並列化による前向き演繹エンジンの高速化

奈良信介[†] 後藤祐一[†] 程京徳[†]

前向き演繹エンジンとは，論理的推論規則を用いて与えられた前提から新しい結論を自動的に導出するシステムである．それは自動定理発見システムや先行計算システムなど様々な分野の情報システムにおいて，不可欠な中心構成要素として求められている．実用的な前向き演繹エンジンを開発する際の最大の課題はその高速化である．本論文では，この課題を解決するために，自動前向き演繹の並列計算モデルを提案し，共有メモリ型並列計算機と PC クラスタ上での実装と実験結果を示し，この並列計算モデルの有効性を示す．

Improving the Performance of Forward Deduction Engines by Parallel Processing

Shinsuke Nara,[†] Yuichi Goto[†] and Jingde Cheng[†]

A forward deduction engine is an automated deduction system which deduces new conclusions from given premises automatically by applying logical inference rules. It is often required as an indispensable component in various advanced information systems, such as automated theorem finding systems and anticipatory computing systems. The most important issue in implementation of a forward deduction engine is its performance. This paper presents a parallel computing model for automated forward deduction and its implementations on a shared-memory parallel computer and a cluster of PCs. We also present some evaluation results of our implementation to show the effectiveness of our model.

1. はじめに

「推論」とは，与えられた前提（既知の事実あるいは仮定）から未知の新しい結論を導出する過程のことである．一方,「証明」とは，あらかじめ明示的に与えられた結論（言明あるいは仮説）に対して，与えられた前提からその結論へ至る論理的道筋を見つけ出す過程のことである．推論の目的は，既知の前提から未知の結論を見つけ出すことであり，証明の目的は，すでに与えられた結論が既知の前提に関してどのように関連づけられるのかを明らかにすることである．

現在，自動定理発見システム，先行計算システムや知識発見システムなど，様々な分野の情報システムで与えられた前提から新しい結論を推論する機能が求められている．このような機能は前向き演繹を計算機上に実装することにより実現できる．

前向き演繹エンジンとは，ある終了条件を満たすまで，与えられた前提やすでに導出された結論に論理的推論規則（以下，推論規則）を適用し，前提やすでに導出された結論に対して新しい結論を導出し続けるシステムである[1)]．実用的な前向き演繹エンジンを開発する際の最大の課題はその高速化である．なぜなら，前向き演繹エンジンは，新しい結論を導出するために膨大な数の中間生成物を処理する必要があるからである．中間生成物とは，前向き演繹エンジンによって導出されるが，利用者にとっては新しくない，または役に立たない結論のことである．

KL1 などの並列論理型言語が開発され，並列化による証明システムの高速化が行われた[2)]．しかし，KL1 の実行系は定理証明法の 1 つである導出原理に基づいている．よって，それらの証明システムにおける高速化の手法は，前向き演繹による推論システムの高速化手法としてはあまり適していない．

本論文では，前向き演繹エンジンの高速化という課題に対し，自動前向き演繹の並列計算モデルを提案し，その有効性を示す．まず自動前向き演繹の処理とその計算量について述べる．次に並列自動前向き演繹のデータの主記憶領域への配置のモデル，および自動前向き演繹の並列計算モデルを提案する．最後に，提案したモデルに基づき，汎用前向き自動帰結演算システム EnCal[1)] における共有メモリ型並列計算機上と

† 埼玉大学大学院理工学研究科
Graduate School of Science and Engineering, Saitama University

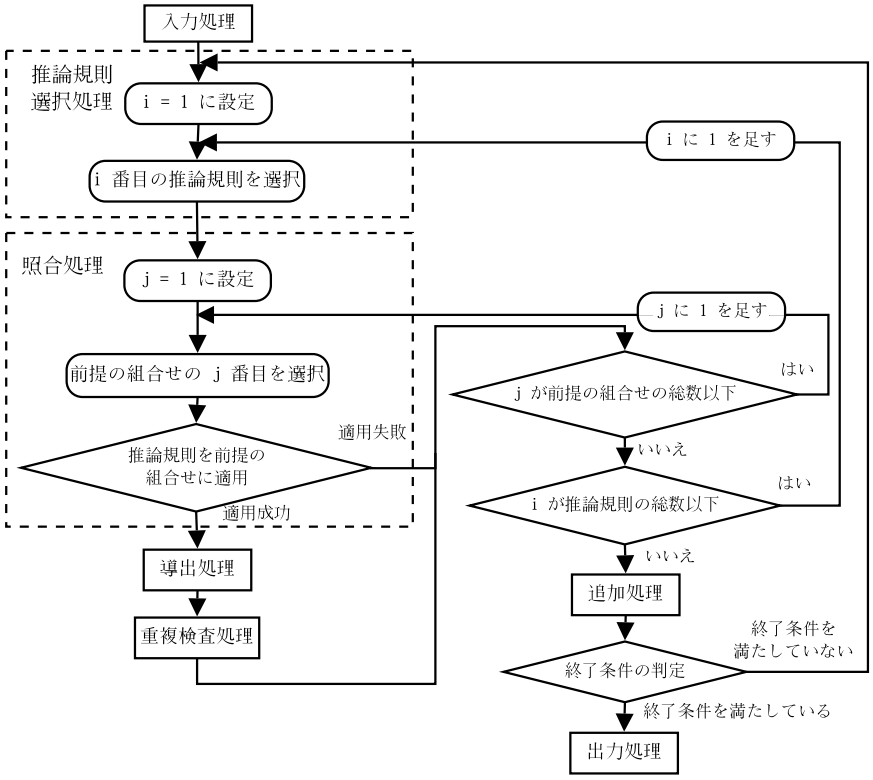

図 1 自動前向き演繹の処理の流れ
Fig. 1 The processes of automated forward deduction.

