自然言語処理 57－2
（１９８６．９．１９）

ロジックプログラミングをベースにした自然言語解析システムの比較

杉村領一，奥西稔幸，松本裕治（ＩＣＯＴ），田村直良，上脇正，田中穂積（東工大・工），
清野正樹（松下電器）

近年、ロジックプログラミングをベースにした自然言語処理システムの研究が盛んに行われているが、それぞれのシステムの比較検討作業が必要な時期にきていると考えられる。本論文ではその第一段階として、研究開発の進んだ構文解析システムＢＵＰ、ＬａｎｇＬＡＢ（ＧＡＬＯＰ）、ＳＡＸをとりあげ、機能面および定量的な速度・メモリ効率の比較を行った。その結果、機能面では東工大で開発されたＬａｎｇＬＡＢが文法開発などで快適な環境を提供していることがわかった。また英語の文法を用いた速度の比較ではＬａｎｇＬＡＢがＢＵＰに比べ３－３０倍速くなっていることがわかった。ＳＡＸはＬａｎｇＬＡＢに比べコンパイル時で、５－１０倍程度速いことがわかった。

Comparison of Logic Programming based Natural Language Parsing Systems

Ryouichi Sugimura, Toshiyuki Okunishi, Yuji Matsumoto（ＩＣＯＴ），
Naoyoshi Tamura, Tadashi Kamiwaki, Hozumi Tanaka (Tokyo Institute of Technology)，
Masaki Kiyono (Matsushita Electric Industrial Co.,Ltd)

This paper compares practical natural language parsing systems that are based on logic programming. The systems we have selected are LangLAB, GALOP, and SAX, of which the former two are successors of BUP. The comparison was mainly done in two aspects. One is on the facility and the environment of the systems and the other is on the parsing time and memory space needed in analyzing sample English sentences. LangLAB has the most advanced facilities and cosy environment for developing natural language grammars. It also offers the grammatical formalism that can express left extrapositions. The parsing time was compared among BUP, LangLAB and SAX, since GALOP has been improved by the same technique used in LangLAB and there is no major difference in the time complexity of these two systems. LangLAB is from 3 to 30 times faster than BUP both in compiled and interpreted time. SAX is from 5 to 10 times as efficient as LangLAB when they are compiled while the efficiency is comparable when they are interpreted.

１．はじめに

ロジックプログラミングをベースとした構文解析システムの比較について報告する。近年、ロジックプログラムの枠組みと自然言語処理との整合性の良さが認識されるに伴い、ロジックプログラムをベースとした自然言語処理システムの研究開発が盛んに行われてきており、種々の解析ツールが提案されている。

これらの研究は基本的には文脈自由文法を基本にしているが、研究範囲は処理の対象とする言語に依存しない研究（構文解析アルゴリズム等）から、対象とする言語に特有の研究（日本語文法、英語文法、形態素）にまで幅広くまたがっている。

例えば、構文解析用のツールの場合でも数多くのものが提案されているが、各々のツールはそれが用いられるシステムの環境と密接につながっている。そして、各システムの扱う言語現象は異なっている。このため、種々のシステムの評価を行うことには困難が予想されるが、今後の研究にとっては、これらの種々の提案の整理検討が必要である。

我々はかかる現状に鑑み、種々あるロジックプログラムをベースとした解析システムの比較検討作業に着手した。比較の対象としては電総研で開発されたＢＵＰ［Matsumoto 83］と、ＢＵＰをベースにして機能拡張と高速化をはかったＬａｎｇＬＡＢ［田中 86］（ＧＡＬＯＰ［横井 83，三吉 84］）、さらに並列構文解析手法を基に最近開発されたＳＡＸ［松本 86ａ，86ｂ］を選んだ。比較は、定性的な機能比較と定量的な構文解析時間およびメモリ効率の比較を行った。構文解析評価用の基本文法として電総研で開発された英語の文法を用いた。本論文はＩＣＯＴにおける構文解析システムの比較検討作業の第一段階として位置付けられるものであり最終報告ではない。

