協調動作や並列処理をできるだけ意識せずに記述したい。

また、我々は特に設計対象に関する知識表現の枠組みが重要と考える。設計で用いられる知識は部品の選択法や設計値の決定法などの設計方法に関する知識と、対象の構造や属性、部品間の関係などの設計対象に関する知識に大別できる^{13),14)}。このうち設計方法

に関する知識は部分問題ごとにモジュール化し、エージェントに与えればよい。しかしエージェントが共有すべき設計対象に関する知識を表現し、利用する枠組みは従来の黒板システムにはない。

(2) 整合性の維持

複数のエージェントにより生成された中間解を合成する場合のデータの整合性の維持も大きな課題である。これにはデータ全体の整合性、問題解決のコンテキストの整合性、データ値保持の整合性（評価中に値が変化しない）などが含まれる¹⁰⁾。従来の黒板システムで用いられているような解を合成した後に特定のエージェントで整合性をチェックする方式ではなく、より自然で効率的な整合性維持機構が必要である。

(3) エージェント間の協調

並列処理に適した新しいエージェント間の協調動作が必要である。従来の黒板システムの並列化ではまず問題になるのがスケジューリングである。複数のプロセッサを効率よく活用するためには、非常に複雑なスケジューリングが必要となる。また、スケジューリング処理自身が並列化におけるボトルネックになる可能性がある¹⁰⁾。したがってスケジューラのようなエージェントを一括管理する推論制御機構を用いるのではなく、本質的に並列性のある協調動作を実現することが必要である。

(4) 並列性の抽出

並列協調問題解決では、高い並列性を自然な形で抽出しなければならない。並列化可能な処理としては、黒板上のデータ操作の並列化、複数のエージェントの並列動作、各知識源内の問題解決の並列実行などがある¹¹⁾。しかし、これらの並列化手法のみでは大きな並列度を得るのは難しい。並列化による効率向上の方法としては基本的に、複数の処理の並列実行、データのパイプライン処理、大量のデータの並列処理があり¹⁰⁾、効率的な並列処理を実現するにはこれらを効果的に活用する必要がある。特に、大きな並列度を得るには大量のデータの並列処理が有効であるとの報告がある¹²⁾。

(5) 並列処理の実装

問題解決方式として高い並列性が抽出できても、実際に並列計算機上で高い効率向上を得るには、複数のプロセッサへの処理の割り当てが問題になる。単に各エージェントをそれぞれプロセッサに割り当てるのでは、大きな並列度は得にくい。黒板へのアクセスがボトルネックになる可能性もある。また分散メモリ型の

マルチプロセッサの場合、黒板をどのようにプロセッサに割り当てるかが難しい¹⁴⁾。

以上のように並列協調問題解決システムを実現するための課題は多い。

3. 並列協調問題解決モデル

3.1 並列協調問題解決モデルの提案

我々は前章で述べた課題を解決するため、黒板モデルを基礎とし、制約充足機能とマルチコンテキスト機能を有するオブジェクトを用いた、新しい並列協調問題解決モデルを提案する。

図1は本モデルの基本構成である。複数のエージェントが制約充足機能を有する設計対象モデルにアクセスしながら、協調して処理を進める。以下本方式により前章で述べた課題をいかに解決するかについて説明する。

(1) 対象知識の表現

我々のモデルでは、黒板モデルでは受動的なデータベースであった黒板の代わりに、能動的なオブジェクトで構成される設計対象モデルを採用する。対象モデルは設計対象の形状や構造、種々の属性、部品間の関係、動作などを表現するもので^{13), 15)}、これらの表現に適したオブジェクト指向による表現を採用する。そして、属性間あるいは部品間の関係や設計対象が満たすべき条件を宣言的に記述可能とするために、制約表現機能を導入する¹⁴⁾。要求仕様も設計対象に関する制約と見なすことができる。

(2) 整合性の維持

設計対象モデルを表現するオブジェクトに制約充足機能を持たせることによりデータの整合性維持を実現する。本システムでは、複数のエージェントが設計途上で少しずつ出力する設計データすなわち中間解を、

対象モデルを表すオブジェクトのデータとして合成する。そして、制約充足機能により、合成したデータのうち制約に違反するものを除く。この方式は整合性チェックのためのエージェントが不要になるだけでなく、制約に違反する解候補をただちに枝刈りできることから、効率よい処理が可能である。

