TR-202

大規模知識ベースマシンにおける単一化エンジンの評価

小黒雅巳 (NTT)

September, 1986

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03) 456-3191～5
Telex ICOT J32964

大規模知識ベースマシンにおける単一化エンジンの評価

ICOT研究報告

（財）新世代コンピュータ技術開発機構

小黒　雅己

NTT電気通信研究所　情報通信処理研究所

期間：昭和61年5月12日～8月8日

目次

１．はじめに

本報告書は、筆者が上記期間中にＩＣＯＴ第３研究室で行った研修内容に関するものである。

第３研究室では、関係データベースマシンを拡張した関係型知識ベースマシンの開発を行っている［伊藤86a ］［伊藤86b ］。本マシンは、知識を変数を含み論理的構造体である項で表現することを仮定し、関係の属性値を定数だけでなく、項の集合に拡張し、項集合を２次記憶に格納する。このため、２次記憶に格納された項の検索の高速化が必要となる。項の検索に対しては、従来の関係演算に単一化の概念を加えたＲＢＵ演算(Retrieval By Unification)［横田85］を用いる。さらに、ＲＢＵ演算を高速に実行する専用プロセッサを複数用いて、検索の高速化を図る。この専用プロセッサを単一化エンジンと呼ぶ。

本稿は、第２章で、単一化エンジンの構成方式を述べ、第３章で、ソフトウェア・シミュレータにより、網羅的データを用いてエンジンの基礎的な性能について述べる。また、第４章では、評価結果を基に、処理高速化の方式を提案し、その性能を評価する。最後に、現在想定されている例を用いて、単一化エンジンの性能を確認する。

２．単一化エンジンの構成方式

単一化エンジン(ＵＥ:Unification Engine)の構成を図2.1 に示す。エンジンの入力は、項のストリング表現、出力は、単一化結合に成功した入力属性タプルのストリング表現である。ＵＥは、ソータ （ ＳＵ:Sort Unit）、ペア生成部(ＰＧＵ:Pair Generation Unit)、及び単一化処理部(ＵＮＵ:Unification Unit)に分類される。これら各部、及び各構成要素の機能について以下に述べる。

タプル記憶部：入力したタプルの格納場所

前処理部　：図2.2 に示すように、知識ベースkb1 とkb2 の間で、ジョイン対象属性をＢ、Ｃとして行う単一化結合では、タプルのＢ属性値、Ｃ属性値のみをソータへ送り、出力属性値Ａ、Ｄをタプル記憶部へ送る。

ソートセル　：2way-merge-sort 法を用い、項をオーダリングにより、ジェネラリティ順に並べ変える。この場合、入力される項は可変長の文字列で表現される。また、IRIE化により、ソートセルの流量を減らす［森田86］。

ペア生成部　：オーダリングにより整列された２つの項の記号列を受け取り、いずれかの項の変数の前までの２項の記号列の一致性をチェックする。一致した項組を単一化成功の可能性のあるペアとして、単一化処理部へ送る。一方、ペアに対応する入力属性タプル（図２．２）の、タプル記憶部上の格納番地を、出力生成部へ送る。

出力生成部　：タプル記憶部を参照して、単一化結合の結果として出力する入力属性タプルの並びを生成する。

単一化セル　：記号列の食い違いを検出し、ペアの１つの変数に対する置換を求め、置

換えセルへ送る。１つの良い違いの検出を、単一化セル１つで行い、残りの処理は、次段セル以降で行うパイプラインである。

置換えセル　　：単一化セルで求まったひとつの変数に対する置換を、ペアに対応する入力属性タプルに適用する。置換が求まればすぐに適用が行えるため、この単一化処理方式を置換随時適用方式（ＳＡＭ：Sabstitution Apply Method ）と呼ぶ。

後処理部　　：データを標準出力形式に変換する。

全体で見ると、これら各部がパイプラインを行っている。

以上の構成をＣ言語を用いて作成されたプログラムを使用してシミュレートし、各セル間のデータの流れる時間を測定することにより単一化Ｔンジンの性能を評価する。なお、本評価では、１回のデータに対する操作(read/write)に要する時間を１クロックとする。

３．単一化エンジンの性能

基礎的なエンジン性能を評価するために、偏りのない網羅的な項集合を用いて、ＲＢＵ演算の中で最も処理が重い単一化結合演算時の性能を評価する。但し、今回は、簡単のために、入力属性タプルとジョイン属性を同一にして測定を行っていく。

３．１　ＵＥの処理概要

はじめに、ＵＥの処理を決める要因となると考えられる入力量とＵＥ処理クロックの関係を見る。図３．１は、入力量／処理クロック関係，及び入力量／出力量関係を示す。図３．１では、入力量／処理クロック関係は、単調増加にならない。さらに、入出力の関係にも同様のことが言える。しかし、入力量に対する出力量の変化と処理クロックの変化には、類似した傾向がみられる。このため、ＵＥの処理は、入力量よりもむしろ出力量と関係が密であると予想される。さて、ＵＥの出力とは、ジョイン属性値の単一化成功組の入力属性タプルである。このため，出力量は、入力量だけでなく、入力に含まれる単一化成功組の量にもよる。さらに、単一化成功組量は、以下のような条件の元で入力量を変化させることにより、入力量と関係を持って変化する。

（１）木構造を一定にし、図３．２に示す各ノードに現われる関数子の種類数を変える。

①根ノードの関数子のみの種類数変化(PARA1)

②中間ノードの関数子のみの種類数変化(PARA2)

③葉ノードの関数子のみの種類数変化(PARA3)

（２）木構造を代える。

根、葉からなる木構造において、根ノードの関数子がとる引数の数を変化(PARA4)

（これは、リストに相当）

本評価では、この４つのパラメータ別に測定を行う。図３．３は、PARA1 ～ PARA3 における入力量／処理クロック関係を示す。上記PARA1 ～PARA4 は、ＵＥ処理を決める一要因であることが分かる。

次に、各部の処理の全体処理への影響を見る。単一化エンジンの主な処理ブロックは、ＳＵ、ＰＧＵ、ＵＮＵであり、全体でパイプラインを行う。このため、処理負荷が一番多い部分が全体の処理を決めると考えられる。図３．４に、各パラメータ毎の各部の処理クロックとエンジンの処理クロックの関係を示す。図３．４より、すべての場合で、エンジンの処理クロックが、ＵＮＵの処理クロックにほぼ一致していることが分かる。これは、ＳＵでの項の整列やペアの生成に要する処理は、ＵＮＵでの単一化の処理より軽いからである。一方、ＵＮＵの処理を決める要因である入力量は、ＰＧＵの出力量である。ＰＧＵの出力量は、ＰＧＵの単一化失敗項組の削減機能（以降、候補削減能力と呼ぶ）で定まる。

以上より、エンジン性能は、ＰＧＵの候補削減能力とＵＮＵの単一化処理能力に左右される。以下、各部の性能を評価する。

３．２　ソート処理

図３．４から明らかなように、ソート処理は、パラメータによらず、入力量にのみ影響を受ける。このため、前述PARA1 ～PARA4 のパラメータは無視して、ソータの評価を行う。図３．４で、入力量と処理クロックが完全に比例関係にならないのは、Ｔｒｉｅ化の効果である。Ｔｒｉｅ化の効果が大きいほど、セルに流れる量は少なくなり、処理が軽くなる。網羅的なデータでは、一般的に、関数子の種類の数が増えると、Ｔｒｉｅ化の効果が大きくなる。図３．５に、Ｔｒｉｅ化の効果例を示す。Ｔｒｉｅ化を行うと、ソートセルの後段に行くに従い処理が軽くなっている。2way-merge-sort では、後段に行くに従い、メモリの容量が増える（2^n タプル分　ｎ：段数）ため、Ｔｒｉｅ化を行わないと、図３．５に示すように、項列をメモリにセットするまでの時間が多くかかる。

次に、入力時の項の整列状況と処理時間の関係を見る。このため、次の３つの入力データを考えた。

(1) あらかじめ完全にソートされたデータ

(2) ある程度ソートが成されているデータ

(3) まったくランダムなデータ

結果を表３．１に示す。ランダムなデータであっても整列したデータであっても処理クロックに大きな差は生じない。これは、2way-merge-sort のソーティングディレイがＮ（Ｎ：ソータ段数）クロックであるためである。このため、データの整列によらない安定したソート処理が可能である。