PC クラスタ上での実装について述べ，その実行時間と速度向上率を調べ，考察を行う．

2. 自動前向き演繹

2.1 自動前向き演繹の処理とその時間計算量

自動前向き演繹は，以下の 5 つの処理をある終了条件を満たすまで繰り返し行う[1]．(1) 推論規則選択処理：推論規則集合から 1 つの推論規則を取り出す．(2) 照合処理：推論規則選択処理で取り出された推論規則が必要とする数の前提を前提集合から取り出し，その前提の組合せにその推論規則を適用できるかを調べる．(3) 導出処理：照合処理に成功した前提の組合せに推論規則を適用し，結論を導出する．(4) 重複検査処理：導出処理で今回導出された結論を前提やその結論が導出される前に導出された結論とマッチングし，今回導出された結論が，それらに対して新しい結論であるかを調べる．(5) 追加処理：重複検査処理で新しい結論であると判断された結論を前提集合に追加する．**図 1** に自動前向き演繹の処理の流れを示す．

マッチングとは，式 α と式 β がある場合，ある処理を適用した結果 β が α と同じ式に変形できるかを調べることである．自動前向き演繹では，マッチングの結果 β が α と同じ式に変形できなかった場合，β は α に対して新しいとし，変形できた場合は新しい結論ではないと判断する．

自動前向き演繹の実行時間を分析する．前提の数を n，推論規則の数を i，ある推論規則が必要とする前提の数を r とする．ここでは，すべての推論規則は，新しい結論を導出するために，r 個の前提を必要とすると仮定する．1 つの推論規則を選択するために必要な実行時間を t_s とする．推論規則選択処理では，推論規則を i 回選択しなければならない．よって，推論規則選択処理の実行時間の合計は，以下のようになる．

$$i \cdot t_s \tag{1}$$

照合処理では，n 個の前提から作ることができる n^r 個の前提の組合せすべてと，i 個の推論規則との照合を行う．よって，1 つの前提の組合せと 1 つの推論規則の照合に必要な実行時間を t_m とした場合，照合処理の実行時間の合計は以下のようになる．

$$n^r \cdot i \cdot t_m \tag{2}$$

導出処理の実行時間が最長となるのは，照合処理で n^r の前提の組合せと，i 個の推論規則の照合処理がす

べて成功した場合である．1 つの前提の組合せを 1 つの推論規則に適用し，結論を導出するために必要な実行時間を t_d とした場合，導出処理の実行時間は，最長で以下のようになる．

$$n^r \cdot i \cdot t_d \tag{3}$$

n 個の前提から，1 つの新しい結論が導出された場合の重複検査処理の実行時間の合計を考える．もし，新しい結論が導出されたならば，それはすべての前提と，その結論より前に導出されたすべての結論との比較を行わなければならない．1 つの前提または，以前に導出された結論と，1 つの導出された結論の比較を行うために必要な実行時間を t_c とした場合，重複検査処理の実行時間の合計は，最長で以下のようになる．

$$n \cdot t_c + 0 \cdot t_c \tag{4}$$

n 個の前提から，2 つの新しい結論のみが導出された場合の重複検査処理の実行時間の合計は，最長で以下のようになる．

$$2 \cdot n \cdot t_c + 1 \cdot t_c \tag{5}$$

n 個の前提から導出される結論は最大で，$n^r \cdot i$ 個となる．この場合の重複検査処理の実行時間は，最長で以下のようになる．

$$\begin{aligned} &n \cdot n^r \cdot i \cdot t_c + (1 \cdot t_c + 2 \cdot t_c \cdots + (n^r \cdot i - 1) \cdot t_c) \\ &\quad = n^{r+1} \cdot i \cdot t_c + \frac{(n^r \cdot i) \cdot (n^r \cdot i - 1)}{2} \cdot t_c \end{aligned} \tag{6}$$

追加処理の実行時間が最長となるのは，導出処理で $n^r \cdot i$ 個の結論が導出され，それらが新しい結論であるときである．1 つの結論を前提集合に加える実行時間を t_a とすると，追加処理の実行時間の最長は，以下のようになる．

$$n^r \cdot i \cdot t_a \tag{7}$$

自動前向き演繹の最長の実行時間のオーダは，前提の数が n ，推論規則に必要とする前提の数が r である場合，$O(n^{2 \cdot r})$ となる．前向き演繹エンジンの実行時間が最長となるのは， $n^r \cdot i$ 個の結論が導出された場合である．よって，式 (1) から，式 (7) までの式から，前向き演繹エンジンの最長の実行時間は以下のようになる．

$$\begin{aligned} &i \cdot t_s + n^r \cdot i\,(t_m + t_d + t_a) + n^{r+1} \cdot i \cdot t_c + \\ &\qquad \frac{(n^r \cdot i) \cdot (n^r \cdot i - 1)}{2} \cdot t_c \end{aligned} \tag{8}$$

また，自動前向き演繹の最短の実行時間のオーダは，$o(n^r)$ となる．前向き演繹エンジンの実行時間が最短となるのは，導出処理において 1 つも結論が導出されなかった場合である．よって，前向き演繹エンジンの最短の実行時間は，以下のようになる．

$$i \cdot t_s + n^r \cdot i \cdot t_m \tag{9}$$

自動前向き演繹の実行時間が膨大である理由は，単に時間計算量のオーダが前提の数に対して多項式的であるということだけではなく，処理するデータである前提の数が膨大になるためでもある．

自動前向き演繹により導出される結論の数について分析する．前提の数を n ，推論規則の数を i ，推論規則が必要とする前提の数を r とした場合，導出される結論の数の最大は以下のようになる．

$$n^r \cdot i \tag{10}$$

n 個の前提から導出された結論の集合を R_1 とする．R_1 を前提として，前向き演繹を行った場合，導出される結論の数は最大で，以下のようになる．

$$n^{r^2} \cdot i^{r+1} \tag{11}$$

R_c を $R_{c-1}(2 \leq c)$ から導出された結論の集合としたとき，R_c の数の最大は，以下のようになる．

$$n^{r^c} \cdot i^{\frac{1-r^c}{1-r}} \tag{12}$$

式 (12) より，仮に前提の数が少なかったとしても，繰り返し処理を行う結果，最終的に導出される結論の数は膨大になることが分かる．

2.2 自動前向き演繹の終了条件

自動前向き演繹の終了条件として，論理式に表れる条件関係を表す論理結合子 $\rightarrow$ の入れ子の深さを基準とする方法が Cheng によって提案された[1)]．この終了条件とは，対象論理体系や形式理論の第 k 級断片を導出するまでというものである．まず，必要な概念の定義を行う．

定義 1 ある論理式 F が第 n 級論理式（$n = 0, 1, \ldots$）であるとは，$deg(F) = n$ であるときとする．また，関数 deg の定義は以下のとおりである．定義中の A と B をそれぞれ論理式を表し，関数 max は，2 つの数を比較し，大きい数を返す関数である．(1) A が $\rightarrow$ をまったく含まないとき，$deg(A) = 0$. (2) $+$ を任意の単項論理結合子とした場合，$deg(+A) = deg(A)$. (3) $*$ を $\rightarrow$ 以外の任意の二項論理結合子とした場合， $deg(A * B) = \max(deg(A), deg(B))$. (4) $deg(A \rightarrow B) = \max(deg(A), deg(B)) + 1$.