(3) エージェント間の協調

協調動作も対象モデルの制約充足機能を利用することにより実現する。各エージェントは、他のエージェントが生成した中間解あるいはそれと関連のあるデータに基づいて自分の中間解を生成する。ここで、制約充足機能により、対象モデルを介した制約伝播を実現して、エージェント間でデータをやりとりし、協調動作を可能とする。この方式は複数の制約伝播を並行に実行することが可能なため、並列性を有する協調動作を実現できる。また、スケジューラのような推論制御機能も不要になる。

(4) 並列性の抽出

本システムでは複数の解候補を並行して取り扱い可能とすることにより、高い並列性を得る。大きな並列度を得るには大量のデータを扱うのがよい。設計解は多数の設計データの集合であり、一般に解候補は複数存在するため、設計データは複数の値をとり得る。設計データの組み合わせによって表される解の状態をコンテキストと呼ぶ。本システムでは複数のコンテキストを並列に処理するため、オブジェクトにマルチコンテキスト管理機能を持たせる。

(5) 並列処理の実装

本システムでは対象モデルを構成するオブジェクトやエージェントの内部の粒度の小さな処理ひとつひとつを並列処理の単位とし、それぞれひとつのプロセスで表現する。そして、並列計算機のプロセッサ数に比較してプロセス数を十分大きなものとし、各プロセッサの負荷ができるだけ均等になるようにそれらを分散させる。

3.2 設計対象モデルの表現

我々は制約充足機能を持つオブジェクトによる対象モデル表現を採用する。設計では設計対象一般に関する知識を用いて、要求仕様を満足する具体的かつ詳細な設計対象モデルを生成する。そこで、対象モデルの一般的知識をクラス宣言で記述し、設計時にオブジェクト(インスタンス)を生成し、設計データ(中間解)をそのスロット(インスタンス変数)に記憶する。

クラス宣言では属性値を記憶するためのスロット名

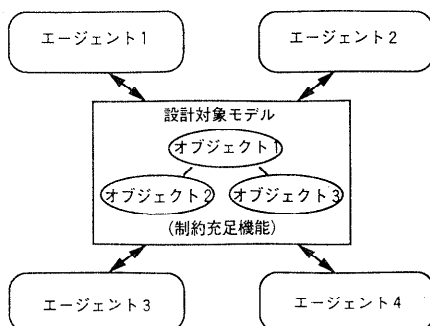


図1 並列協調問題解決モデル

Fig. 1 Parallel cooperative problem solving model.

と、スロット値の範囲やスロット値間の関係に関する制約を宣言する。制約は方程式、不等式、あるいは述語形式の制約式で表現する。制約式中で参照できるのは、そのオブジェクトの持つスロットと、スロットに格納されているオブジェクト（正確にはポインタが格納される）のスロットのみとする。オブジェクトはそれ自身が制約充足機能を持つ¹⁴⁾。

図2(a)に、マイクロプロセッサのクラス宣言の例を示す。“slot”以下に属性を表現するスロット名を宣言し、“constraint”以下にスロット値の組み合わせ禁止（データパスが1パスのときは制御方式として命令先読みは採用できないことをnogoodで指定）や方程式（性能の計算式）などの制約式を記述している。

クラス宣言はトランスレータにより実行形式に変換する。オブジェクトへのアクセスは、同一スロットの場合を除き並列に実行できる。そして、マルチコンテキスト機能により、ひとつのスロットについて複数の値を扱うことができる。また、オブジェクトの持つスロットの変化をエージェントに知らせるデモン機能を提供する。デモンを設定するスロットは、トランスレータがルールを解析して求める。

3.3 エージェントの知識表現と動作

設計方法に関する知識はヒューリスティクスを含む手続き的知識であり、エージェントごとに記述する。我々のモデルは原理的にはエージェントごとに異なる

問題解決方式と、それにあった知識表現を用いることができる。しかし今回試作したプロトタイプでは、現在最も広く用いられていることと開発が容易であることからプロダクションシステムを採用した。

プロダクションルールは図2(b)のようにif-then形式で記述する。この例はクラスが「マイクロプロセッサ」のオブジェクトのスロット「命令動作」の値からデータパス構成を生成し、それをスロット「データパス」に設定することを表している。