以上より、ソート処理は、可変長であっても、Ｏ（ｎ）（ｎは入力量）で処理を行うことが可能であり、Ｔｒｉｅ化による処理の高速化を実証した。

３．３　ペア生成処理

図３．４では、ペア生成部の処理は、パラメータPARA1 ～PARA4 により傾向が異なっているため、別々に評価を行う。

３．３．１　入出力関係

PARA1 ～PARA4 で、入力項数に対する出力項数の変化を図３．６及び図３．７に示す。根ノード関数子種類数の変化に対するペア生成では、入出力項関係は比例関係となる。これは、根ノードに現われる関数子の数は１個であり、しかも項の系列化により一番先頭にあらわれるために、変数の前までの記号列の一致性を検査するペア生成のアルゴリズムは、効果的であり、単一化失敗項組を多く落とすことができるからである。一方、中間・葉ノードの違いに対するペア生成では、入出力関係は比例しない。図３．８に示す位置に変数（Ｘ）が発生すると、相手項の点線枠内のチェックは行わず、すべてペアと判定する。一方、中間・葉ノードの関数子の種類数が増加すると、点線枠内が異なる項の数が増加する。この増加率は関数子の種類数と比例関係にならない。しかも、点線枠内はチェックを行わず、すべて出力するからである。また、図３．７に示す木構造変化時の入力に対する出力の処理量は、入力項数をｎとすれば、Ｏ（n^2）となる。これは、中間・葉ノードの場合と同様の理由によるが、この場合上記点線枠内で異なる項が２乗のオーダで増加する。このため、根ノードの関数子が引数を多くとると、出力量はＯ（n^2）で増加していくこととなる。

３．３．２　ペア生成部の出力と処理クロック

網羅的データを使用した例では、ペア生成部の出力量は、入力量より大きくなる。このため、ペア生成部の出力量の処理への影響を考える。図３．９に、ペア生成部処理クロック／出力量関係を示す。図３．９から明らかなように、処理クロックは、出力量と比例関係にある。従って、出力量が入力量より十分大きければ、出力処理がペア生成部の処理時間となりペア生成の処理は出力処理に隠れる。これは、ペアを生成する処理は断続的であり一度に多数組を生成する。このため、一度に複数生成されたペアを流す間に次のペアの集合を生成してしまうからである。３．３．１と合わせて、ペア生成部の処理は、最悪Ｏ（n^2）を必要とすることが明らかである。一方、３．１に示したように、単一化処理は、ペアの生成より処理時間がかかる。このため、単一化処理部の処理が、エンジンの性能が低下する最大の原因となっている。

３．３．３　候補削減能力

ＰＧＵの性能として、明らかにすべき候補削減能力について評価する。表３．２にＰＧＵ出力項組数／単一化項組数特性を示す。また、表３．３には、エンジン入力に含まれる単一化失敗項組を、ＰＧＵでふるい落とした割り合いを示す（削減率）。木構造が複雑になるか、または木構造が簡単であっても関数子の種類数が多く存在する程、単一化成功項組数に対して、ＰＧＵの出力項組数が多くなる。一方、削減率を見ると、この条件は、明らかであり、このような場合に、ＰＧＵの候補削減能力が低下し、エンジン全体の性能が低下する。

３．３．４　オーダリングによる候補削減能力の違い

項のオーダリングには、left-most 方式とouter-most方式がある（図３．１０）［森田86］。表３．２は、left-most 方式の場合の候補削減能力であり、表３．４にouter-most方式の候補削減能力を示す。これは、ＰＧＵ出力項組数／単一化項組数を木構造一定の場合について示した。

表３．２と表３．４を比較すると、中間ノードの変化に対しては、outer-most方式の候補削減能力が優れ、葉ノードの変化に対しては、left-most 方式の候補削減能力が優れている。これらは、次の理由による。ＰＧＵは、記号列で表した項間の最初の変数までの一致性をチェックする。項の変数は、葉ノードにしか現われない。このため、変数より上位のノードにある関数子の、種類数が多い場合には、outer-most方式、変数より左のノードにある関数子の、種類数が多い場合にはleft-most 方式に有利であるからである。よって、中間・葉ノードにくる関数子の種類数が同じであれば、両方式は違いがない。

３．４　単一化処理部

３．４．１　パイプライン処理

ＵＮＵの各エレメントの処理は、パイプラインを行っている。このパイプラインの効果を、UNION-FINDメモリを用いて、１組の項の単一化の高速化を行った［安浦84］と処理クロックを比較して、表３．５に示す。一組の項の単一化では、本方式は処理が遅くなるが、複数組の項の単一化には、パイプラインが効果的である。

３．４．２　単一化処理部と入力量

単一化処理部の置換えセルへの入力量と処理クロックの関係を図３．１１に示す。図より、処理クロックは、置換えセルへの入力量にほぼ比例することが分かる。これは、単一化セルに流れる量（ジョイン属性単）より置換えセルを流れる量（入力属性タプル量）が多く、単一化セルの処理量は無視できるほど小さくなるからである。一方、置換えセルの入力量は、ＰＧＵの候補削減能力で決まるため、ＰＧＵの候補削減能力が、前述のように低下すると処理に時間を要することになる。ＰＧＵの候補削減能力が低下するとは、ＵＮＵ入力中に、単一化失敗項組が多く含まれている事である。このため、ＰＧＵ候補削減能力低下に対して単一化処理部を高速に実行するために、以下のような手法が有る。

①単一化処理を高速に実行できる新アルゴリズムの提案
②失敗組に対する単一化処理の省力化
③入力属性タプルへの変数に対する適用の高速化

今回は、②による高速化の検討を行った。単一化処理の高速化の考察を次章で述べる。

４．単一化処理部の考察

４．１　原理

置換随時適用方式（ＳＡＭ）は、以下のような特徴を持つ（図４．１）。

①単一化セルには、単一化結合対象属性が流れ、変数に対する置換を随時求める。各セルでは、項間に食い違いが検出されるまで、記号列を検査し、検出すれば、まだ検査されていない部分のみを、次段以降に送るため、各セルの入力量は、後段に行くほど減少する。

②置換えセルには、入力属性タプルが流れ、変数に対する置換の適用が単一化セルで求まるとすぐ適用を受ける。後段に行くほど、置換の適用を多く受けた入力属性タプルが流れるため、各セルの入力量は増加する。

ＳＡＭの問題点として、以下があげられる。

入力属性タプルへの置換の適用は、随時行われるため、m段目（m＜n）でペアの単一化失敗が判明したときに、m－１段目までに求まった置換の適用は既に行われており、置換えセルに無駄な量が流れていることになる。３章で述べたように、単一化処理は、置換えセルに流れる量で決まるため、無駄な量が存在することは問題である。

この問題を除去するために、図４．２に示すｍｇｕ適用方式（ＭＡＭ：Most-general-unifier Apply Method ）についての考察を行う。ＭＡＭには、以下の特徴がある。

(1) 単一化セルは、ＳＡＭと同様の機能を持つ。

(2) ｍｇｕセルでは、単一化セルで求まった変数に対する置換を集合にして流す。このとき、流れてきた置換集合の中に、置換を適用すべき変数が含まれている場合は、置換の適用を行う。例えば、流れてきた置換集合が$\theta = \{f(a)/X, Z/Y\}$であるとき、単一化セルで{ａ／Ｚ}なる置換が求まれば、Ｚに対する置換の適用を受けて、$\theta' = \{f(a)/X, a/Y, a/Z\}$が出力される。ｍｇｕセルの最終段出力は、ペアに対するｍｇｕ（最汎化作用素）である。

(3) 置換えセルでは、ペアの単一化が終わった後に、単一化成功（ｍｇｕが求まった）ペアに対してのみ、ｍｇｕの適用を行う。置換えセルには、単一化成功ペアに対する入力属性タプルしか流れないため、置換えセルの処理量に無駄な量は含まれない。

ＭＡＭは、以上のように、入力に単一化失敗ペアが多く存在する場合には、非常に、効果的であるが、以下の問題が考えられる。

・(2) により、ｍｇｕを流れる量は、後段にいくほど増加するため、ｍｇｕセルの入力量の影響を受け、遅くなる。

・(3) により、置換えセルをパイプラインで行えなくなった影響

次節でＳＡＭとＭＡＭの比較をシミュレータにより行う。

４．２　実験結果

ＳＡＭ、ＭＡＭの比較を次の条件の元で行う。

①入力されるペアの種類、数を一定にする。

②入力ペアに含まれる単一化失敗項組の数を変化させる。このときの処理クロック／出力量の関係を測定する。

測定結果を図４．３に示す。点線は、ＳＡＭで、単一化失敗項組が無い場合の処理クロック／出力量関係である。図４．３より以下が観察される。

Ⅰ．失敗項がなければ、処理クロックは、出力量による。

Ⅱ．交点Ｐの出力データ長以上では、２方式の処理クロックは、ほぼ一致する。

Ⅲ．失敗項組が多いとき、両方式とも、出力データ長によらず、処理クロックが一定になり、しかも、ＭＡＭのそれは、ＳＡＭのそれより小さい。

これらより、次のことが考えられる。

Ⅰは、単一化失敗項組が無いため、出力量は入力量（入力属性タプル長）に比例する。入力量による処理クロックは、出力量による。

Ⅱは、失敗組が少ないとき、ＵＮＵの出力量（L_0）が、他のどのセルを流れる量よりも多くなり、処理時間がL_0で決まるからである。また、失敗組が多いとき、ＳＡＦでは、L_0よりも、最終段の置換えセルの入力量、ＭＡＦでは、L_0よりも、最終段のｍｇｕセルから出るｍｇｕの量の方が多くなり、処理時間がそれぞれの量で決まる。Ⅲは、これらの量が、ＵＮＵの入力（L_i）一定の時、ほぼ一定の量になるためである。

