TR-I-0333

# 単一化に基づく構文解析: 実践編
## --- 日本語解析システムの現状と課題 ---

An Empirical Study on Unification-Based Parsers

永田昌明　田代敏久　松尾秀彦*　高橋誠*
Masaaki NAGATA, Toshihisa TASHIRO,
Hidehiko MATSUO, Makoto TAKAHASHI

1993.3

## 概要

本報告では、単一化に基づく日本語解析システムの実行効率を改善するために行われた様々な研究に関する評価実験の結果を述べる。具体的な評価項目は、アジェンダ制御、弧の共有、素性記述の評価時期、入力の形式、辞書の構成、単一化アルゴリズム、文法の規模と粒度、などである。本報告では、各項目における最適な手法と、それらの最適な組合せについて述べ、これらの改良により従来の手法と比べて大幅な性能の向上が達成できたことを実証する。

ATR 自動翻訳電話研究所
ATR Interpreting Telephony Research Laboratories



*(株)東洋情報システム

# 目次

# 第 1 章

# はじめに

## 1.1 本報告の構成

本報告では、単一化に基づく日本語解析システムの実行効率を改善するために行なわれた様々な研究に関する評価実験の結果を述べる。

本報告は、我々が開発した日本語解析システムの現状の性能と今後の課題を解説することが目的であり、単一化に基づく構文解析の技術的な解説書ではない。技術的な解説を必要とする場合には、「単一化に基づく構文解析: 入門編」[永田, 92]を参照して欲しい。また、日本語文法に関する解説書としては、[永田 ほか, 93a]や[永田 ほか, 93b]がある。

本報告の構成を以下に示す。

2章では、構文解析プログラムに関する研究項目の概要および実験結果について述べる。ここでは、アジェンダ制御、弧の共有、素性記述の評価時期、入力の形式、辞書の構成の 5 つの項目について、各項目の実装方法に関する技術的な選択肢を述べ、最も実行効率が良い組み合わせを実験的に決定する。

3章では、単一化プログラムに関する研究項目の概要および実験結果について述べる。ここでは、選言を含まない素性構造の単一化アルゴリズムを 3 つ述べ、構造共有型の準破壊的グラフ単一化が最も優れていることを示す。

4章では、日本語文法の構成方法に関する研究項目の概要および実験結果について述べる。ここでは、まず、文法の粒度と素性構造の評価時期の関係に関する実験結果を述べる。次に、文法の規模と日本語解析システムの実行効率の関係に関する実験結果を述べる。

5章では、2, 3, 4章で説明した手法を組み合わせた場合の総合的な性能改善効果について述べる。そして、素性記述最終評価戦略、構造共有型準破壊的グラフ単一化、中粒度文法を組み合わせることにより、従来手法に比べて、大きな実行効率の改善が達成できたことを示す。

6章では、各評価実験から得られた結果をまとめ、今後の課題を述べる。

付録では、日本語解析システムに関して、本編で解説できなかった事柄について述べる。付録 Aでは、選言を含む素性構造の単一化のために用いられているデータ構造について述べる。付録 Bでは、文法コンパイラの概要を述べる。付録 Cでは、実験に用いられた文法の規模と粒度を表す統計量について述べる。最後に、付録 Dでは、日本語解析システムの開発の経緯を解説し、数々の失敗から我々が学んだ教訓について述べる。