ルールはトランスレータにより実行形式に変換する。コンテキストを維持するため、オブジェクトにスロット値を設定する場合、コンテキストの指定が必要である。本システムでは、例えば図示したルールの実行部のset_slot()の部分を実行形式により

```
set_slots (Obj, [ {命令動作, B},
                  {データパス, D} ])
```

という形に変換する。すなわち、実行時には新たに生成するスロット値と条件部で指定されたスロットとの組み合わせを設定することで、正しいコンテキストを維持する。

各エージェントは対象モデルの変化に応じて前向きに推論を行う。具体的には、オブジェクトから変化のあった（新たに設定された）スロット値のデータをデモン機能により受け取り、それが条件部にマッチしたルールを発火し、そのルールの実行部により生成したスロット値（中間解）を返す。生成した中間解はただちに対象モデル上で組み合わせ、制約に違反するものを取り除く。対象モデルとエージェントとのインタフェースはデータフローとし、複数の解候補を次々に処理することにより、パイプライン並列の効果を得ることができる。

複数の解候補を並列に取り扱うため、本システムでは従来のプロダクションシステムでは不可欠であった競合解消処理を行わない。すなわち、発火可能なルールはすべて同時に発火する。そのかわり、複数のルールが発火した後のオブジェクト状態をそれぞれ別のコンテキストとして扱うことにより整合性を保つ。

4. 制約充足

4.1 整合性維持

オブジェクトは新たなスロット値が設定された時に制約評価を実行する。制約評価には受動的な評価と能動的な評価がある。受動的評価とは設定されたスロット値が制約を満足するかどうかをチェックするもの

```
class マイクロプロセッサ has
  slot
    演算器, 内部転送方式, 命令動作,
    制御方式, データパス, 性能, マシンサイクル,
    平均ステップ数, . . . . .;
  constraint
    nogood ( [内部転送方式, 制御方式],
              [1パス, '命令先読み'] ),
    性能 = 1 / (平均ステップ数 * マシンサイクル),
    . . . . .;
end.
```

(a) オブジェクトのクラス宣言の例
(a) An example of class definition

```
agent データパス設計 has
  rule
    if class (Obj, マイクロプロセッサ),
       slot (Obj, {命令動作, B} )
    then generate_data_path(B, D),
         set_slot (Obj, {データパス, D} );
       . . . . .;
end.
```

(b) エージェントのルールの記述例
(b) An example of rule definition

図2 知識表現

Fig. 2 Knowledge representation.

で、能動的評価とは制約評価によってそれまで未定であったスロット値を決定するものを言う。例えば $x+y=10$ という制約が存在する場合、受動的評価とは x と y の両方の値が与えられた後その和が 10 かどうかを調べるもので、能動的評価とはどちらか一方の値が与えられた時にその和が 10 になるように他方の値を決定するものである。

制約の受動的評価は整合性の維持のための基本処理である。すなわち、複数のエージェントが異なるスロットに値を設定した場合、それらのスロット間の制約を評価し、制約を満足しない組み合わせ（コンテキスト）を生成しないようにする。したがって全体として整合性のある解のみを保持するとともに、組み合わせ爆発の回避に利用できる。オブジェクト外に整合性維持機構を設けて解の整合性をチェックする方式は、全体の解を生成した後に整合性チェックを行うため効率がよくない。並列処理のボトルネックとなる可能性があるが、我々の方式はそれらの問題を回避できる。

4.2 制約伝播

制約を能動的に評価することにより、あるエージェントの処理の影響を対象モデルを通して他のエージェントに伝達することができる¹⁶⁾。例えば図3のようにエージェント1の出力したスロットaの値とa・c間の制約を能動的に評価してスロットcの値を決定し、他のエージェントに伝播する。

このような対象モデルを経由した制約伝播を利用することにより、共有データを持たないエージェント間でも直接通信を行うことなく協調動作が可能である。また、制約伝播の方向は実行時に動的に決定されるから、エージェントの動作順序にかかわらず双方向の情報交換ができる。

本システムでは能動的な評価によって値が一意に決定可能な制約式についてのみ能動的評価を行う。例え

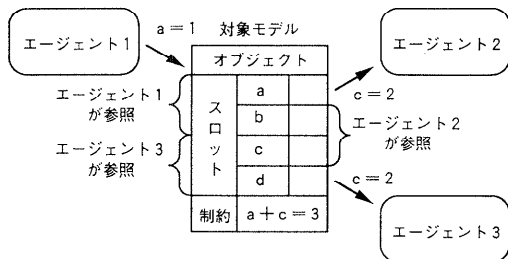


図3 対象モデルを経由したエージェント間の制約伝播
Fig. 3 Constraint propagation among agents.