定義 2 ある論理体系を $(F(L), \vdash_L)$ とし，ある自然数を k とする．論理体系 L における第 k 級断片（以下，$Th^k(L)$）とは，以下に定義される論理定理の集合である．(1) もし，ある論理式 α が論理体系 L の公理であるならば $\alpha \in Th^k(L)$ である．(2) もし，ある論理式 α が $Th^k(L)$ に含まれる論理式に推論規則を適用して得られた第 j 級論理式（$j \leq k$）であるな

らば，$\alpha \in Th^k(L)$ である．(3) 上記の (1) と (2) 以外の論理式は $Th^k(L)$ に属さない．

この終了条件を利用する場合，自動前向き演繹では，導出された結論の級を計算する．その級が指定された級を超えた場合，その結論は追加処理を適用されない．

3. 自動前向き演繹の並列化

前向き演繹エンジンの実行時間を短縮するために，マスタ・スレーブモデルに基づき自動前向き演繹の並列計算モデルを設計した[3),4)]．以下の理由から自動前向き演繹の並列化には，マスタ・スレーブモデルが適していると考えられる．(1) 自動前向き演繹で行われる各処理は，それが行われる 1 つ前の処理の結果に依存している．(2) 自動前向き演繹で行われる各処理の実行時間に差があるため，各処理を並列化した場合，1 つ前の処理結果を待つ時間が大きくなる．(3) 重複検査処理と追加処理以外の推論規則選択処理，照合処理と導出処理は，それぞれの前提の組合せに対して，独立に処理を行える．

すべてのスレーブが，前提と導出された結論のすべてを参照できれば，重複検査処理と追加処理も独立に実行できる．しかし，分散メモリ型並列計算機上での実装を考えた場合，すべてのスレーブが前提と導出されたすべての結論を参照できるとは限らない．このように各スレーブが前提や導出された結論の一部をそれぞれ保持するという場合には，他のスレーブが保持する前提や導出された結論を利用しなければならなくなる．よって，それらの処理を独立に行えなくなる．

3.1 並列自動前向き演繹における前提の主記憶領域への配置のモデル

本研究では前提の主記憶領域への配置について 2 つモデルを設計した．1 つは，すべてのスレーブがすべての前提を共有するモデル（前提共有モデル）であり，もう 1 つは各スレーブに前提の一部を分散配置するモデル（前提分散モデル）である．

図 2 は，前提共有モデルの図である．前提共有モデルでは，主記憶領域をすべてのスレーブが利用できる全スレーブ共有主記憶領域と各スレーブのみが利用するスレーブ固有主記憶領域に分ける．全スレーブ共有主記憶領域には前提を配置し，スレーブ固有主記憶領域には各スレーブが導出した結論を配置する．

前提共有モデルにおいて結論をスレーブ固有主記憶領域に配置する理由は実行時間の短縮のためである．結論を全スレーブ共有主記憶領域に配置すると，結論を追加する際に排他処理を行う必要がある．自動前向き演繹では膨大な数の結論が導出される．そのため，

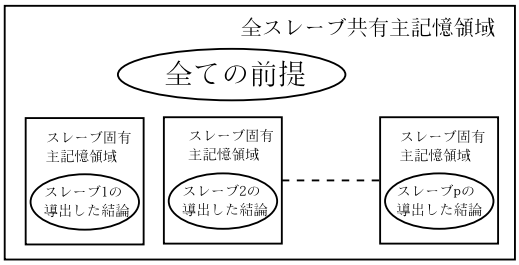

図 2 前提共有モデルにおける前提の主記憶領域への配置
Fig. 2 Arrangement of premises on main memory in shared premises model.

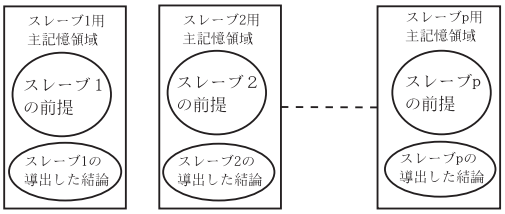

図 3 前提分散モデルにおける前提の主記憶領域への配置
Fig. 3 Arrangement of premises on main memory in distributed premises model.

結論を導出するたびに排他処理を行うと，結論を追加する処理がボトルネックになると考えられる．

図 3 は，前提分散モデルの図である．前提分散モデルでは，各スレーブがそのスレーブ専用の主記憶領域を持つ．スレーブの数を p とした場合，前提分散モデルでは他のスレーブが保持する前提に対して重複のない前提を，前提の総数の $1/p$ 個ずつ各スレーブ用主記憶領域に配置する．また各スレーブが導出した結論もそのスレーブのスレーブ用主記憶領域に配置する．

前提分散モデルは前提共有モデルと較べ，実行時間が増加すると考えられる．前提分散モデルでは，前提の組合せのすべてを処理するために，スレーブ間で前提の送受信を行う必要があるからである．1 つのスレーブが 1 つのプロセスと対応すると考えると，前提共有モデルの場合，すべてのプロセスが 1 つの主記憶領域を直接参照できる．これに対し，前提分散モデルの場合，各プロセスは，複数存在する主記憶領域の中の 1 つしか直接参照できない．よって前提共有モデルではスレーブはすべての前提を高速で参照できるが，前提分散モデルでは，スレーブは，他の主記憶領域に配置された前提を参照するために低速な通信路を利用したプロセス間通信を行わなければならない．

しかし，多くの結論を導出するためには，前提分散モデルを分散メモリ型並列計算機上で実装する必要がある．なぜなら，利用者にとって，新しく面白い結論を導出するために，前向き演繹エンジンがどれだけの