さらに、一般的に置換えセルに流れる入力属性タプルの量は、ｍｇｕセルを流れる置換集合の量よりも大きい。

以上より、置換集合の量が入力属性タプルの量より小さいことを前提にすると、ＵＮＵ入力に失敗項が多く含まれる場合は、ＵＮＵの処理は、置換随時適用方式よりもｍｇｕ適用方式が優れる。最後に、３．１で述べた図について、単一化処理方式をＭＡＭに代えた場合の単一化処理部、及びエンジン全体の処理クロックの違いを図４．４に示す。ＵＮＵ入力中に単一化失敗項組を含む率は、30～60％程度であるため、ＭＡＭを用いたほうが処理が速くなる。

５．実際例による性能評価

前章まででは、単一化エンジンの一般的性能を導いた。この一般的性能が実際の知識ベースで扱う項集合の検索を行ったときの性能の違いがみられる点を明らかにする。本稿では、実際の項の例として、ＤＣＫＲ［田中85］で表現された知識の例を用いる。ＤＣＫＲは、ＲＢＵ演算により、知識検索処理が可能であることは［村上86］で示された。今回用いたＤＣＫＲの例は、属性を２つ持つタプル数８１の項集合であり、その一部を付録に示す。

５．１　ソート処理

ＤＣＫＲでは、ＴＲＩＥ化による処理量の減少率が、約５０％であり、網羅的データの場合とほぼ等しい。実際の知識検索処理におけるソート処理では、ＴＲＩＥ化による処理量削減が、処理高速化の有効な手法であることが得られた。

5．2　ペア生成処理

ＰＧＵの候補削減能力について評価する。候補削減能力については、ＰＧＵ出力項組数／単一化項組数特性と削減率の２つの評価法があった。実際例では、エンジンに入力された項集合中に、単一化可能項組が含まれる割り合いが、網羅的データに比べて小さい（表５．１）。このため、ＰＧＵ出力項組数／単一化項組数特性は正当な評価となり得ない。以上より、ここでは、削減率（ＰＧＵにより落とした単一化失敗項組／ＵＥ入力内の単一化失敗項組）の評価を行う。ＤＣＫＲにおける削減率は、測定データの最悪値が約９９．１％であり、削減率は、網羅的データの場合に比べて良い。

5．3　単一化処理

ここでは、前述の方式による違いを評価する。５．２で、ＤＣＫＲの例では、削減率は高いが、ＰＧＵ出力項組中に含まれる単一化失敗項組の数は多い。このため、４章の検討から、単一化処理方式としてｍｇｕ適用方式（ＭＡＭ）が、優れていることが予想される。単一化処理の性能が全体の性能を左右したため、今回は、方式による違いを単一化処理部単独でなく、検索処理に於ける性能の違いで評価する。この結果、処理クロック（検索時間）は、ＭＡＭでは、１０１４９クロック、ＳＡＭでは、１０１５５クロックとなり、方式による違いが顕著にみられなかった。これは、次のような理由による。

・単一化失敗項組数が単一化成功項組数に対して多くても、単一化成功項組数が少ないため、ペア生成部の出力数は入力数より少ない。従って、網羅的データの場合と異なり、ペア生成処理が出力処理に隠れない。このため、図５．１のように、あるペアが出力されてから、つぎのペアが出力されるまでの時間Ｔｓが大きくなり、Ｔｓ時間内に単一化処理が終わり、２つの方式の違いが見られない。

以上より、ＤＣＫＲの例では、ペア生成部のペア生成処理の時間が問題となることが分かる。

6．おわりに

単一化エンジン性能は、ＵＮＵの性能で決まる。ＰＧＵでは、項構造が複雑になると候補削減機能が低下し、単一化失敗項組が多く残るため、ＵＮＵに、置換随時適用方式を採用すると性能が低下し、エンジン全体の性能低下につながる。このため、ＵＮＵに、ｍｇｕ適用方式を採用することで、単一化失敗項組が多く含まれるデータに対するＰＧＵの候補削減能力低下を補償できた。一方実際例の性能評価で、以下のことが明らかになった。

(1) ソータでのＴＲＩＥ化表現は効果的である。

(2) ＰＧＵの候補削減能力が優れており、ＰＧＵの出力項組数は少なくなる。

(3) (2) により、単一化処理よりもペア生成処理のほうがネックとなるため、ＰＧＵのペア生成処理の高速化が必要である。

今後の課題には、ＰＧＵの処理方式の改良、ＵＮＵの処理方式として、スイッチングネ

ットワーク［森田86］の評価を行い、シミュレーション解析を反映して、単一化エンジンの詳細設計を行う。

この研修では、専用プロセッサの構成方式について、評価を通しての研究手法について学んだ。しかし、上記以外にも、研究の上で必要な知識ベース、ユニフィケーション、論理型言語、学習等の現状についても知る機会にも恵まれ、研修は非常に有益なものであった。

謝辞

研修期間中終始御指導頂いた伊藤英則第３研究室長および森田幸伯研究員に深謝致します。また、評価データをまとめるに当り御協力頂いた日科技研の白瀬勝次氏、高橋正寿氏に感謝します。さらに、熱心に議論して頂いた第３研究室の諸氏、及びVLKB会議メンバに感謝致します。なお本レポートはＩＣＯＴでの３か月間の報告であり、このような貴重な機会を与えて下さいましたＩＣＯＴならびにＮＴＴの関係者の皆様に感謝致します。さらに、研修期間中いろいろとお世話して下さいましたＩＣＯＴの皆様に感謝致します。

参考文献

［森田86］Morita,Y.,et al,"Retrieval-BY-Unification on a Relational Knowledge Base Model",Proc. 12th Int. Conf. on VLDB,pp.52-59,August,1986

［横田86］Yokota,H,et al,"A Model and Architecture for a Relational Knowledge Base",13th Int. Sym. on Computer Architecture,pp.2-9,June,1986

［伊藤86a］Itoh,H,"Resarch and development on knowledge base systems at ICOT",12th Int. Conf. on VLDB,pp.437-445,August,1986

［伊藤86b］伊藤他「大規模知識ベースマシン実験機の開発（１）」
第３３回情報処理学会全国大会(1986) 3B-1

［柴山86］柴山他，「大規模知識ベースマシン実験機の開発（２）」
第３３回情報処理学会全国大会(1986) 3B-2

［酒井86］酒井他，「大規模知識ベースマシン実験機の開発（３）」
第３３回情報処理学会全国大会(1986) 3B-3

［小黒86］小黒他，「大規模知識ベースマシン実験機の開発（４）」
第３３回情報処理学会全国大会(1986) 3B-4

［安浦86］安浦他，「論理型言語の単一化操作のためのハードウェアアルゴリズム」
EC84-67

［田中85］田中他，「Definite Clause Knowledge Representation」，
The Logic Programming Conference 85,1985