ば $x+y=10$ という方程式は能動的評価可能であるが、 $x+y>10$ という不等式は受動的評価しか行わない。

5. マルチコンテキスト管理

5.1 マルチコンテキスト

オブジェクトのコンテキストとはスロット値の組み合わせであり、その変化を木構造で表現できる¹⁷⁾。図4はその例で、スロット値の設定に従って、木構造は成長する。しかし、図示したように制約に違反するコンテキストは削除できるため、制約を満足するコンテキストは単調に増加するわけではない。逆に言うとも制約に違反するコンテキストをできるかぎり早期に刈り取ることが処理の効率上重要である。

また、我々の問題解決モデルでは各エージェントが非同期かつ並列に処理を行うため、オブジェクトの生成やスロット値の設定の順序は非決定的である。したがって図示したように分岐したコンテキストがその後同じ状態になることがあり、この場合には両者を統合化して同一のコンテキストとすべきである。

しかしこのような機能を並列処理環境上に効率よく実現する方式は知られていない。複数の仮説の組み合わせを扱う機能を有するシステムとして ATMS¹⁸⁾ があり、各スロット値を仮定 (assumption) と見なして ATMS を適用することが考えられるが、スロット値ひとつひとつを仮定としたのではノード数が膨大になるうえ、制約に違反する組み合わせのみでなく、コンテキストの形成に意味のない組み合わせ（例えば“ス

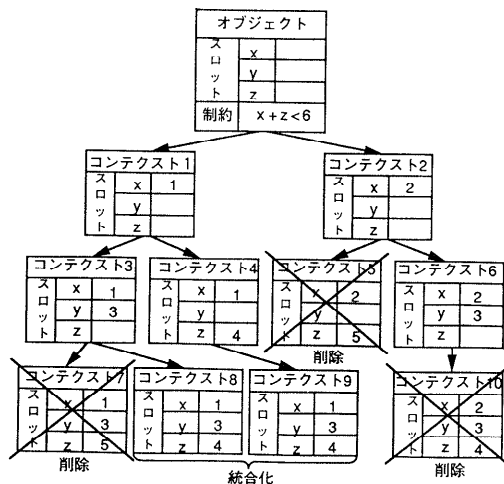


図4 木構造表現したコンテキストの例
Fig. 4 An example of context tree.

ロット 1=1”と“スロット 1=2”の組み合わせ) も生成しないように大量の nogood を与える必要があり、処理が複雑になる。

5.2 マルチコンテキスト・オブジェクト

我々はオブジェクトにスロットの組み合わせを管理するノードを設け、各ノードがスロット値の組み合わせを扱うマルチコンテキスト処理方式を考案した¹⁶⁾。すなわち、図5のように、ひとつのスロットを扱うノードを先頭に、先行するノードが扱うスロットの集合が後続するノードが扱うスロットの集合の部分集合となるようにノード網を形成する。ATMS のノード網と異なり、各ノードはスロット値の組み合わせでなくスロットの組み合わせであり、ひとつのノードが複数の組み合わせデータを扱う。ノード網はオブジェクト(インスタンス)と同時に生成する。

ノードの基本動作は複数の先行ノードからデータを受け取り、組み合わせデータを生成し記憶するとともに、それを後続のノードに送出することである。エージェントのルールで条件部が参照するスロットの組み合わせを扱うノードにはデモンを設定して、組み合わせデータをエージェントにも送るようにする。

制約が参照するスロットの組み合わせを扱うノードにはその制約式を記憶し、ノード自身が制約充足処理を実行する。制約を満足しないスロット値の組み合わせは生成しない。図5では最後尾のノードに制約を満足するスロット値の組み合わせが記憶されている。

ここで、スロットのすべての組み合わせを扱うとノード数が膨大になるため、不要なスロットの組み合わせを扱うノードをなくすように、ノード網の最適化を行う。オブジェクトにとって必要なのは制約が参照するスロットの組み合わせで、エージェントにとって必要なのはルールが参照するスロット値の組み合わせ

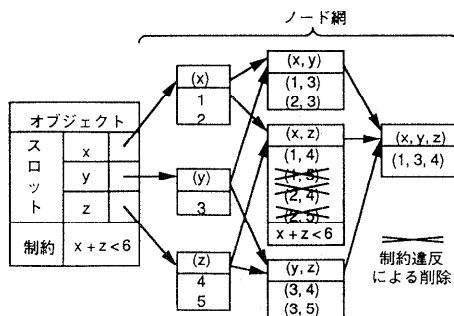


図5 マルチコンテキストのためのノード網
Fig. 5 Node network for multiple contexts.