２．システムの概要

２．１．ロジックプログラミングに基づいた構文解析システム

Prologに埋め込まれたパーザとしてＤＣＧ［Pereira 80］がある。ＤＣＧはPrologに翻訳され直接実行される。しかしＤＣＧは完全なバックトラックベースのトップダウンパーザであるため１）左再帰性のある文法規則が扱えない、２）大きな文法では効率が悪くなるという問題がある。そこで電子技術総合研究所では、このような欠点を補うべくＤＣＧで記述された文法規則をボトムアップに構文解析する手法ＢＵＰの開発を行った［Matsumoto 83］、［松本 83］。これに改良を加えることでＣＹＫ法やEarly法に匹敵する高速アルゴリズムが実現された。

ＬａｎｇＬＡＢ（ＧＡＬＯＰ）は、以上のＢＵＰのアルゴリズムを基本原理として、それぞれ東京工業大学（ＩＣＯＴ）で開発された構文解析システムである。これらはいずれもＢＵＰに対し、さらに独自の幾つかの効率化を図っており、特にＬａｎｇＬＡＢではＢＵＰに比べ６０倍程度速くなっている。またこれらのシステムは単に構文解析を行うだけではなく、文法記述の枠組みの拡張やトレーサ等文法開発のための有用なツールなどによりトータルな自然言語処理システムとなっている。

これらの動きとは別にさらに最近では、ＤＣＧを文法記述言語と仮定し、ＢＵＰと同じ上昇型アルゴリズムを用いて、並列に実行する構文解析手法ＡＸが提案された［松本 86ａ］。これは並列型言語への変換を目的としているが、その実現のためにＢＵＰでは手続き的に記述されていた部分の、Prolog処理系に依存しないようなコードへの書き直しを行った。それは結果的に、Prolog処理系のコンパイル向きのコードともなったため、並列処理だけでなく逐次型言語の上でも高速実行が期待できることがわかった。このように逐次型言語の上で開発されたシステムを特にＳＡＸと呼ぶ（また並列型言語のシステムをＰＡＸと呼ぶ）［松本 86ｂ］。

２．２．システムの比較

機能比較では ＢＵＰ、ＬａｎｇＬＡＢ（ＧＡＬＯＰ）、ＳＡＸをとりあげる。

表１．　機能比較

		BUP	LangLAB	SAX
1	文法記述	DCG	XGS (DCGの拡張)	制限付DCG
2	解の探索方式	ボトムアップ 縦型全解	ボトムアップ 縦型全解	ボトムアップ 横型全解
3	補強項評価方法	手続き(Prolog)的	手続き(Prolog)的	決定(GHC)的と 遅延評価の組合せ
4	副作用	使用	使用	使用せず
5	環境複写	必要無し	必要無し	解析後に必要
6	形態素解析部	無し	有り	検討中
7	熟語処理	無し	有り	検討中
8	トレース・ デバッグ機能	無し	ゴール、解析木、 辞書引き、等各種 パラメータの表示	検討中
9	文法修正	全文法の再変換	修正部分の再変換 動的な文法修正	全文法の再変換
10	並列処理	考慮せず	考慮せず	可能（PΛX）

２．２．１．ＢＵＰ

ＢＵＰシステムでは、ＤＣＧ形式で記述された文法規則をＢＵＰトランスレータによりＢＵＰ節と呼ぶPrologプログラムに変換する。ＢＵＰ節は、自然言語の入力文に対しボトムアップで縦型探索の構文解析を行う。ＢＵＰ節がどのように構文処理を行うかは［Matsumoto 83］を参照されたいが、一般にボトムアップに処理を行う時に問題となる１）無駄な部分木の生成、２）同じゴールの計算 についてもＢＵＰではアルゴリズムをわずかに改良することで解決している。具体的には１）link述語によるトップダウン予測の導入、２）成功ゴールと失敗ゴールの登録 である。これにより理論的にはＣＹＫ法やEarly法に匹敵する高速アルゴリズムが実現されている。今回は、形態素、熟語処理を比較の対象としないということで表１．では示さなかったが、形態素処理、熟語処理等に関しても［Matsumoto 84］でＢＵＰに組み込まれている。例えば、単語辞書とは別に熟語辞書というものを準備することで熟語処理を行う。また形態素処理も下で述べるＬａｎｇＬΛＢとは全く異なる方法で処理する。それらの比較については［上脇 85］に詳しい。