［村上86］村上他，「単一化検索言語による知識ベースソフトウェアの記述」
第３２回情報処理学会全国大会(1986) 1H-9

図表

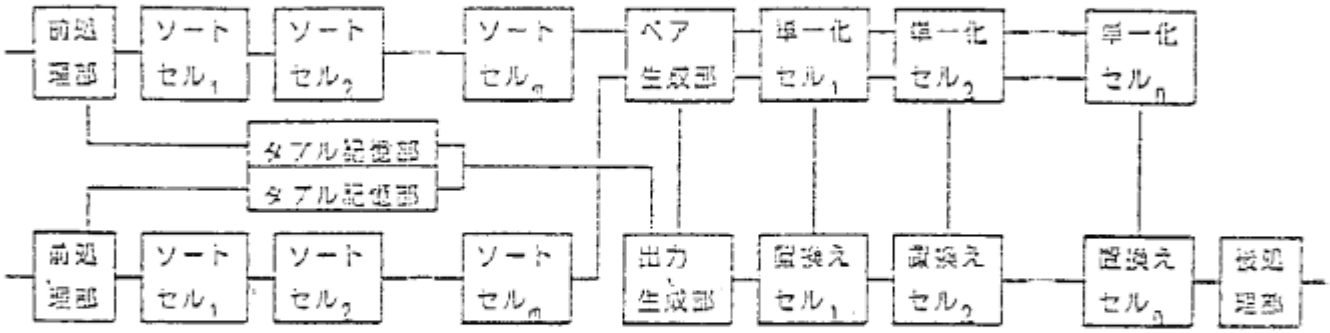


図２．１　単一化エンジンの構成

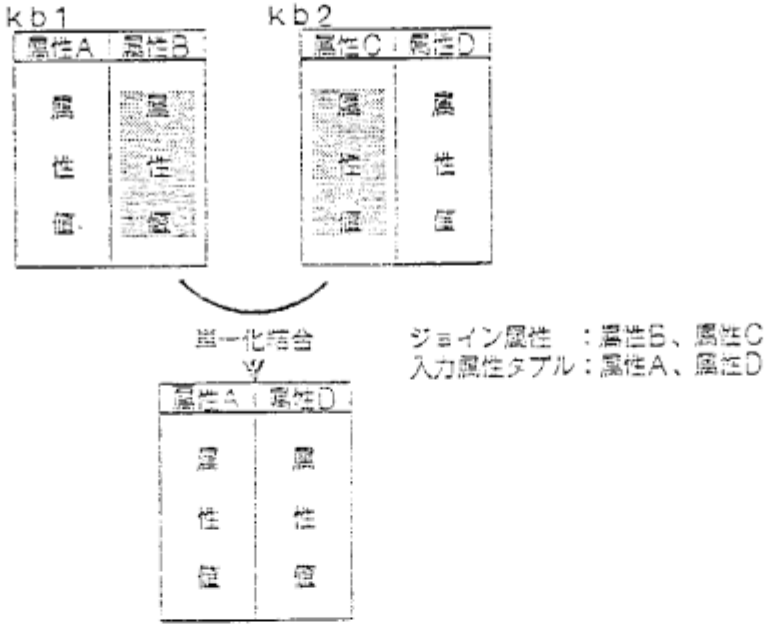


図２．２　単一化結合

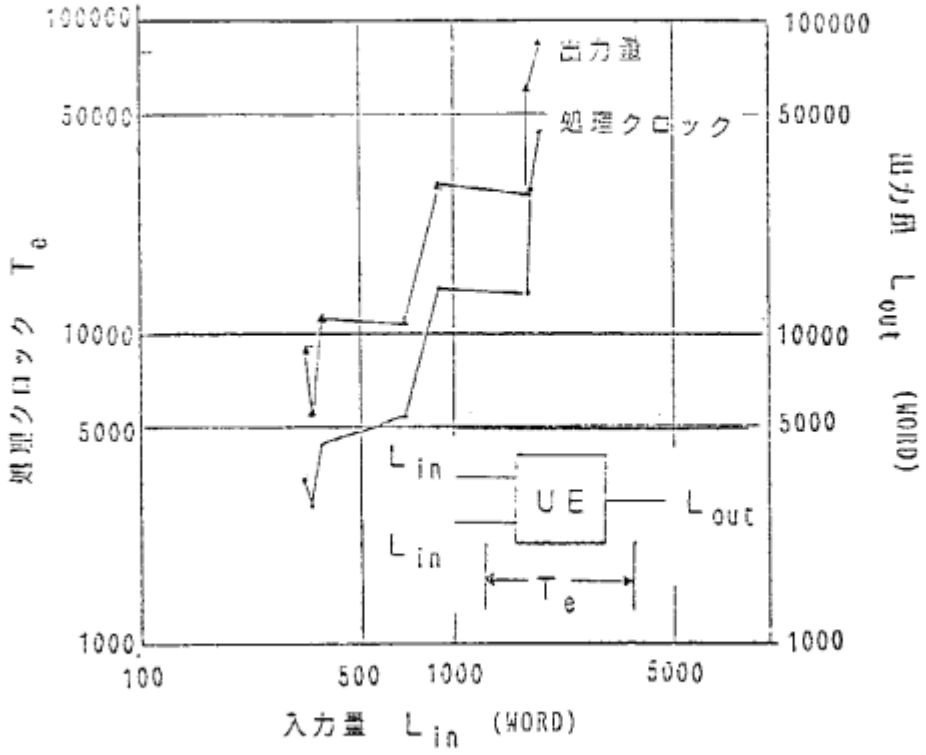


図３．１　入力量と処理クロック、出力量

図３．２　項の木構造

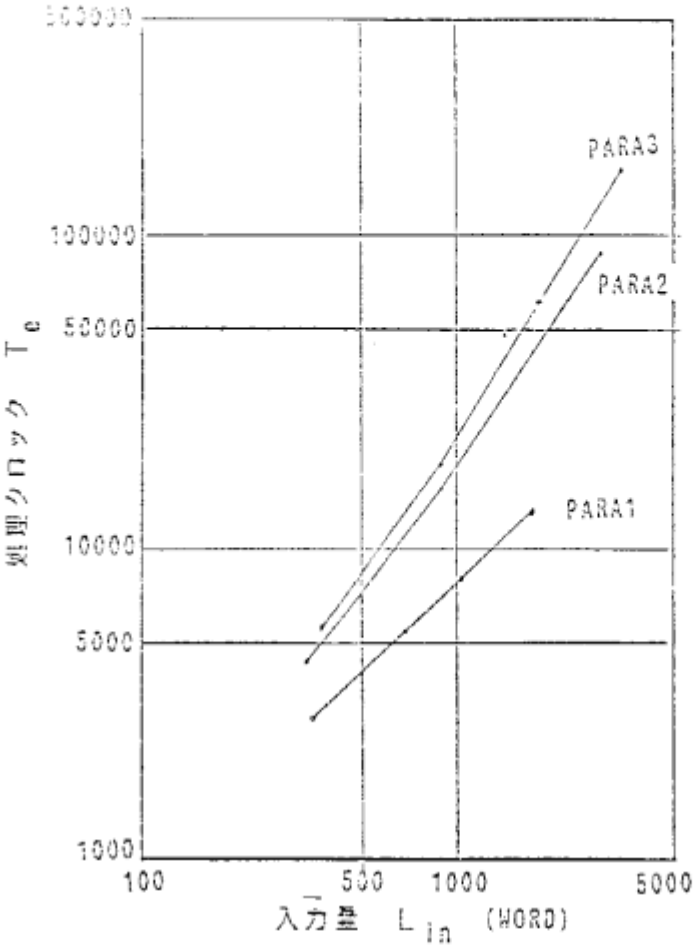
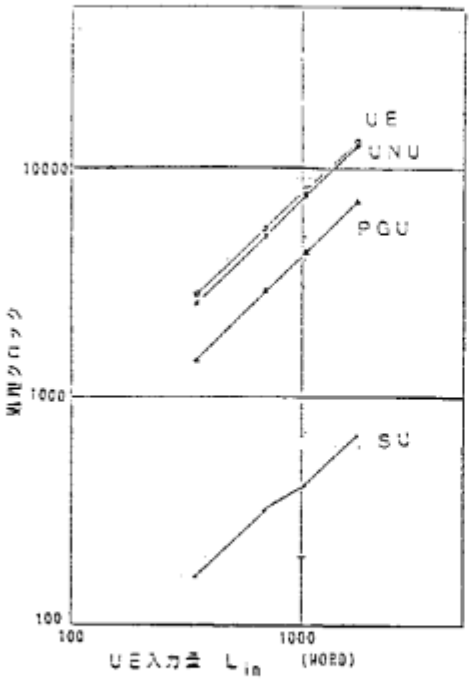
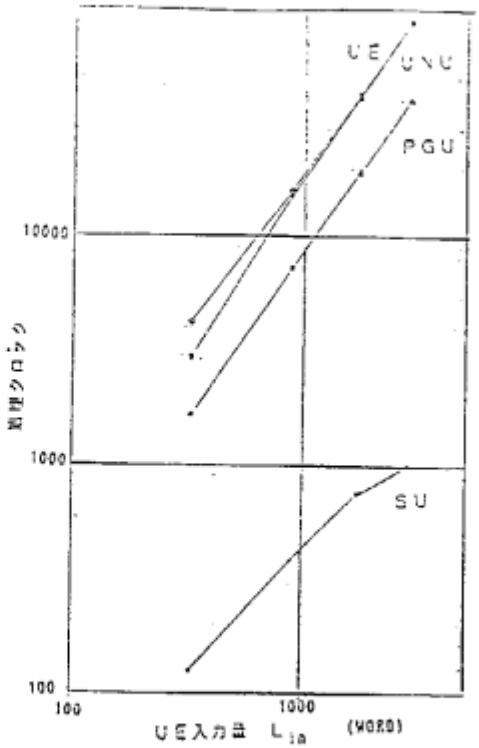