である。したがって、それら両者の組み合わせを扱うノードのみを生成する¹⁶⁾。最適化はトランスレータによる変換時にオブジェクトの制約とエージェントのルールを解析して行う。

ノードには動的に制約式を追加することができる。この機能は制約表現した要求仕様を与える時に利用できる。もしノードに記憶されているデータのうち、追加した制約に違反するデータが存在すればそれを削除するとともに、それを部分集合としているデータの削除を後続ノードに命じる。これにより制約を満足しないすべてのデータを削除できる。

5.3 動作

エージェントのルールと対象モデルを表現するオブジェクトのノードの動作を図6を例に説明する。対象モデルを4つのエージェントが囲んでいる。図ではルールの表現を簡略化している。以下では、例えば、スロット a, b の組み合わせを扱うノードをノード(a, b)と呼ぶ。

まず要求仕様を与えるときエージェント1とエージェント2の同じ条件部を持つすべてのルールが競合解消なしに並列に発火し、オブジェクトのノード(a)に値1と2、ノード(b)に値3と4が記憶される。それらのデータをノード(a, b)で組み合わせ、制約を満足するデータのみを記憶しノード(a, b, c)に送る。ノード(a, b, c)は制約式を能動的に評価してスロットcの値を算出し、a, b, cの組み合わせデータを生成する。また、算出したcの値はノード(c)に送られ、それを

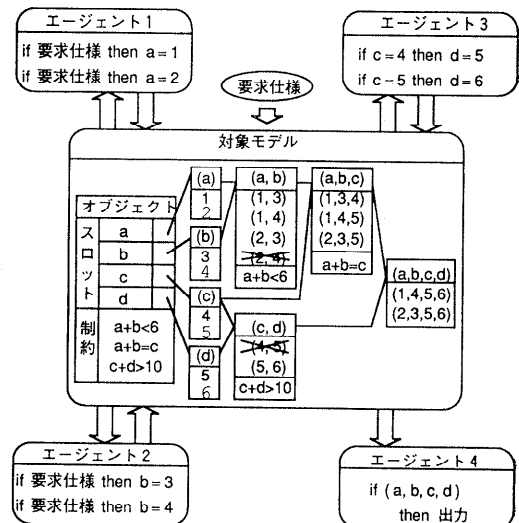


図6 エージェントとノード網の動作
Fig. 6 Agents and node network.

参照するエージェント 3 のルールが発火し、スロット d の値を設定する。ただしノード (c, d) の制約を満足しないデータは削除する。最後にノード (a, b, c, d) で組み合わせデータを生成し、エージェント 4 により出力される。

上記の組み合わせ処理で、複数の先行ノードのデータが同一のスロットを含む場合は、その共通スロットの値が同じもののみ組み合わせ可能である。例えば図 6 でスロット a, b, c の組み合わせデータは 3 つ存在するが、c, d の組み合わせデータは (5, 6) のみであるため、a, b, c, d の組み合わせデータはスロット c の値が 5 のものしか許さない。

以上のようにオブジェクトが持つノード網により複数のコンテキストを同時に取り扱うことができる。ノードはそれぞれひとつのプロセスとして並列に処理を実行可能とする。

6. プロトタイプの試作

6.1 構成

以上提案した並列協調問題解決方式を用いた簡単な設計システムのプロトタイプを試作した¹⁹⁾。対象はマイクロプロセッサの命令セットや性能仕様を入力し、レジスタ・トランスファ・レベルのブロック構造やデータパス、動作記述等を出力する問題である。ただし、本プロトタイプは方式評価を目的としているため問題をできるだけ単純化した。

まず問題を分析し、全体を 5 つの部分問題に分割した。すなわち、マイクロプロセッサのアーキテクチャの基本構造を決定する設計方針決定、命令セットを解析して基本的な動作を決定する動作設計、命令の実行に必要な機能ブロックを抽出するブロック抽出、データの流れる経路を設計するデータパス設計、命令フェッチや命令動作に必要なタイミングをマシンサイクル単位で設計するタイミング設計の 5 つである。