２．２．２．ＬａｎｇＬＡＢ（ＧＡＬＯＰ）

ＬａｎｇＬＡＢは東京工業大学田中研究室で開発された自然言語処理システムである。本システムはＢＵＰを基本原理に左外置変形が扱えるような拡張を施したＢＵＰ＿ＸＧ［今野 86］と呼ばれるシステムをベースとしたシステムである。

ＬａｎｇＬΛＢシステムでは文法規則の記述は、ＤＣＧに左外置が扱えるように拡張を施したＸＧＳと呼ばれる形式を用いる。関係代名詞、疑問文など構造中の一部が欠落する文法を、従来のＤＣＧのように文法に加えることは文法の見通しを悪くする。そこで、ＸＧ［Percira 81］等に見られる痕跡発見機構を文法記述形式と解析メカニズムの拡張とによりＢＵＰに導入したのがＢＵＰ＿ＸＧである。これにより、従来のＤＣＧに比べ文法規則の数が３割程度減少し、特に、Yes-no疑問文に関する規則の減少が著しいことが報告されている［今野 86］。

解析では、ＢＵＰの解析アルゴリズムをリンク節の参照及び辞書引き等に関して改良しており、インタプリット時の処理速度はB-UPに比べて６０倍程度速くなっている。

またＬａｎｇＬＡＢは、辞書記述にＴＲＩＥ構造と呼ばれる共通の単語が共有できるような木構造を導入することにより、形態素、熟語等を効率よく処理する能力を持っている［上脇 85］。特に熟語処理では、一般に問題とされるフリーズ型（get up等）およびノンフリーズ型（not only (A) but also (B) 等）の両方が統一的に処理できるようになっている。例えば get up は get の辞書項目に記述するだけでよく、更に形態素処理とも関連してgot up, gets upなども認識出来る。またnot only (A) but also (B) では(A)と(B)が同じ機能を持つ語句（例えば同じ人称・数の名詞句）となるように指定

することが容易に出来る。

文法デバッグ機能としては部分木などのパラメータ表示機能等を持つトレーサを有する。このトレーサの特徴はＤＣＧ規則をＢＵＰのアルゴリズムでインタプリットしているため動的な文法修正が容易にできることである。この他に大規模な文法開発には有効な部分的な文法修正機能もＬａｎｇＬＡＢにはある。このようにＬａｎｇＬＡＢは総合的な自然言語処理環境を提供している。

ＧＡＬＯＰはＩＣＯＴで開発されたＬａｎｇＬＡＢと同様ＢＵＰをベースにしたものであり、ＰＳＩマシン、ＤＥＣ２０６０、等で使用出来るように整備されたものである。ＬａｎｇＬＡＢと異なり、左外置変形は扱わない。そこで関係代名詞や疑問文を扱う文法記述に出現することが多い空ルールの除去を行うためのシステムを変換前処理部として持つ。またデバッグ機能としては本来のターゲットマシンであるＰＳＩのマルチウィンドウシステムを利用した部分木などを表示するトレーサを有する。

２．２．３．ＳＡＸ

構文解析システムＳＡＸは、ＤＣＧで記述された文法をPrologのプログラムに変換するトランスレータより成る。変換手法は、ＤＣＧで記述された文法を並列論理型言語へ変換する事を目的として考えられたものであるが、逐次型言語による実現をＳＡＸと呼んでいる。