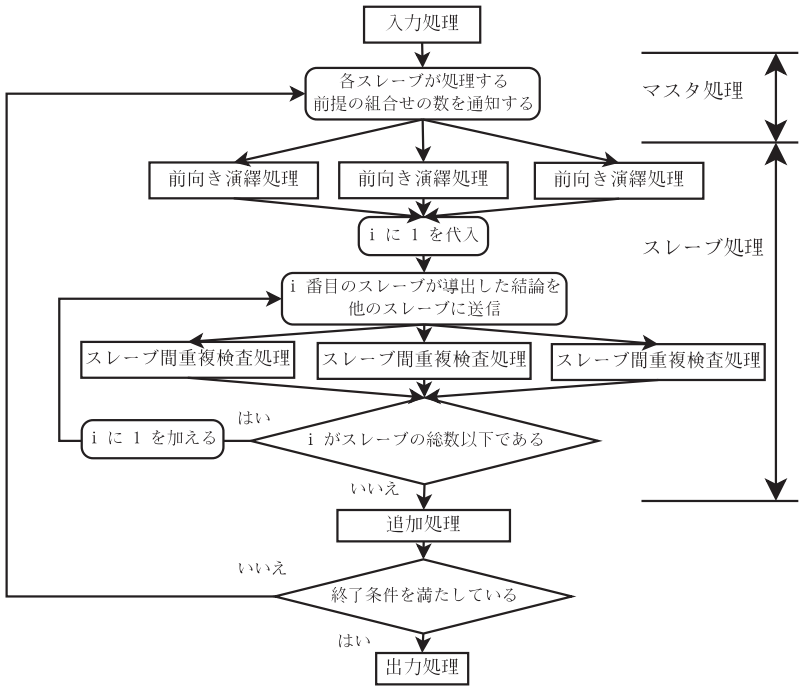

図 5 自動前向き演繹の並列計算モデル
Fig. 5 The parallel computing model of an automated forward deduction.

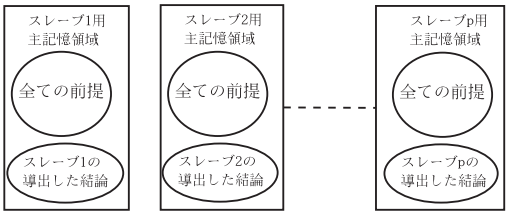

図 4 分散メモリ型並列計算機上における前提共有モデルにおける前提の主記憶領域への配置
Fig. 4 Arrangement of premises on main memory of distributed-memory parallel computers in shared premises model.

結論を導出すればよいかをあらかじめ知ることはできず，そのために必要となる主記憶容量も，前もって知ることができないからである．

図 4 は，分散メモリ型並列計算機上で前提共有モデルを実装する場合の前提の主記憶領域への配置のモデルの図である．前提共有モデルの特徴は，すべてのスレーブがすべての前提を直接参照できるという点である．分散メモリ型並列計算機における前提共有モデルでは，すべてのスレーブ用主記憶領域にすべての前提を配置することにより，これを実現したモデルである．

前提共有モデルを分散メモリ型並列計算機上に実装した場合，前提分散モデルを分散メモリ型並列計算機上に実装した場合ほど多くの主記憶容量が利用できない．しかし，前提のプロセス間通信がないため，前提分散モデルより実行時間が短いと考えられる．

3.2 自動前向き演繹の処理の並列計算モデル

図 5 で，今回設計した自動前向き演繹の並列計算モデルを示す．以下に，各処理の説明を行う．(1) 入力処理：前提，推論規則と終了条件を受け取る．(2) マスタ処理：入力処理直後のマスタ処理では，推論規則をすべてのスレーブに送信する．それ以外のとき，前提共有モデルの場合，マスタ処理では，各スレーブが処理する前提の組合せを静的に決定し，各スレーブに前提の組合せを同じ数ずつ分配する．前提分散モデルの場合，マスタ処理では，各スレーブが保持する前提の数を静的に決定し，各スレーブに同じ数ずつ前提を分配する．(3) スレーブ処理：スレーブ処理は，前向き演繹処理とスレーブ間重複検査処理に分かれる．(3a) 前向き演繹処理：前提共有モデルの場合，推論規則選択処理，照合処理，導出処理と重複検査処理を各スレーブが独立に行う．前提分散モデルの場合，まず，前提取得処理を行い，他のスレーブの前提を取得する．次に，推論規則選択処理，照合処理，導出処理と重複検査処理を他のスレーブと独立に行う．そして，前提消去処理を行うことにより，他のスレーブから取得した前提を消去する．(3b) スレーブ間重複検査処理：前向き演繹処理で取り除けない重複した結論を取り除

く.(4) 追加処理：スレーブが導出した結論を前提集合に加える.(5) 出力処理：導出された結論を出力する.

今回提案した自動前向き演繹の処理の並列計算モデルでは，マスタ処理，スレーブ処理と追加処理をある終了条件が満たされるまで繰り返し行う．今回の並列計算モデルでは，スレーブはマスタから前提の組合せを分配された後，処理を開始するため，マスタとスレーブの処理は同時には行われない．

前提分散モデルでは，前提共有モデルにはない前提取得処理と前提消去処理が必要である．なぜなら前提分散モデルは，前提共有モデルと異なり，スレーブ間で前提を送受信しなければ前提の組合せのすべてを処理できないからである．

自動前向き演繹の並列計算モデルを設計する際に，スレーブ間重複検査処理を新たに追加した．なぜなら，各スレーブが他のスレーブが導出した結論や保持している前提を利用しなければ取り除くことができない重複した結論を導出する可能性が生じたからである．これは，マスタ処理で分配された前提の組合せから，各スレーブが他のスレーブとは独立に結論を導出し，各スレーブ専用の主記憶領域に結論を配置するからである．

あるスレーブの導出した結論は，他のすべてのスレーブが導出した結論とのスレーブ間重複検査処理を行わなければならない．なぜならスレーブ v とスレーブ w があった場合，1 回のスレーブ間重複検査処理では，スレーブ v の結論に対して，スレーブ w の重複した結論を取り除くか，その逆しか行えないからである．2 章で述べたようにマッチングは，2 つの式があった場合に片方の式を変形して，もう片方の式になるかを調べる処理である．よって，スレーブ v と w の重複した結論を両方取り除くためには，スレーブ間重複検査処理を 2 回行う必要がある．