試作したシステムは、図 7 に示すように各部分問題を扱う 5 つのエージェントと、設計結果を出力するエージェント、およびマイクロプロセッサを表す対象モデルから成る。エージェントのルールのほとんどは、要求仕様から制御方式や内部転送方式を生成したり、命令動作記述からそれに合うデータパスを生成するなど、中間解の生成ルールである。

我々は問題の性質を失わない範囲内で、できるだけ少数のオブジェクトおよびルールで動作可能とした。ルール数は 30、オブジェクト数は 2、ノード数は計

45、制約式の数 は 6 である。制約式は性能やゲート数に関する方程式と不等式、および組み合わせ禁止の制約である。このうち方程式のみを能動的評価の対象とする。要求仕様を満足する解はすべて出力する。これは解の評価が難しく、最適解の決定が困難なためである。

6.2 実装

我々はこのシステムを GHC²⁰⁾ をベースとした並列論理型言語 KL1²¹⁾ を用いて開発した。設計対象を表現するオブジェクトのクラス定義と、if-then 形式で記述した各エージェントのルールを、トランスレータにより KL1 プログラムに変換し、それをコンパイルすることにより実行可能となる。ひとつのルールはひとつの KL1 節に変換する。

オブジェクトを構成するノード網は、各ノードを表す永久プロセス間をストリームで接続することにより実現する。永久プロセスは内部データを引数として自分自身を再起的に呼び出す KL1 プロセスである。KL1 を用いることにより、データ駆動に基づく同期や、ストリームを利用したパイプライン並列を容易に実現できる。

本システムは分散メモリ型のマルチプロセッサ計算機、Multi-PSI²¹⁾ 上で動く。現在のシステムでは静的な負荷分散を採用した。まず、各エージェントを各プロセッサに割り振り、次にノードを各プロセッサがほぼ同数のノードを扱うように分散させる。ただし通信オーバーヘッドを考慮し、ノード間の通信量が大きいものに対しては、できるだけそれらノードを同一プロセッサ上に割り当てるようにした。

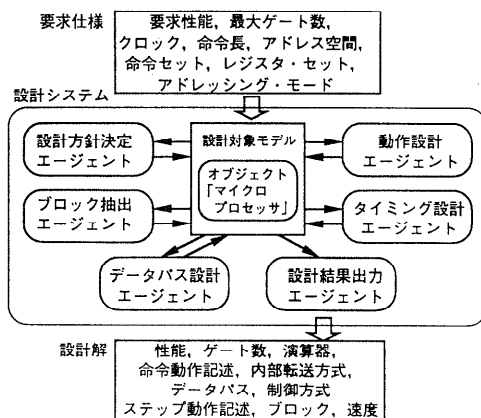


図 7 プロトタイプシステムの構成
Fig. 7 Structure of prototype system.

6.3 結果および評価

(1) システム構築の容易さ

プロトタイプを開発することにより、本方式を用いれば並列協調システムを容易に構築できることを確認した。すなわち、エージェント間の協調動作や、エージェントおよびオブジェクトの並列実行を特に意識することなく、設計対象のオブジェクトと設計方法に関するルールを記述するのみでシステムを構築できた。

今回は対象問題が簡単なため、方程式、不等式、および組み合わせ禁止の制約式でオブジェクトの制約を表現できたが、複雑な問題に対応するにはより多くの種類の制約を扱う機能が必要だと考えている。

(2) 制約充足機能の効果

本方式の特徴である、制約充足機能による解候補の枝刈りについてその有効性を確認した。対象とした問題では、要求仕様および設計対象に関する制約を満足する解は多くの場合数個程度であるが、可能性のある中間解は多数存在し、単純に組み合わせるとおよそ千個の解候補が生成される。ところが、試作したシステムでは各ノードが扱う組み合わせデータの数は数個から十数個程度に押えられ、枝刈り機能が有効に働いている。

今回は問題を単純化して扱ったが、現実の設計では各部分問題で生成する中間解の数、制約の数ともに多数存在する。このように現実の問題、特に組み合わせ設計では、多数の方式あるいは部品の組み合わせがあるが、多数の制約（特に組み合わせ禁止制約）の存在により要求仕様を満足する設計解は少数であるため、本方式の枝刈り機能がより効果的に働く可能性がある。