解析メカニズムはＢＵＰとは基本的に異なるアルゴリズムを基本にしており、このためＬａｎｇＬＡＢやＧＡＬＯＰとの共通点は主にトップダウンの文法カテゴリの予測だけである。相違点は、まずＳＡＸは解の検索が並行して行われ、全解を一度に得られる点である。またＬａｎｇＬＡＢ、ＧＡＬＯＰは解析の途中結果を順次副作用を用いて記録して行くが、ＳＡＸはこのような副作用を用いない。またＳＡＸにはバックトラックがなくその動作が決定的になっており、上記の副作用を用いない点とあわせて完全にコンパイルされた形で高速に動作する。

ＢＵＰ系のＬａｎｇＬＡＢ、ＧＡＬＯＰは補強項の評価を手続き的に行うが、ＳＡＸは決定的にこれを評価する。つまり、ＢＵＰ系の解析では補強項の評価の後でバックトラックが起きると補強項に曖昧さがある場合には最初に評価された解釈が棄てられて残りの曖昧さの中の一つを評価するが、ＳＡＸでは基本的にバックトラックがおきないためこのような処理は行われない。ＤＣＧの補強項の中に曖昧さがある場合、この中の一つだけの解しか引き出さない。また、並列処理を前提として開発されたため、逐次型言語の上で実現する場合には論理変数による環境複写の問題を有する。これらの問題点は現在、遅延実行制御によってほぼ対応が取れるが、現在遅延実行制御のメカニズムを持たない制御系でも十分に機能を発揮するように改良中である［杉村 86］。 それまではユーザに対する文法記述における制限となる（制限付ＤＣＧ）。

また現時点ではＳＡＸには形態素処理能力はなく、ＤＣＧの書き替え規則で対応しているが、全解探索による形態素解析能力を持たせることはさほど困難な事ではない。

デバッグ機能に関しては、ＳＡＸはまだ開発されたばかりということもあり、現在のところサポートは十分ではない。特にＳＡＸはその基本アイデアが並列実行であるため、何らかのデバッグ機能はぜひ必要である。

３．構文解析評価

３．１．構文解析評価用の文法・辞書および評価環境

- 計算機環境…VAX 11/785 VMS(4.3版)上のQuintus Prolog(1.1版)
- 文法規則・辞書…英語、ＤＣＧ規則総数６０７（文法規則約２００＋辞書約４００）
- 例文…１４文 （＜付録＞参照）
- 対象システム…ＢＵＰ（［松本 83］版）、ＬａｎｇＬＡＢ、ＳＡＸ
- 評価項目 ・速度…コンパイル時、インタプリット時の全解探索に要する時間
  （ただしガーベイジコレクション、スタックシフトの時間は除く）
  ・メモリ効率…プログラムサイズと解析時ヒープ使用量
- 評価条件 ① 形態素解析は含めない
  ② ＬａｎｇＬＡＢの評価文法はＸＧではなくＤＣＧとする

ＬａｎｇＬＡＢの文法をＢＵＰ、ＳＡＸと同じＤＣＧとしたのは、この評価文法にはＸＧに最も効果的とされる疑問文が含まれないためである。この結果、ＧＡＬＯＰとＬａｎｇＬＡＢには本質的な差がなくなり、今回はＬａｎｇＬＡＢだけを測定した。ＧＡＬＯＰにもＬａｎｇＬＡＢと同じ最適化を施せばほぼ等しい結果が得られる。またこれらの最適化の効果を調べるために、［松本 83］版のＢＵＰも評価の対象とした。

現在、ＳＡＸには形態素解析が組み込まれていないため、形態素解析を評価の対象にしなかった。

３．２．評価結果の比較・検討

時間評価結果を表２．に示す。ＢＵＰとＬａｎｇＬＡＢを比べるとインタプリット時においてもコンパイル時においても３－３０倍の高速化ができていることがわかる。リンクに関する最適化などＬａｎｇＬＡＢに導入された最適化の中には、文法が大きいほど効果の大きいものがある。今回の評価文法の文法数が２００と中規模であったため、実用規模の文法になるとこの差はさらに広がる。