スレーブ間重複検査処理の実行時間を分析する．各スレーブが同じ数の結論を導出し，それらすべてが前提や他の結論に対して新しい結論であると仮定し，その結論の数を m とする．スレーブ間重複検査処理で，処理を行うスレーブの総数を p とする．1 つの結論と，1 つの前提，または結論との比較を行うために必要な実行時間を t_c とする．この場合，スレーブ間重複検査処理の実行時間は，最長で以下のようになる．

$$(p-1)\cdot m^2\cdot t_c \tag{13}$$

3.3 本計算モデルにおける速度向上率の理論値

自動前向き演繹の並列計算モデルの速度向上率の理論値は，重複検査処理とスレーブ間重複検査処理の実行時間によって近似できる．なぜなら，自動前向き演繹の実行時間の大半は重複検査処理とスレーブ間重複検査処理により占められているからである．重複検査処理の実行時間とスレーブ間重複検査処理の実行時間は，すでに式 (6) と式 (13) で述べた．よって，前提の数を n，推論規則の数を i，推論規則が必要とする前提の数を r，スレーブの数を p，導出される結論の数を c（$0 < c \leq n^r \cdot i$）とし，すべてのスレーブが導出する結論の数を c/p と仮定した場合の速度向上率の理論値は，以下のようになる．

$$\frac{n\cdot c+\frac{c\cdot(c-1)}{2}\cdot t_c}{\frac{n\cdot c}{p}\cdot t_c+\frac{\frac{c}{p}\cdot(\frac{c}{p}-1)}{2}\cdot t_c+(p-1)\cdot\frac{c^2}{p^2}\cdot t_c} \simeq \frac{\frac{c\cdot(n+c)}{2}}{\frac{c\cdot(n+c)}{p}} \simeq \frac{p}{2} \tag{14}$$

また，$c = 0$ の場合の自動前向き演繹の並列計算モデルにおける速度向上率の理論値は，照合処理の実行時間によって近似できる．なぜなら，$c = 0$ の場合は，推論規則選択処理と照合処理以外の処理は行われず，さらに推論規則選択処理の実行時間は，照合処理の実行時間に対して十分に小さいからである．この場合の速度向上率の理論値は，以下のようになる．

$$\frac{n^r\cdot i\cdot t_c}{\frac{n^r}{p}\cdot i\cdot t_c}=p \tag{15}$$

よって，理論的には前向き演繹エンジンの速度向上率は，p から $p/2$ の間になる．

この理論値は前提共有モデルと前提分散モデルの両方のモデルで利用できる．なぜなら前提共有モデルと前提分散モデルの処理の違いは，前提取得処理と前提消去処理が追加されたということだけだからである．

自動前向き演繹の並列計算モデルに基づき実装する際に実行時間を増加させる原因として前向き演繹処理では取り除けない，重複した結論の数の増加が考えられる．前提共有モデルでは，各スレーブの前向き演繹処理で導出される結論は，他のスレーブが導出した結論に対して重複している可能性がある．このような結論は，逐次型並列計算機上での実装では取り除かれるものである．しかし，前提共有モデルでは，このような本来は取り除かれるべき結論との重複検査処理も行うため，実行時間が増加すると考えられる．

前提分散モデルは，前提共有モデルに比べ，スレーブ処理での処理時間が増加すると考えられる．なぜなら前提分散モデルは，前提共有モデルに比べ前向き演繹処理で取り除けない重複した結論の数が増加するからである．前提分散モデルは前提を各スレーブに分散配置しているため，他のスレーブが導出した結論に加え，他のスレーブが保持する前提に対して重複した結

論も取り除けなくなる．

また，自動前向き演繹の並列計算モデルに基づき実装する際に実行時間を増加させるもう 1 つの原因としてスレーブの数が増加することによるスレーブ間の同期の回数の増加が考えられる．スレーブ間重複検査処理において各スレーブが保持する重複した結論を完全に取り除くためには各スレーブの結論は 1 回ずつ他のスレーブの結論との重複検査処理を行わなければならない．このためには，各スレーブは他のすべてのスレーブに 1 回ずつ結論を送信しなければならない．あるスレーブが，他のすべてのスレーブに結論の送信するためには，同期をとる必要がある．そのため，スレーブ間重複検査処理における同期の回数は，スレーブの数を p 個とした場合， p 回となり，スレーブの数が増加するごとにスレーブ間重複検査処理で必要となる同期の回数も増加する．

また，前提分散モデルは，前提共有モデルに比べ，スレーブの待機時間が増加すると考えられる．なぜなら前提分散モデルは前提共有モデルに比べ，スレーブ間の同期を行う回数が増加するからである．前提分散モデルでは，スレーブ間重複検査処理に加え，前提取得処理においてもスレーブ間の同期を行う．

前提分散モデルの前提取得処理におけるスレーブ間の同期の回数を分析する．今回のモデルでは，スレーブの総数を p ，推論規則が必要とする前提の数を r 個とした場合，前提分散モデルにおいて前提の組合せを調べ尽くすことを保証する前提の送受信の方法として，1 回の通信で $r-1$ 個のスレーブが他のスレーブに前提を送信し，それを p^{r-1} 回繰り返し行うという方法を利用した．この方法では，前提共有モデルの同期の回数が p 回であるのに対し，前提分散モデルでは， $p^{r-1}+p$ 回の同期が必要となる．

4. 実験と結果

設計した自動前向き演繹の並列計算モデルが高速化に有効であることを示すために，3 章で設計した前提共有モデル，前提分散モデルと分散メモリ型並列計算機上における前提共有モデルの 3 つのモデルに基づき，汎用前向き自動帰結演算システム EnCal[1] を共有メモリ型並列計算機上と PC クラスタ上で実装した．以下の文章では，前提共有モデルと分散メモリ型並列計算機上における前提共有モデルを合わせて前提共有モデルと呼ぶ．EnCal とは 2 章で説明した前向き演繹の 5 つの処理の中の推論規則選択処理を除く 4 つの処理を備えた前向き演繹エンジンである．現在のところ，EnCal で利用できる推論規則は Modus Ponens のみである．Modus Ponens は 2 つの前提を利用し，1 つの結論を導出する．

今回の実装では，1 つのプロセッサを，1 つのスレーブに対応させる．また今回の実装では，ある 1 つのプロセッサがマスタとスレーブの両方の処理を行う．なぜなら今回設計した並列計算モデルでは，マスタの処理とスレーブの処理は並列に行われないため，マスタの処理専用に 1 つのプロセッサを利用することが非効率だからである．