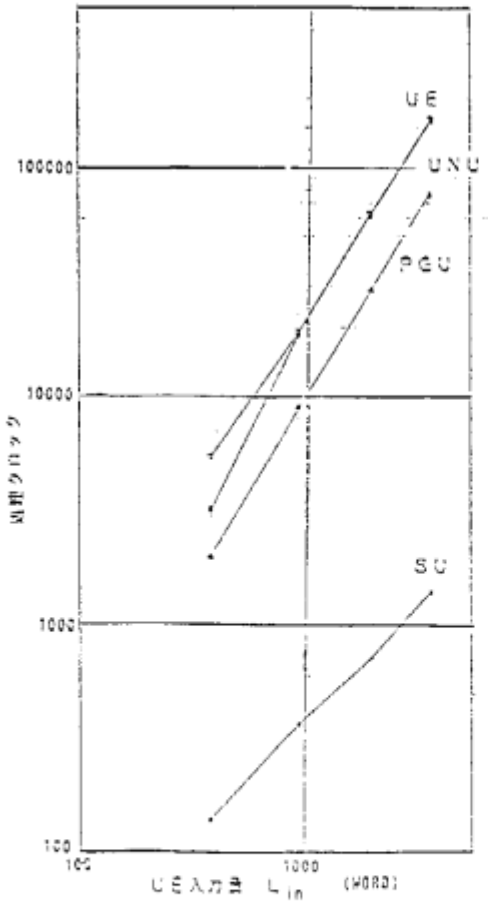
図３．３　入力量／処理クロック関係



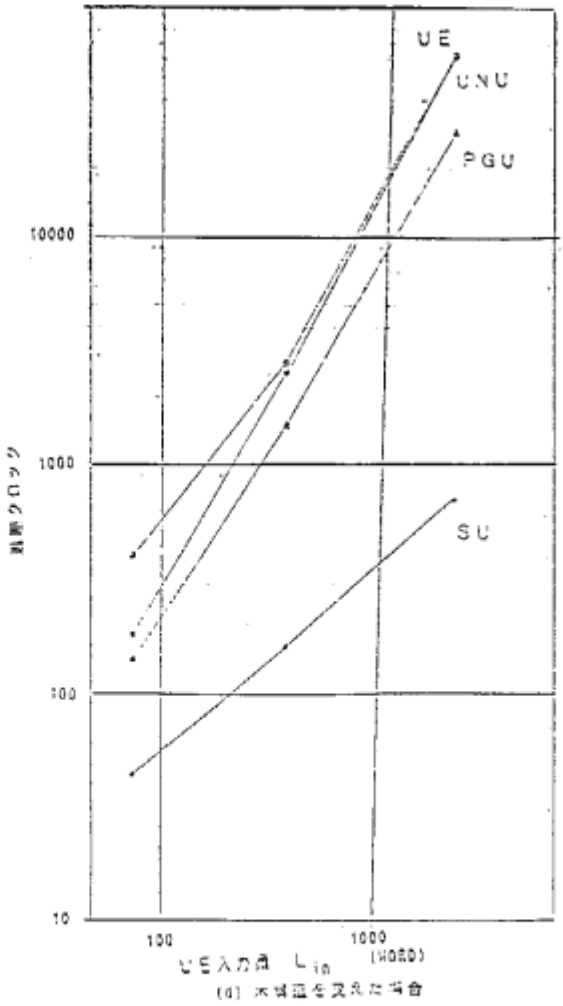
(a) 葉ノードのみの構成子の処理数を変えた場合



(b) 中間ノードのみの構成子の処理数を変えた場合



(c) 葉ノードのみの構成子の処理数を変えた場合



(d) 木構造を変えた場合

図2.14 各処理と全体の処理時間

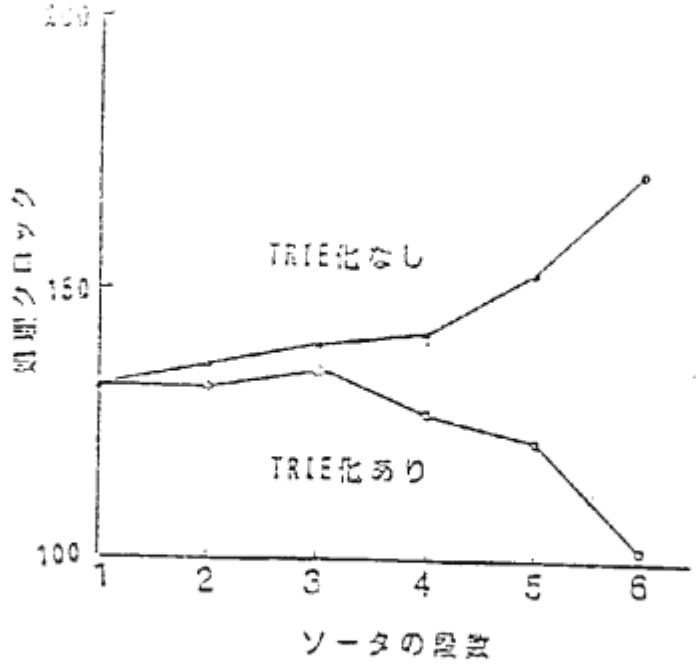


図３．５　Ｔｒｉｅ化の効果