ＳＡＸとＬａｎｇＬＡＢのインタプリット時の結果をみると、ほとんど変わらない。両者ともトップダウン予測を用いる等、理論的な探索空間がほぼ等しいことを表している。これに対し、コンパイル時にはＳＡＸがＬａｎｇＬＡＢに比べ５－１０倍以上の速度が得られている。

このような顕著な差がでたのは

１）ＳＡＸがコンパイラ処理の最適化を有効に利用している。すなわち、ＳＡＸはインデックスサーチ（ハッシュ）機能および末尾再帰の最適化（ＴＲＯ）を強く意識したパーザであり、

２）さらに、ＳＡＸでは途中結果を副作用を用いて記憶する必要がない

からだと考えられる。両者の差は文が長いほど、また解釈数が多いほど大きくなっている。ＳＡＸでは文の長さに関わらず１単語平均20msecで解析し、多くの解釈数を持つ時でも、せいぜい２倍程度の時間しか必要としない。これに対しＬａｎｇＬＡＢでは、文が長くなりまた解釈数が多くなるにつれ、１単語の平均解析時間が増えている。これらは次のようなことも、両者の実行メカニズムの違いとして原因となっていることを示唆している。

３）バックトラックのないＳＡＸに対し、ＢＵＰ系のＬａｎｇＬＡＢ、ＢＵＰではバックトラックが頻繁に行われるため、大きな構造を持ち回るほどスピード低下を招く。

４）ＢＵＰ系では登録ゴールの増加に伴い、その探索時間が大きくなる。これはQuintus Prologではアサートされたゴールの探索がインタプリータモードで実行されるためである。

ここで時間評価の参考データとして、ＰＳＩ上でのＳＡＸのデータを示しておく。いずれもVAX-Quintus Prologに比べ２－２．５倍程度速くなっており全文１秒以内に解析できている。

表３．に、評価文法と変換後のプログラムサイズおよびコンパイル後のオブジェクトサイズを示す。表４．には、ＢＵＰとＬａｎｇＬＡＢの解析中に使用するヒープ量を示す。コンパイル時とインタプリット時が同じ使用量であったため値は１つだけである。またアサートを用いないＳＡＸは、解析中にヒープを使用しないためデータはない。

表３．から明らかなように、ＳＡＸではコンパイラの最適化を働かせるために、変換後のプログラムのサイズは４－５割程度大きくなっている。しかし、ＢＵＰ系では解析時に動的にヒープを使用するため、長い文になるとメモリ使用総量（プログラムサイズ＋解析時使用量）は、表４．によると逆に多くなっている。また各文の解析中のヒープ使用量をみると、それぞれの解析時間と非常に深い相関関係があることがわかる。これは、ゴール登録および探索の処理が解析時間に大きく影響していることを示唆している。これについては、確認のためのなんらかの測定を必要とするが、もしそうであるならばＢＵＰ系の今後の改良の方向を示している。

スタック使用量に関してはProlog処理系の都合上、今回は測定できなかったが、大きな構造体を持ち回すＳＡＸの特性を知る上でぜひ必要な値である。

（ここでのメモリ効率の値はあくまで相対的な評価のために現在の値を示したものであり、この値は処理系に依存することはもちろんであり、さらに各システムの改良が今後十分に可能な項目でもある）

４．考察と結論

３．で示した速度結果のように、コンパイル時には、ＳＡＸがＢＵＰ系に比べ５－１０倍の速度があることがわかった。この理由としてコンパイラ処理系のハッシュ、ＴＲＯ等の最適化が非常に有効に働いているのは間違いないが、ＳＡＸの実行メカニズムとして、バックトラック、副作用を用いなかったのも大きいといえる。ＢＵＰ，ＬａｎｇＬＡＢでは３．４で示したようにゴール登録および探索の処理が解析時間に大きく影響を及ぼしている。これは言い替えれば、バックトラックと副作用機能が高速な－例えば配列を持つ、またアサートするデータにはハッシュが働く等の－処理系が得られれば、ＳＡＸの値に近づくことができることを示唆する。その意味でコンパイル時のＳＡＸの解析速度はＢＵＰ系のシステムにとって１つの上限を示していると考えている。したがってＳＡＸは速度的に逐次型計算機の上での汎用の構文解析システムとしては、十分に満足ゆくものであると考える。もし今後さらに高速なパーザの必要性が生じれば、それは並列処理により実現されるかもしれない。その点でもＳＡＸはもともとのアイデアが並列処理を意識したものであるため、整合性はよい。