本研究の実行環境として，共有メモリ型並列計算機上での実装では，Sun Enterprise 6000（16 processors）を利用し，実装には C 言語および OpenMP を利用した[5),6)]．また，分散メモリ型並列計算機上での実装では，100Base-TX Megabit Ethernet スイッチによって接続した 8 ノード dual pentium III 1 GHz PC SMP クラスタ（i840 chipset，各ノードに 1 GB RDRAM）を利用した．実装には C 言語を利用し，通信ライブラリとして PC Cluster Consortium[7)] が提供する MPICH-SCore を利用した．共有メモリ型並列計算機には前提共有モデルを実装し，分散メモリ型並列計算機には前提共有モデルと前提分散モデルを実装した．

今回の実験では，前提として 3 つの論理体系の公理を利用した．それぞれの論理体系は，相関論理 Te，Ee，Re である[8),9)]．また，今回は 2.2 節で述べた終了条件を利用する．今回の実験の終了条件は各論理体系における $Th^4(L)$ を導出するまでである．

前提共有モデルでの実装，前提分散モデルでの実装ともに，Te の公理から導出された結論の数は，10,512 であり，Ee の公理から導出された結論の数は，17,902 であり，Re の公理から導出された結論の数は，38,663 であった．

表 1 は，前提共有モデルと前提分散モデルの EnCal の実行時間を示している．表 1 から両方のモデルにおいてプロセッサ数の増加につれて実行時間が減少していることが分かる．また，結論の数に依存せず実行時間が減少している．

図 6，**図 7** と **図 8** は，前提共有モデルの実装と前提分散モデルの実装の 1 プロセッサの実行時間に対する速度向上率のグラフである．これらのグラフから，以下のことが分かる．(1) 共有メモリ型並列計算機上と PC クラスタ上の実装ともに前提共有モデルでは，速度向上率は，速度向上率の理論値 $p/2$ から，p の間に収まっている．(2) PC クラスタ上での前提分散モデルの実装の速度向上率は，プロセッサ数が増加するごとに向上しているが，理論値には至っていない．

表 1 共有メモリ型並列計算機上での実装と PC クラスタ上での実装の実行時間
Table 1 The execution time on the shared-memory parallel computer and the cluster of PCs.

論理体系		1 プロセッサ	2 プロセッサ	4 プロセッサ	8 プロセッサ	16 プロセッサ
Te	共有メモリ型並列計算機	2,341 s	1,272 s	681 s	386 s	238 s
	PC クラスタ前提共有	410 s	222 s	121 s	71 s	44 s
	PC クラスタ前提分散	423 s	308 s	183 s	107 s	81 s
Ee	共有メモリ型並列計算機	11,710 s	6,276 s	3,422 s	1,894 s	1,009 s
	PC クラスタ前提共有	2,089 s	1,078 s	571 s	295 s	179 s
	PC クラスタ前提分散	2,152 s	2,038 s	1,103 s	554 s	361 s
Re	共有メモリ型並列計算機	75,623 s	38,939 s	20,548 s	12,461 s	6,566 s
	PC クラスタ前提共有	14,059 s	7,197 s	3,885 s	2,106 s	1,223 s
	PC クラスタ前提分散	14,386 s	13,425 s	7,361 s	3,541 s	2,333 s

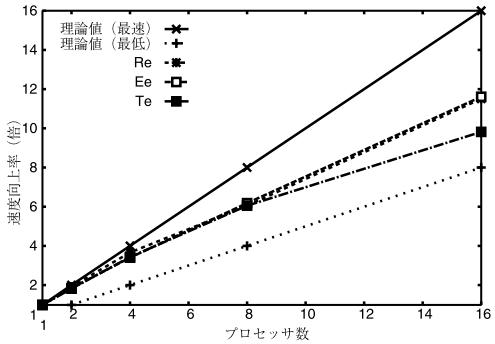

図 6 前提共有モデルにおける共有メモリ型並列計算機上での実装の速度向上率
Fig. 6 Speed-up ratio on the shared premises model on the shared-memory parallel computer.

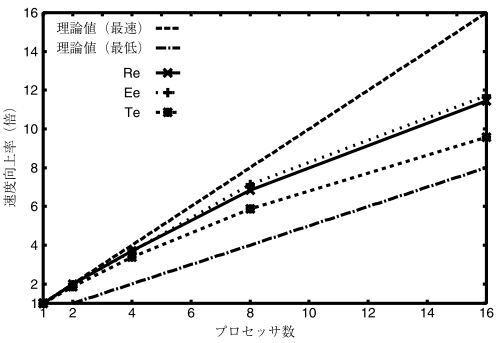

図 8 前提共有モデルにおける PC クラスタ上での実装の速度向上率
Fig. 8 Speed-up ratio on the shared-memory premises model on the cluster of PCs.

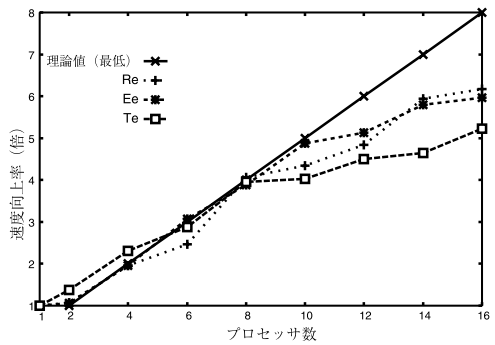

図 7 前提分散モデルにおける PC クラスタ上での実装の速度向上率
Fig. 7 Speed-up ratio on the distributed premises model on the cluster of PCs.

PC クラスタ上での前提分散モデルの実装における速度向上率が，前提共有モデルの実装における速度向上率に及ばない理由を調べるために，PC クラスタ上での前提共有モデルの実装と前提分散モデルの実装の平均待機時間を調べた．**表 2** は，PC クラスタ上での 2 種類のモデルの実装における平均待機時間を示している．表 2 から，前提共有モデルと前提分散モデルの同じプロセッサ台数を利用して実行した場合の平均待機時間は，多くの場合，前提分散モデルの平均待機時間の方が長くなっていることが分かる．

表 3 は，PC クラスタ上での 2 種類のモデルの実装における平均処理時間を表している．これは，各スレーブの実行時間から待機時間を除き，各スレーブが自動前向き演繹の処理のみに費やした時間の平均である．この表から前提を分散することにより，単に待機時間が増加しているのではなく，スレーブ処理の処理時間自体が増加していることが分かる．

表 4 と **表 5** は PC クラスタ上での前提共有モデルと前提分散モデルの実装における Re を前提とした場合の各ループの実行時間を表している．表中の L はループを表し，s は秒を表す．また表中の数字はすべて小数点 1 位以下を切捨てた値である．まず，ループの説明を以下に記す．(1) 前提の組合せのすべてと推論規則のすべてから導出できるすべての結論を導出する．(2) 導出された結論の中から前提や他の結論に対して重複しているものを取り除く．(3) 終了条件の判定を行う．(3a) 終了条件を満たしていないのであれば，今回導出された結論を前提集合に加え，(1) に戻

表 2 PC クラスタ上での 2 種類のモデルにおける実装の平均待機時間
Table 2 The average stand-by time on two models of the cluster of PCs.