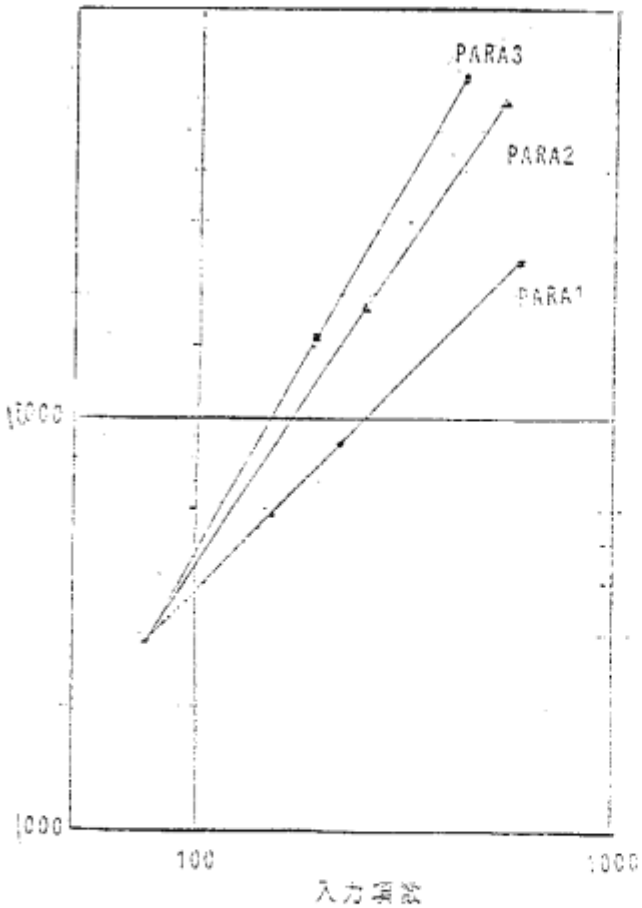


図３．６　木構造一定のＰＧＵの入出力関係

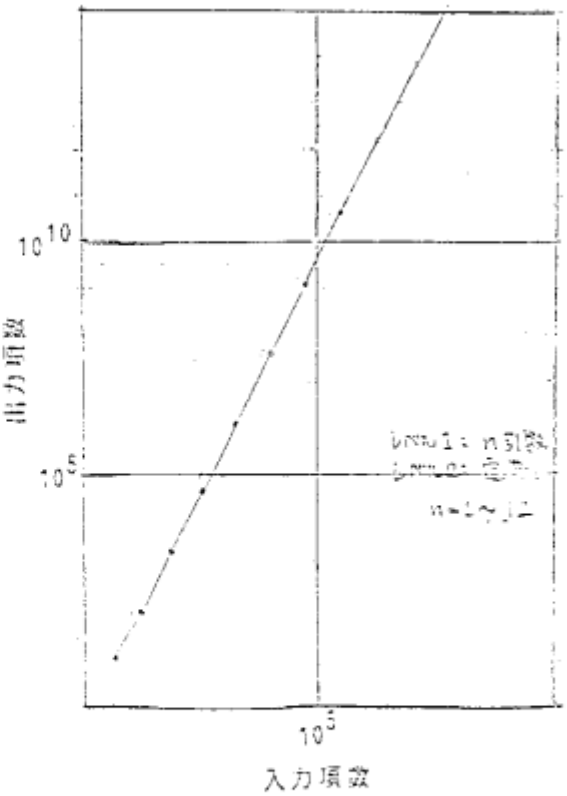


図３．７　木構造を変えたときのＰＧＵの入出力関係

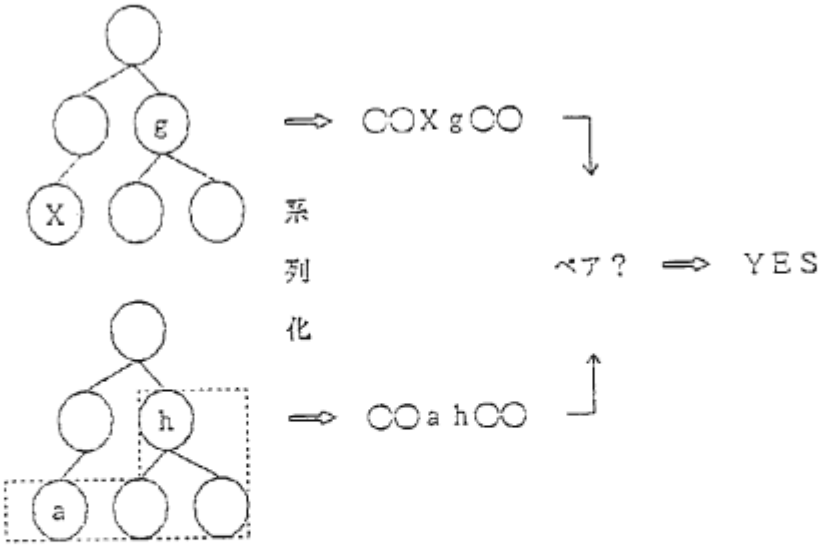


図３．８　ペア生成出力における単一化失敗項組の例

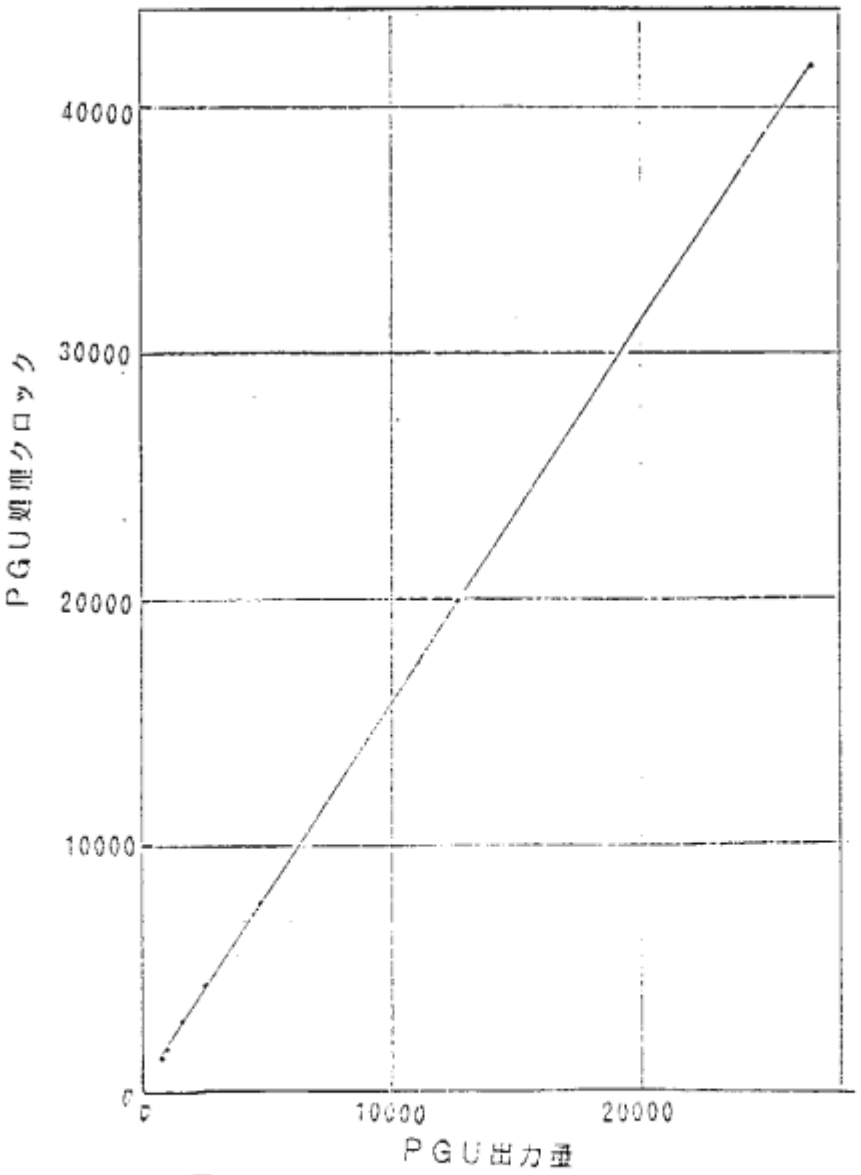
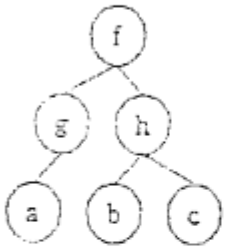