(3) マルチコンテキスト機能の効果

並列処理を意識せずに並列協調問題解決システムを構築できたのはマルチコンテキスト機能の効果である。試作システムのノード網の動作を観察し、各ノードが並列に処理を実行し、中間解をパイプライン処理するのを確認した。

また、ノード網はプロダクションシステムの実行効率向上に役立っている。すなわち、ノード網は RETE ネットワーク²²⁾の機能の一部を含んでいるため、エージェントにおけるルール条件部のマッチングは非常に簡単な処理ですむうえ、競合解消処理を不要としている。したがって、本技術はプロダクションシステムにマルチコンテキストを導入するときの汎用的な技術として利用できると考えている。

(4) 並列実行

本方式により自然な形で並列性を抽出できた。しかし並列実行効率に関しては、現在のシステムの台数効果がプロセッサ4台の時1台の時の約2倍、8台の時約2.7倍と、必ずしも高い効果は得られていない。処理の前半は各プロセッサの稼働率が100%近いものの、処理の後半ではデータの流れによって処理量の多いノードとそうでないノードが発生し、プロセッサの稼働率に大きなばらつきが生じている。

これは負荷分散が最適でないのと、問題が小さいため、問題中に存在する逐次的に処理せざるをえない部分の影響が大きいとみられている。このうち負荷分散に関しては並列処理の実装の問題で、今後台数効果を向上させるための検討が必要である。また、逐次的な処理の影響についてはより大きな問題に適用して評価、検討を深める必要がある。

7. おわりに

以上、部分問題間の依存性が大きい問題に適した、新しい並列協調問題解決方式を提案した。本方式では、制約充足機能とマルチコンテキスト管理機能を持つオブジェクトにより表現した対象モデルを共有する複数のエージェントが並列に問題解決を実行する。

制約充足機能は解の整合性を維持するとともに、対象モデルを経由した間接的な制約伝播によるエージェント間の協調的な動作を実現する。また、スロットの組み合わせを扱うノード網を用いたマルチコンテキスト機能により、多数の解候補を並列に処理できる。

そして、この方式に基づいた並列協調設計システムのプロトタイプを並列論理型言語を用いて、分散メモリ型のマルチプロセッサ計算機上に試作し、本方式の有効性を確認した。

今後の課題として、現実の大きな問題に対応するため多くの種類の制約を取り扱い可能とするとともに、制約を仮説として扱うことやコンテキストによって異なる制約を扱うなど、機能面での充実を図る必要がある。また実装上の課題として、台数効果のよい負荷分散方式の検討が必要である。

謝辞 本研究は(財)新世代コンピュータ技術開発機構において行った。本研究の機会を与えていただくとともにご指導いただいた、淵一博所長、生駒憲治部長代理（現在 NTT データ通信）、新田克己室長、および日頃ご討論いただく第七研究室の皆さんに感謝する。