論理体系		2 プロセッサ	4 プロセッサ	8 プロセッサ	16 プロセッサ
Te	前提共有モデル	15 s	16 s	16 s	16 s
	前提分散モデル	11 s	26 s	26 s	33 s
Ee	前提共有モデル	37 s	59 s	39 s	49 s
	前提分散モデル	66 s	107 s	119 s	150 s
Re	前提共有モデル	130 s	408 s	377 s	363 s
	前提分散モデル	497 s	758 s	609 s	1,010 s

表 3 PC クラスタ上での 2 種類のモデルにおける実装の平均処理時間
Table 3 The average processing time on two models of the cluster of PCs.

論理体系		2 プロセッサ	4 プロセッサ	8 プロセッサ	16 プロセッサ
Te	前提共有モデル	207 s	105 s	55 s	28 s
	前提分散モデル	297 s	157 s	81 s	48 s
Ee	前提共有モデル	1,041 s	512 s	256s	130 s
	前提分散モデル	1,972 s	996 s	435 s	211 s
Re	前提共有モデル	7,067 s	3,477 s	1,729 s	860 s
	前提分散モデル	12,928 s	6,603 s	2,932 s	1,323 s

表 4 前提共有モデルの PC クラスタ上での実装における各ループの実行時間と前提の数の変化
Table 4 The processing time in each loop on the shared premises model on cluster of PCs.

プロセッサ数	1L	2L	3L	4L	5L	6L	7L	8L	9L	10L	11L	12L	13L
1 プロセッサ	0 s	0 s	1 s	11 s	357 s	2,883 s	5,236 s	3,086 s	985 s	431 s	61 s	7 s	0 s
2 プロセッサ	0 s	0 s	0 s	6 s	187 s	1,475 s	2,643 s	1,599 s	526 s	222 s	33 s	5 s	0 s
4 プロセッサ	0 s	0 s	0 s	3 s	96 s	731 s	1,475 s	847 s	276 s	121 s	19 s	3 s	0 s
8 プロセッサ	0 s	0 s	0 s	2 s	49 s	413 s	778 s	466 s	157 s	62 s	12 s	2 s	0 s
16 プロセッサ	0 s	0 s	0 s	2 s	26 s	218 s	458 s	306 s	81 s	34 s	7 s	4 s	1 s
前提の数	7	25	110	582	3,111	11,314	14,034	7,725	2,544	1,083	159	28	1

表 5 前提分散モデルの PC クラスタ上での実装における各ループの実行時間と前提の数の変化
Table 5 The processing time in each loop on the distributed premises model on cluster of PCs.

プロセッサ数	1L	2L	3L	4L	5L	6L	7L	8L	9L	10L	11L	12L	13L
1 プロセッサ	0 s	0 s	0 s	12 s	391 s	3,178 s	5,736 s	3,355 s	1,064 s	468 s	65 s	9 s	0 s
2 プロセッサ	0 s	0 s	0 s	9 s	305 s	3,332 s	5,640 s	2,829 s	846 s	325 s	38 s	7 s	0 s
4 プロセッサ	0 s	0 s	0 s	5 s	166 s	1,669 s	3,110 s	1,556 s	485 s	190 s	21 s	4 s	1 s
8 プロセッサ	0 s	0 s	1 s	3 s	92 s	881 s	1,512 s	713 s	210 s	82 s	11 s	2 s	0 s
16 プロセッサ	0 s	1 s	1 s	2 s	73 s	587 s	962 s	477 s	140 s	53 s	9 s	1 s	0 s
前提の数	7	25	110	582	3,111	11,314	14,034	7,725	2,544	1,083	159	28	1

る（3b）終了条件を満たしているならば終了する．上記の（1）から（3a）または（3b）までの流れをループとよび，(1）から（3a）または（3b）を順番どおり 1 回行うことを 1 ループと呼ぶ．上記の流れから自動前向き演繹は，終了条件を判定された後，新しい前提を入力され，処理を繰り返しているとみることができる．

この表から前向き演繹エンジンの各ループの実行時間が，各ループにおいて処理する前提の数にかかわらずスレーブの数の増加に対して理論値と同じ傾向を示し減少していることが分かる．

以上の結果から，今回設計した前向き演繹の処理の並列計算モデルに基づいた EnCal の共有メモリ型並列計算機上での実装と PC クラスタ上での実装は，高速化に有効であることが分かった．双方の実装とも，プロセッサ台数が増加するごとに速度向上率は増加した．また，双方の実装とも速度向上率はプロセッサ台数に対して速度向上率は飽和していない．よって，プロセッサ台数を増加させることにより，よりいっそうの速度向上率が得られると考えられる．

5. 考　　察

本実験から，自動前向き演繹の並列計算モデルに基づいた前向き演繹エンジンの実装が，前向き演繹エンジンの高速化に有効であることが分かった．前提共有モデルに基づいた実装の速度向上率は，自動前向き演繹の並列計算モデルの理論値と同じ傾向を示した．また，各ループの実行時間も処理する前提の数にかかわらず理論値と同じ傾向を示した．前提分散モデルに基

づいた PC クラスタ上での実装の速度向上率と各ループの実行時間はともに，理論値には至らなかったが，プロセッサ台数が増加するごとに速度向上率も増加した．双方の実装ともに速度向上率はプロセッサ台数の増加に対して飽和しなかった．

前提分散モデルに基づいた PC クラスタ上での実装の速度向上率が前提分散モデルに基づいた共有メモリ型並列計算機上での実装の速度向上率に至らない理由として，同期の回数の増加と，各スレーブの前向き演繹処理では取り除けない重複した結論の増加が考えられる．3.2 節で述べたように，前提分散モデルは前提共有モデルに比べ多くの同期が必要となる．よって，前提分散モデルは，前提共有モデルより，多くの待機時間が必要になったと考えられる．また，3.2 節で述べたように前提分散モデルでは前提共有モデルに比べ，各スレーブの前向き演繹処理で取り除けない重複した結論の数が多くなる．この各スレーブの前向き演繹処理で取り除けない重複した結論との比較が前提分散モデルの速度向上率が理論値に至らないもう 1 つの理由であると考えられる．