構文解析システムとしては、単に文章を解析するだけでなく、文法開発環境等さまざまなユーティリティが必要である。その面では、現在のところＬａｎｇＬＡＢが、ＸＧの導入、熟語処理・形態素処理、文法開発支援等非常に快適な環境を提供している。逆に、速度面で非常に優れているＳＡＸの場合、そのアイデアが並列環境を対象としているためデバッグが困難である。さらに熟語処理、形態素処理なども今後の課題である。現時点では、文法開発時にはインタプリタモードでＬａｎｇＬＡＢを用い、デバッグ終了時に文法をＳＡＸトランスレータで変換し、コンパイルして実行する方式が考えられる。それには、ＳＡＸに表１．の検討中の項目を早急に実現することが必要であろう。

謝辞

最後に、本研究の機会を与えていただいたＩＣＯＴ第二研究室横井俊夫室長に感謝します。また本研究を進めるにあたり、御討論して頂いたＩＣＯＴ自然言語処理グループならびに、東工大情報工学科田中研究室の皆様には深く感謝致します。

参考文献

[上脇 85] 上脇正、田中穂積：辞書のＴＲＩＥ構造化と熟語処理、
Proc. of The Logic Programming Conference'85,ICOT,329-340(1985).

[今野 86] 今野聡、田中穂積：左外置を考慮したボトムアップ構文解析、
日本ソフトウエア科学会編コンピュータソフトウエア、3巻,2号,115-125(1986).

[Matsumoto 83] Matsumoto, Y. et.al.: BUP--A Bottom-up Parser Embedded in Prolog, New Generation Computing, 1,2, 145-158(1983).

[松本 83] 松本裕治、清野正樹、田中穂積：ＢＵＰの高速化、
情報処理学会自然言語処理研究会、39-7 (1983).

[Matsumoto 84] Matsumoto, Y., Kiyono, M. and Tanaka, H.: Facilities of the BUP Parsing System, Proc. of 1st International Workshop on Natural language and Logic Programming, Rennes, France, Sept. (1984), also in 'Natural Language Understanding and Logic Programming', V.Dahl and P.Saint-Dizier(ed.), Elsevier Science Publishers B.V., (1985).

[松本 86ａ] 松本裕治：並列構文解析、情報処理学会自然言語処理研究会、53-2 (1986).

[松本 86ｂ] 松本裕治、杉村領一：論理型言語に基づく構文解析システムＳＡＸ、
日本ソフトウエア科学会編コンピュータソフトウエア、3巻,4号 (1986).

[三吉 84] 三吉秀夫、平川秀樹、安川秀樹、向井国昭、古川康一、森下太郎：ＢＵＰシステム、
ICOT,TR-38 (1984).

[Pereira 80] Pereira, F. and Warren, D.: Definite Clause Grammars for Language Analysis --A survey of the Formalism and a Comparison with Augmented Transition Networks, Journal of Artificial Intelligence, 13, 231-278(1980).

[Pereira 81] Pereira, F.: Extraposition Grammars, American Journal of Computational Linguistics, 7, 4, 243-256 (1981).

[杉村 86] 杉村領一、松本裕治：構文解析解析システムＳＡＸのＣＩＬによる実現、情報処理学会第33回全国大会、(1986).

[田中 86] 田中穂積、上脇正、奥村学、沼崎浩明：自然言語処理のためのソフトウェアシステムＬａｎｇＬＡＢ、Proc. of The Logic Programming Conference'86,ICOT,5-12(1986).

[横井 83] 横井俊夫、向井国昭、三吉秀夫、安川秀樹、平川秀樹：論理プログラミングによるボトムアップパーサ（ＢＵＰ）トランスレータの開発、情報処理学会第26回全国大会、(1983).