参 考 文 献

- 1) Bond, A. H. and Gasser, L. (eds.): *Readings in Distributed Artificial Intelligence*, Morgan Kaufmann, California (1988).
- 2) Nii, H. P.: Blackboard Systems: The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures, *The AI Magazine*, Vol. 7, No. 2, pp. 38-53 (1986).
- 3) Smith, R. G. and Davis, R.: Frameworks for Cooperation in Distributed Problem Solving, *IEEE Trans. Systems, Man and Cybernetics*, SMC-11-1, pp. 61-70 (1981).
- 4) Yonezawa, A. and Tokoro, M. (eds.): *Object-Oriented Concurrent Programming*, MIT Press, Massachusetts (1987).
- 5) Ishida, T.: Methods and Effectiveness of Parallel Rule Firing, *IEEE Conference on Artificial Intelligence Application*, pp. 116-122 (1990).
- 6) Klein, M. and Lu, Stephen C.-Y.: Conflict Resolution in Cooperative Design, *Artificial Intelligence in Engineering*, Vol. 4, No. 4, pp. 168-180 (1989).
- 7) Tong, C.: Toward an Engineering Science of Knowledge-Based Design, *Artificial Intelligence in Engineering*, Vol. 2, No. 3, pp. 133-166 (1987).
- 8) Bushnell, M. L. and Director, S. W.: ULYSES—a Knowledge-Based VLSI Design Environment, *Artificial Intelligence in Engineering*, Vol. 2, No. 1, pp. 33-41 (1987).
- 9) Temme, K.-H. and Nitsche, A.: Chip-Architecture Planning: an Expert System Approach, Gero, J. S. (ed.), *Artificial Intelligence in Engineering: Design*, pp. 137-161, Elsevier, Amsterdam (1988).
- 10) Nii, H. P. et al.: Frameworks for Concurrent Problem Solving: A Report on CAGE and POLIGON, Englemore, R. and Morgan, T. (eds.), *Blackboard Systems*, pp. 475-501, Addison-Wesley, Wokingham (1988).
- 11) Corkill, D. D.: Design Alternatives for Parallel and Distributed Blackboard Systems, Jagannathan, V. et al. (eds.), *Blackboard Architectures and Applications*, pp. 99-136, Academic Press, Boston (1989).
- 12) Rice, J. et al.: See How They Run... The Architecture and Performance of Two Concurrent Blackboard Systems, Jagannathan, V. et al. (eds.), *Blackboard Architectures and Applications*, pp. 153-178, Academic Press, Boston (1989).
- 13) 上野晴樹: 知識工学入門, 6.1 対象モデル, オーム社 (1985).
- 14) 横山孝典, 佐塚秀人: 制約に基づくオブジェクト指向知識表現システム, 情報処理学会論文誌, Vol. 31, No. 1, pp. 68-75 (1990).
- 15) Ohsuga, S.: Conceptual Design of CAD Systems Involving Knowledge Base, Gero, J. (ed.), *Knowledge Engineering in Computer Aided Design*, pp. 29-88, North-Holland, Amsterdam (1985).
- 16) 横山孝典ほか: 並列協調問題解決のための対象モデル表現方式, 第 41 回情報処理学会全国大会論文集, 1 K-6 (1990).
- 17) Walters, J. R. and Nielsen N. R.: *Crafting Knowledge-Based Systems*, Chapter 15 Knowledge Crafting with Multiple Contexts, John Wiley & Sons, New York (1988).
- 18) de Kleer, J.: An Assumption-based TMS, *Artif. Intell.*, Vol. 28, pp. 127-162 (1986).
- 19) 小野昌之ほか: 設計向き並列協調問題解決システムの提案, 第 41 回情報処理学会全国大会論文集, 1 K-8 (1990).
- 20) Ueda, K.: Guarded Horn Clauses, ICOT Technical Report, TR-103 (1985).
- 21) Uchida, S. et al.: Research and Development of the Parallel Inference System in the Intermediate Stage of the FGCS Project, *Proceedings of the Fifth Generation Computer Systems*, pp. 16-36 (1988).
- 22) Forgy, C. L.: RETE: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem, *Artif. Intell.*, Vol. 19, pp. 17-38 (1982).

(平成 3 年 12 月 12 日受付)

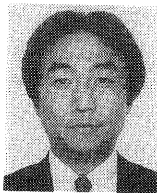
(平成 4 年 11 月 12 日採録)

横山 孝典 (正会員)



1959 年生。1981 年東北大学工学部通信工学科卒業。1983 年同大学院工学研究科電気及通信工学専攻修士課程修了。同年(株)日立製作所入社。1987 年(財)新世代コンピュータ技術開発機構に出向。1990 年(株)日立製作所日立研究所に帰属。ユーザインタフェース, 知識工学, 並列処理, 分散処理の研究に従事。電子情報通信学会, IEEE 各会員。

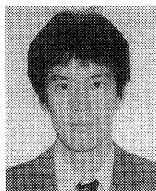
小野 昌之 (正会員)



1962 年生。1985 年上智大学理工学部化学工学科卒業。同年沖電気工業(株)入社。設計向きの知識処理に興味を持つ。1989 年(財)新世代コンピュータ技術開発機構出向。現在, 沖電気工業(株)にて知的 CAD に関する企画に従事。第 43 回情報処理学会全国大会奨励賞受賞。

**和田 正寛**

1963年生。1986年慶應義塾大学理工学部計測工学科卒業。同年シャープ(株)入社、情報システム研究所に勤務。1989年9月より(財)新世代コンピュータ技術開発機構に出向、知識獲得と負荷分散の研究に従事。1993年1月シャープ情報技術研究所に帰社、分散処理の研究に従事。通称わだまん。

**大崎 宏 (正会員)**

1962年生。1985年青山学院大学理工学部化学科卒業。同年(財)日本情報処理開発協会に入社。以来、ネットワークの設計／開発、知識処理システムなどの研究に従事。