今回は重複検査処理で線形検索を利用した．しかし，今回の自動前向き演繹の並列計算モデルは，二分検索などの他の検索方法を利用した場合でも有効であると考える．なぜなら今回設計した並列計算モデルは，線形検索に特化した並列計算モデルではなく，処理するデータを各プロセッサに静的に分散することにより実行時間を短縮するというものだからである．

また，表 4 と 表 5 の結果から，今回提案した並列計算モデルは，より多くの前提を利用した場合でも有効であると考える．なぜなら，本モデルでは，各スレーブが処理する前提の組合せの数が同じであり，また 1 スレーブあたりで処理する前提の組合せの数に関係なく各ループの処理時間は理論値と同じ傾向を示しているからである．

上記の理由から，今回の自動前向き演繹の並列計算モデルを利用した前向き演繹エンジンの高速化は，EnCal だけでなく，照合処理，導出処理，重複検査処理，追加処理を行う他の前向き演繹エンジンの高速化にも有効であると考える．

6. ま と め

本論文では，前向き演繹エンジンの高速化のために，並列自動前向き演繹における前提の主記憶領域への配置のモデルを 2 つ提案し，自動前向き演繹の並列計算モデルを提案した．また，提案した並列計算モデルと前提の主記憶領域への配置のモデルに基づき，汎用前向き自動帰結演算システム EnCal の共有メモリ型並列計算機上と PC クラスタ上での実装と実験を述べ，モデルの有効性を実証した．

今後の課題として，前提分散モデルにおけるスレーブの処理時間の削減とスレーブの待機時間の削減があげられる．スレーブの処理時間の増加の原因は，前向き演繹処理では取り除けない結論と新しく導出された結論の比較の回数の増加である．よって，このような比較の回数をいかに減少させるかが重要である．

また，スレーブの待機時間を削減するために，動的負荷分散を用いた負荷の均等化が必要であると考える．負荷の均等化の方法として，マルチレベル動的負荷分散方式[10] が考えられる．マルチレベル動的負荷分散方式は，疎結合型並列計算機上での OR 並列型全解探索問題に適した動的負荷分散方式である．全解探索問題が，ある問題に対する解を探索するのに対し，前向き演繹エンジンでは，前提から導出できるすべての結論を導出するというものである．しかし，これらはある問題に対し，すべての場合について処理を行うという点において，類似している．よって，この手法が有効であると考える．

今後は，上記の 2 つの課題を解決できる自動前向き演繹の並列計算モデルを提案し，そのモデルに基づき前向き演繹エンジンを実装し，そのモデルの有効性の検証を行いたい．

謝辞 本論文の原稿に様々なご指摘をくださった査読者に深く感謝いたします．

参 考 文 献

1) Cheng, J.: Encal: An Automated Forward Deduction System for General-Purpose Entailment Calculus, *Advanced IT Tools, IFIP World Conference on IT Tools, IFIP96 — 14th World Computer Congress*, Terashima, N. and Altman, E. (Eds.), pp.507–514, Chapman & Hall (1996).
2) Ueda, K.: GUARDED HORN CALUSES, ICOT Technical Report TR-103 Institute for New Generation Computer Technology (ICOT) (1985).
3) Goto, Y., Nara, S. and Cheng, J.: Efficient Anticipatory Reasoning for Anticipatory Systems with Requirements of High Reliability and High Security, *International Journal of Computing Anticipatory Systems* Vol.14, pp.156–171 (2004).
4) Nara, S., Goto, Y., Takahashi, D. and Cheng, J.: Parallel Forward Deduction System for General-Purpose Entailment Calculus on Clus-

ters of PCs, *Proc. IASTED International Conference on Networks, Parallel and Distributed Processing, and Applications* (*NPDPA'02*), Tsukuba, Japan, pp.359–364 (2002).
5) OpenMP.: Simple, Portable, Scalable SMP Programming. http://www.openmp.org/
6) Omni.: RWCP OpenMP compiler project. http://www.hpcc.jp/Omni/
7) MPICH-SCore.: PC Clusters Consortium. http://www.pccluster.org
8) Anderson, A.R. and Belnap, Jr. N.D.: *Entailment: The Logic of Relevance and Necessity*, Vol.1, Princeton University Press (1975).
9) Anderson, A.R., Belnap, Jr. N.D. and Dunn, J.M.: *Entailment: The Logic of Relevance and Necessity*, Vol.2, Princeton University Press (1992).
10) 古市昌一，瀧 和男，市吉伸行：疎結合並列計算機上での OR 並列問題に適した動的負荷分散方式とその評価，ICOT Technical Report TR-517 Institute for New Generation Computer Technology (ICOT) (1989).

(平成 16 年 4 月 16 日受付)
(平成 16 年 9 月 30 日再受付)
(平成 16 年 12 月 28 日再々受付)
(平成 17 年 2 月 2 日採録)

奈良 信介

1978 年生．2002 年 3 月埼玉大学工学部情報システム工学科卒業．2004 年 3 月同大学院理工学研究科博士前期課程情報システム工学専攻修了．現在，同大学院理工学研究科博士後期課程情報数理科学専攻在学中．

後藤 祐一（正会員）

1978 年生．2001 年 3 月埼玉大学工学部情報システム工学科卒業．2003 年 3 月同大学院理工学研究科博士前期課程情報システム工学専攻修了．2005 年 3 月同大学院理工学研究科博士後期課程情報数理科学専攻修了，博士（工学）．2005 年 4 月埼玉大学工学部情報システム工学科助手．人工知能学会会員．

程 京徳（正会員）

1982 年 7 月中国清華大学計算機科学技術系卒業．1989 年 3 月九州大学大学院工学研究科博士後期課程情報工学専攻修了，工学博士．1989 年 4 月九州大学工学部情報工学科助手．1991 年 1 月同助教授．1996 年 5 月九州大学大学院システム情報科学研究科情報工学専攻教授．1999 年 4 月埼玉大学大学院理工学研究科情報数理科学専攻教授．ソフトウェア工学，知識工学，情報セキュリティ工学に関する研究に従事．日本ソフトウェア科学会，ACM，IEEE-CS，IEEE-SMC，IEEE，AAAI 各会員．