図３．９　ペア生成部処理クロック／出力量関係



Left-most ⇨ f2 g1 a0 h1 b0 c0

Outer-most⇨ f2 g1 h1 a0 b0 c0

図3．10　項の木構造のストリング表現

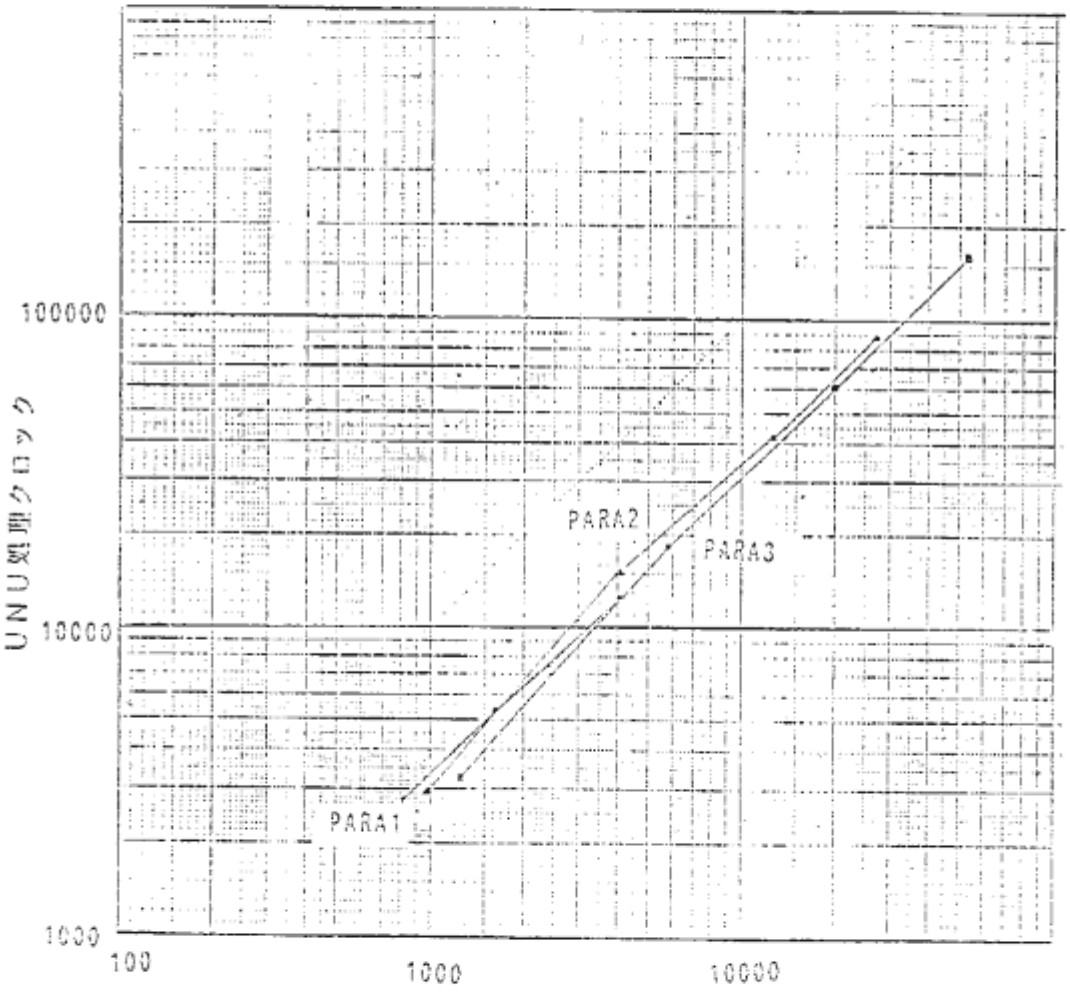


図3．11　UNU処理クロック／入力量

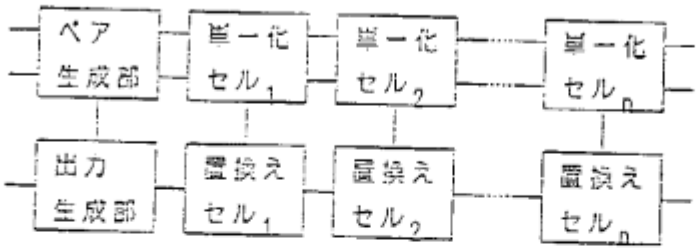


図４．１　置換適時適用方式（ＳＡＭ）

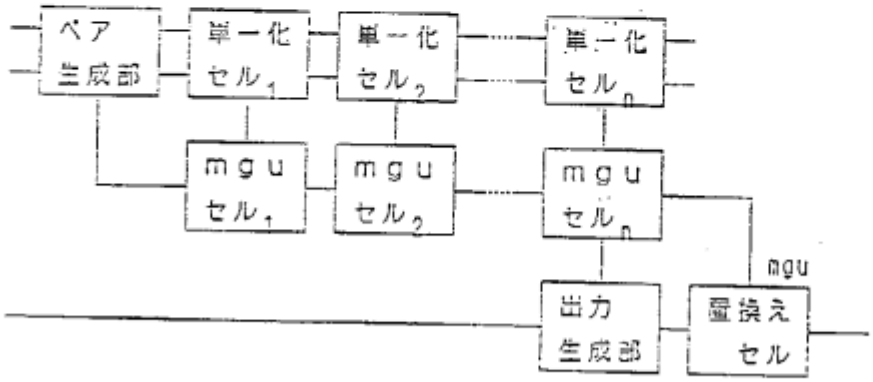


図４．２　ｍｇｕ適用方式（ＭＡＭ）

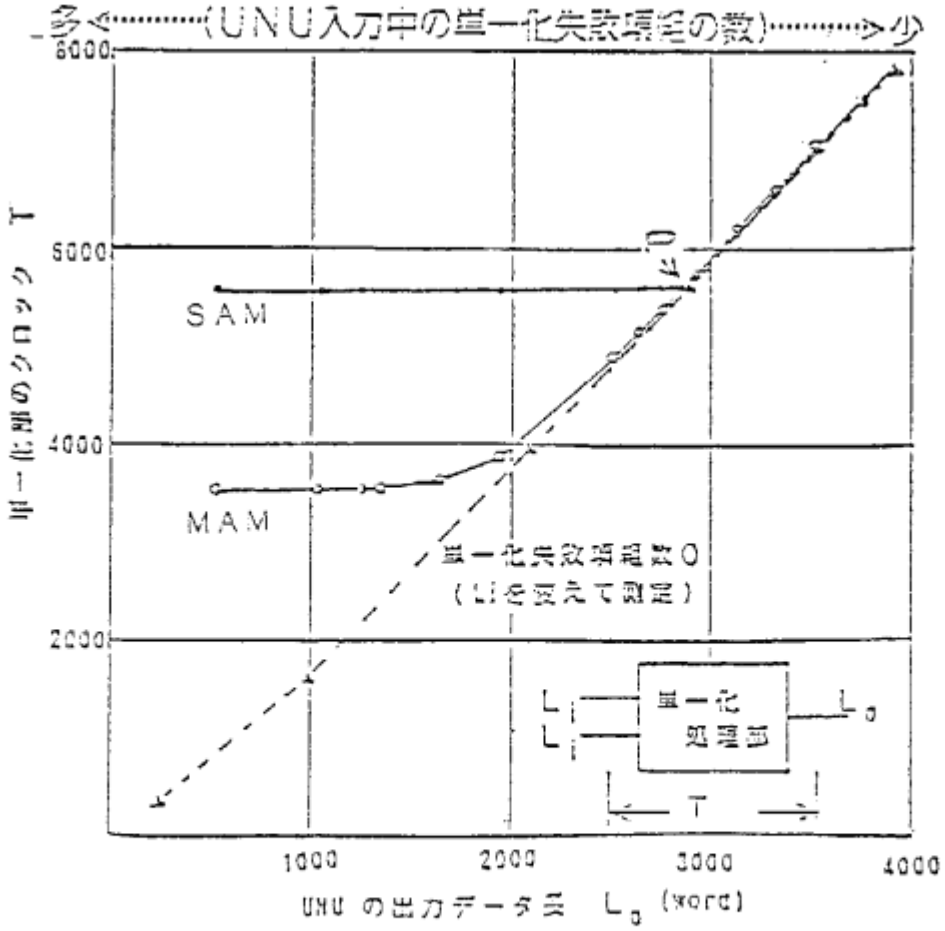
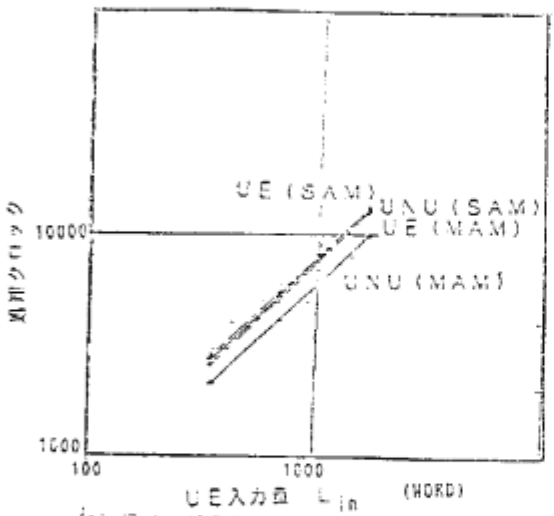
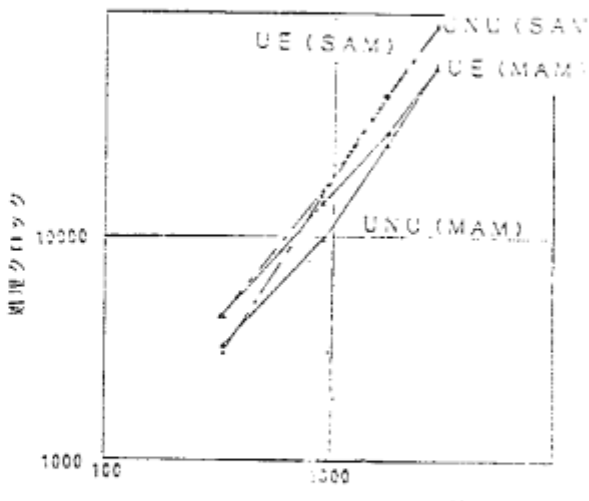