表２．　構文解析時間測定結果

番号	語数	解釈数	インタプリット時 CPU TIME			コンパイル時 CPU TIME			[参考] PSI
			BUP	LangLAB	SAX	BUP	LangLAB	SAX	SAX
1	8	2	22,200	6,930	7,250	4,810	630	160	60
2	11	1	29,450	9,090	9,780	6,320	780	170	73
3	11	1	28,460	8,630	9,870	6,440	740	190	79
4	12	1	48,730	16,170	17,330	11,500	1,510	290	126
5	15	2	61,210	18,950	21,640	18,720	1,930	360	155
6	20	1	73,190	22,710	27,100	22,190	2,350	420	186
7	21	3	100,070	28,880	33,950	40,820	3,120	540	239
8	23	7	287,460	66,640	78,000	163,280	8,200	1,100	508
9	24	1	109,840	33,290	33,110	40,760	3,340	590	250
10	24	4	156,790	57,460	55,760	65,780	5,230	690	332
11	28	1	207,070	58,220	56,860	101,220	6,300	870	394
12	32	12	744,950	116,180	118,460	568,950	16,610	1,530	741
13	34	4	209,890	60,980	54,240	108,190	6,440	820	375
14	39	1	251,520	65,180	56,460	134,020	6,230	860	390

（時間の単位 ： msec）

注意）「語数」は入力した形態素の数を示す。

表３．　モジュール・サイズ

	モジュール・サイズ bytes			
	文法(DCG)	BUP	LangLAB	SAX
テキスト形式	43,520	70,144	59,392	96,256
実行形式	58,136	103,364	106,668	156,986

注意）「テキスト形式」では文法、ＬａｎｇＬＡＢ、ＳＡＸはＤＣＧ記述のため引数が２つ少ない。
「実行形式」はコンパイルコードの大きさである。

表４．　　実行時メモリ使用量測定結果

番号	語数	解釈数	解析時ヒープ使用量	
			BUP	LangLAB
1	8	2	7,556	4,840
2	11	1	8,216	4,104
3	11	1	8,804	4,480
4	12	1	19,592	12,392
5	15	2	25,472	14,952
6	20	1	32,264	18,008
7	21	3	49,316	27,904

番号	語数	解釈数	解析時ヒープ使用量	
			BUP	LangLAB
8	23	7	138,664	100,608
9	24	1	53,472	29,800
10	24	4	49,508	26,304
11	28	1	84,788	49,744
12	32	12	242,992	155,424
13	34	4	139,366	78,584
14	39	1	106,844	39,010

（メモリ量の単位 ： bytes）

＜付録＞　評価用例文

以下に評価に用いた例文を示す。下線の部分は分離して入力した。

1. He explains the example with rules.
2. The structural relations are holding among consistents.
3. He explained the example and he illustrates the rule.
4. It is not tied to a particular domain of applications.
5. Diagram analyzes all of the basic kinds of phrases and sentences.
6. This paper presents an explanatory overview of a large and complex grammar that is used in a sentence.
7. Diagram analyzes all of the basic kinds of phrases and sentences and many quite complex ones.
8. The annotations provide important information for other parts of the system that interpret the expression in the context of a dialogue.
9. For every expression it analyzes , diagram provides an annotated description of the structural relations holding among its constituents.
10. Procedures can also assign scores to an analysis , rating some applications of a rule as probable or as unlikely.
11. This paper presents an explanatory overview of a large and complex grammar, diagram , that it used in a computer for interpreting english dialogue.
12. Its procedures allow phrases to inherit attributes from their constituents and to acquire attributes from the larger phrases in which they themselves are constituents.
13. Consequently , when these attributes are used to set context_sensitive constraints on the acceptance of an analysis , the contextual constraints can be imposed by conditions on constituency.
14. It is not tied to a particular domain of applications , and it can be extended to analyze additional constructions , using the formalism in which it is currently written in the rule.