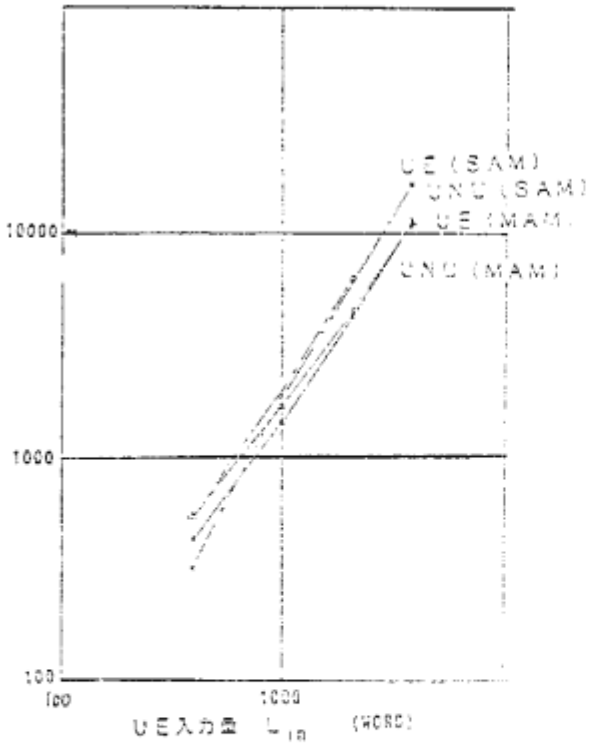
図４．３　方式の違いによる処理の違い　　（L1 ＝3265一定）



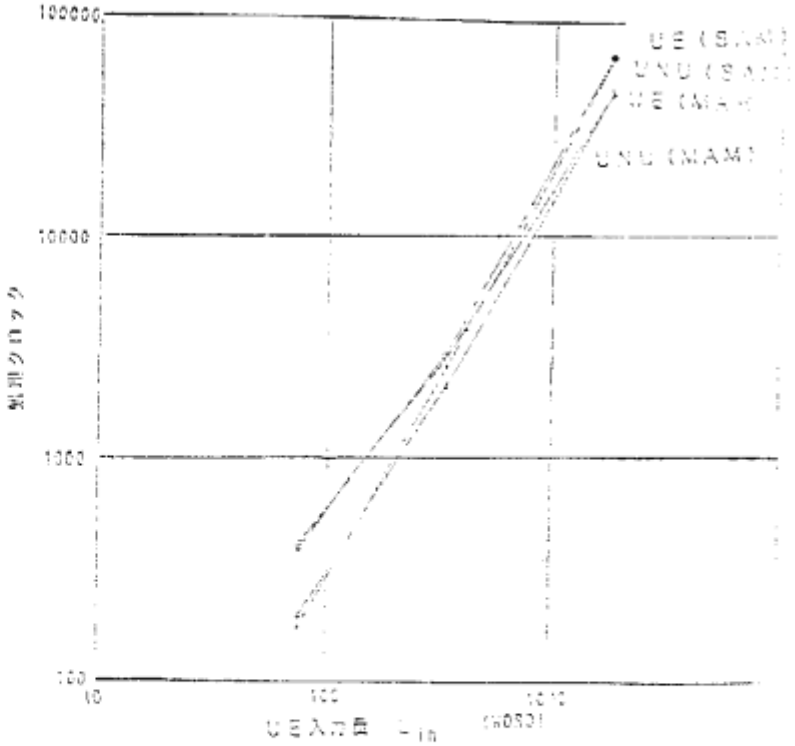
(a) 端ノードのみの関数子の種類数を変えた場合



(b) 中間ノードのみの関数子の種類数を変えた場合



(c) 葉ノードのみの関数子の種類数を変えた場合



(d) 木構造を変えた場合

図4．4　SAMとMAM

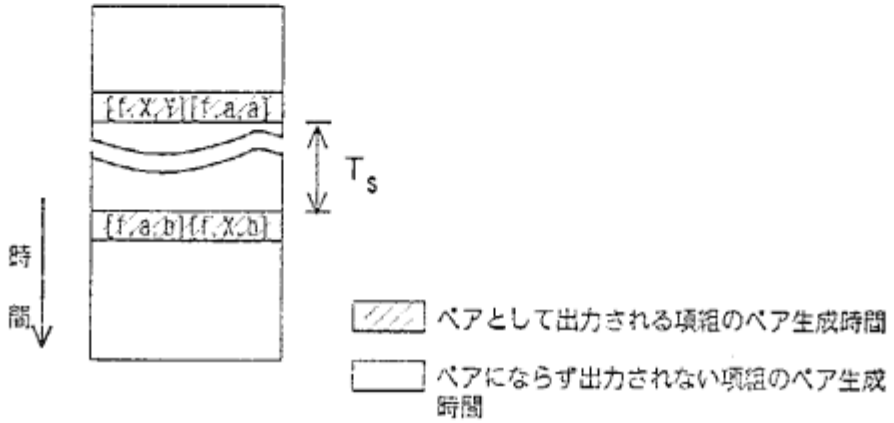


図5．1　DCKRのペア生成出力

種類数が変化するノード	関数子の種類数		
	1	2	3
浅	1.08	1.08	1.08
中間	1.08	1.86	2.63
深	1.08	1.69	2.73

(a) 木構造を一定にした場合

木構造は、深さ2
根ノードがn引数関数1種
葉ノードが定数3種と変数

n	1	2	3	4
	1.00	1.35	2.08	3.13

(b) 木構造を変化させた場合

表3．2　候補削減能力（1）
PGU出力項組数／単一化項組数

(%)

種類数が変化するノード	関数子の種類数		
	1	2	3
浅	93.0	87.5	98.5
中間	93.0	84.0	86.1
深	93.0	75.3	72.0

(a) 木構造を一定にした場合

木構造は、深さ2
根ノードがn引数関数1種
葉ノードが定数3種と変数　(%)

n	1	2	3	4
	100.0	88.6	79.4	73.2

(b) 木構造を変化させた場合

表3．3　候補削減能力（2）

種類数が変化するノード	関数子の種類数		
	1	2	3
根	1.08	1.08	1.08
中間	1.08	1.30	1.53
葉	1.08	1.71	2.83

表３．４　Outer-most方式の候補削減能力（１）
（木構造を一定にした場合）

t1=f(a1,a2,...,ak)とt2=f(X1,X2,...,Xk)の単一化

k	単一化対象項数				
	パイプライン				文献3
	1：1	5：5	20：20	50：50	1：1
1	18	9	8	6	10
2	21	12	9	8	13
10	56	37	27	26	37
25	131	76	56	−	82

単位：クロック／項組

表３．５　パイプラインの効果

	入力項数	出力項組数
候補対データ	35	311
DCKR	36	2

表５．１　UEの入出力項組数の相違

付録

```
dot(sem(c),A),B)   dot(sem(ele,A),B)
dot(sem(ele,A),B)   dot(sem(nom,A),B)
dot(sem(jewel,A),B)   dot(sem(stone,A),B)
dot(sem(jewel,A),B)   dot(sem(accessory,A),B)
dot(sem(jewel,A),B)   dot(sem(fortune,A),B)
dot(sem(diamond,A),B)   dot(sem(jewel,A),B)
dot(sem(sapphire,A),B)   dot(sem(jewel,A),B)
dot(sem(ruby,A),B)   dot(sem(jewel,A),B)
(dot(sem(A,sinkOin(water)),B)   dot(sem(A,density(heavy)),B) )
dot(sem(hare,color(white)),A)   dot(sem(currentOseason,state(winter)),A)
dot(sem(hare,color(brown)),A)   dot(sem(currentOseason,state(summer)),A)
'ot(sem(man,A),B)   dot(sem(human,A),B)
dot(sem(bird,hasOa(A)),B)   dot(sem(wing,hasOa(A)),B)
dot(sem(mike,A),B)   dot(sem(cat,A),B)
dot(sem(gonbe,A),B)   dot(sem(cat,A),B)
dot(sem(hideyoshi,A),B)   dot(sem(cat,A),B)
dot(sem(jobanni,A),B)   dot(sem(cat,A),B)
dot(sem(juliano,A),B)   dot(sem(cat,A),B)
dot(sem(kampaneria,A),B)   dot(sem(cat,A),B)
dot(sem(cat,A),B)   dot(sem(animal,A),B)
dot(sem(car,A),B)   dot(sem(trafic,A),B)
dot(sem(bmw,A),B)   dot(sem(car,A),B)
dot(sem(four0st0engine,A),B)   dot(sem(engine,A),B)
 ,t(sem(pc,A),B)   dot(sem(computer,A),B)
dot(sem(bit0map0terminal,A),B)   dot(sem(terminal,A),B)
dot(sem(computer,hasOa(A)),B)   dot(sem(terminal,hasOa(A)),B)
dot(sem(computer,hasOa(A)),B)   dot(sem(terminal,isOa(A)),B)
dot(sem(pc,hasOa(A)),B)   dot(sem(bit0map0terminal,hasOa(A)),B)
dot(sem(pc,hasOa(A)),B)   dot(sem(bit0map0terminal,isOa(A)),B)
dot(sem(bit0map0terminal,hasOa(A)),B)   dot(sem(bit0map0display,hasOa(A)),B)
dot(sem(bit0map0terminal,hasOa(A)),B)   dot(sem(bit0map0display,isOa(A)),B)
dot(sem(keyboard,A),B)   dot(sem(input0device,A),B)
dot(sem(terminal,hasOa(A)),B)   dot(sem(keyboard,hasOa(A)),B)
dot(sem(terminal,hasOa(A)),B)   dot(sem(keyboard,isOa(A)),B)
dot(sem(keyboard,hasOa(A)),B)   dot(sem(key,hasOa(A)),B)
dot(sem(keyboard,hasOa(A)),B)   dot(sem(key,isOa(A)),B)
```

```
dot(sem(sky,color(grey)),A)   A
dot(sem(moon,be(dim)),A)   A
dot(sem(accessory,looks(beautiful)),A)   A
dot(sem(accessory,on(ring)),A)   A
dot(sem(accessory,on(necklace)),A)   A
dot(sem(stone,density(heavy)),A)   A
dot(sem(stone,hardness(high)),A)   A
dot(sem(fortune,price(expensive)),A)   A
dot(sem(fortune,in0the(safe)),A)   A
dot(sem(diamond,color(clear)),A)   A
dot(sem(ruby,color(red)),A)   A
dot(sem(sapphire,color(blue)),A)   A
dot(sem(pearl,color(white)),A)   A
d  (sem(current0moon,state(not0fullmoon)),A)   A
dot(sem(wolf,face(shaggy)),A)   A
dot(sem(wolf,attacks(lady)),A)   A
dot(sem(wolf,attacks(children)),A)   A
dot(sem(human,face(not0shaggy)),A)   A
dot(sem(man,attacks(lady)),A)   A
dot(sem(birds,can0fly(yes)),A)   A
dot(sem(bat,can0fly(yes)),A)   A
dot(sem(tobiuo,can0fly(yes)),A)   A
dot(sem(airplane,can0fly(yes)),A)   A
dot(sem(penguin,can0fly(no)),A)   A
d  (sem(bird,can0fly(yes)),A)   A
dot(sem(bird,has0a(wing)),A)   A
dot(sem(cat,likes(milk)),A)   A
dot(sem(cat,likes(fish)),A)   A
dot(sem(cat,likes(katatsu)),A)   A
dot(sem(juliano,color(blown)),A)   A
dot(sem(gonbe,color(black)),A)   A
dot(sem(hideyoshi,color(white)),A)   A
dot(sem(josanki,color(stripe)),A)   A
dot(sem(car,has0a(wheel)),A)   A
dot(sem(wheel,has0a(tire)),A)   A
dot(sem(car,has0a(four0st0engine)),A)   A
```

```
dot(sem(fourCatEngine,hasCa(valve)),A)    A
dot(sem(engine,hasCa(piston)),A)    A
dot(sem(piston,hasCa(ring)),A)    A
dot(sem(A,isCa(A)),B)    B
dot(sem(computer,hasCa(terminal)),A)    A
dot(sem(osi,hasCa(bitMapCterminal)),A)    A
dot(sem(bitMapCterminal,hasCa(bitMapCdisplay)),A)    A
dot(sem(terminal,hasCa(keyboard)),A)    A
dot(sem(keyboard,hasCa(key)),A)    A
```

ゴールのリレーション	⑥
dot(sem(cl,A),nil)　dot(sem(ele,A),nil)	