なお、この日本語解析システムは、 ATR が開発した世界初の自動翻訳電話システム ASURA [竹沢 ほか, 93] の日本語解析部として用いられている。

## 1.2 日本語解析システムの構成

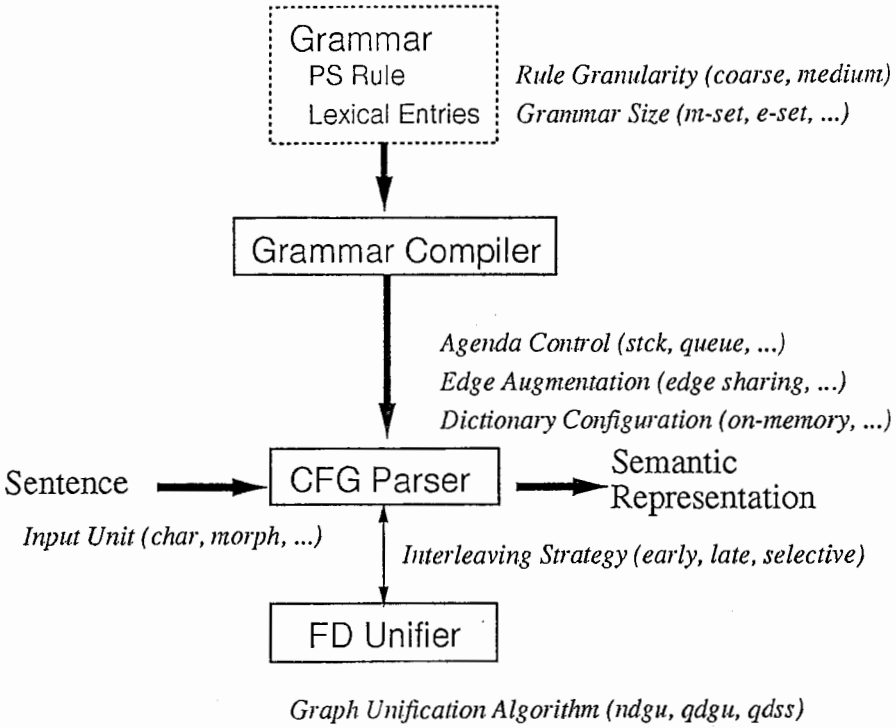


図 1.1: 日本語解析システムの構成

図 1.1 に日本語解析システムの構成を示す。日本語解析システムは、大きく分けて、次の三つのモジュールから構成されている。

**パーザ (Parser)** 句構造規則に基づいて入力文の構造を解析する。

**ユニファイア (Unifier)** 選言を含む素性構造の単一化を行なう。

**文法コンパイラ (Grammar Compiler)** 文法記述言語で書かれた文法を、内部データに変換する。

日本語解析システムでは、言語的知識記述である文法と、推論 (処理) 系である構文解析システムが、完全に分離されている。このような構成にすることの利点は、(1) 開発された文法は、一般性が高く、拡張・保守や再利用が容易になる、(2) 文法とは独立に、様々な解析戦略を試すことができる、などである。

以下では、各モジュールの概要を説明し、その中での研究項目、および、その評価実験の結果と考察を述べる。ただし、文法コンパイラについては、特にアルゴリズム的な改良が行なわれていないので、付録 B に概要を示すにとどめる。

また、日本語解析システムの文法は、単一化に基づく語彙統辞論的 (lexico-syntactic) な枠組である HPSG [Pollard, 87] や JPSG [Gunji, 87] に基づいて、宣言的 (declarative) に記述されている。その具体的な内容については、本報告では触れないが、実行効率に関連が深いと思われる数値については、付録 C で述べる。

なお、本報告では、読者は、構文解析アルゴリズムや単一化アルゴリズムに関する基本的な知識を有するものと仮定し、日本語解析システムの評価項目を説明するのに最低限必要な内容しか解説しない。

# 第 2 章

# 構文解析プログラム (パーザ)

## 2.1 構文解析プログラムの概要

本システムでは、文脈自由文法[1]を解析する基本的な枠組として、Kay によって提案されたアクティブチャート法 (Active Chart Parsing) [Kay, 1980] を用いる。

図 2.1に示すように、アクティブチャートパーザ (Active Chart Parser) は、解析過程の動作を記録する表であるチャート (chart) と、解析過程の動作を制御するアジェンダ (agenda) から構成される。

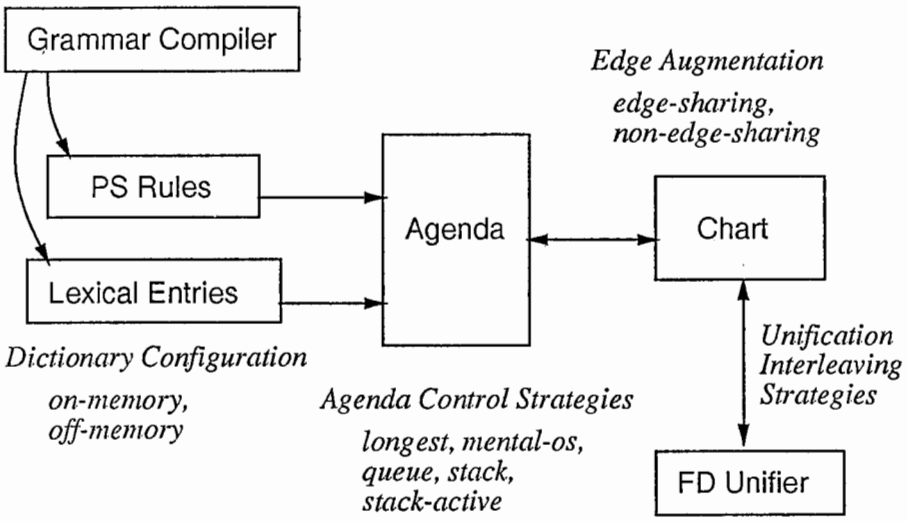


図 2.1: アクティブチャートパーザの構成

チャートの構成要素は弧 (edge) である。弧は、入力文の部分文字列と、句構造規則が規定しているカテゴリの列との対応関係を示す。不活性弧 (inactive edge) は、一つの句構造規則に規定されたカテゴリ列の制約を完全に満足する部分文字列に対応する。不活性弧は、一つの規則の全ての娘カテゴリが認識され、親カテゴリが認識された状態を表す。活性弧 (active edge) は、句構造規則に規定されたカテゴリの列の制約を (左から右にみて) 部分的に満足する部分文字列 (空列でもよい) に対応する。活性弧は、一つの規則の一部の娘カテゴリが認識されているが、親カテゴリはまだ認識されていない状態を表す。

[1]正確には、素性構造によって制約が付加されているので拡張文脈自由文法 (Augmented Context Free Grammar) である。

```
procedure ActiveChartParsing(sentence,grammar)
begin
  AcpContinue(AcpInitializeChart(sentence,grammar))
end

procedure AcpContinue(chart)
begin
  if SatisfySuspendingCondition?(chart)
    then return chart
  else
    AcpContinue(AcpOneStep(chart))
end

procedure AcpOneStep(chart)
begin
  pendingedge = GetPendingEdge(chart)
  AddEdge(pendingedge)
  if EdgeActive?(pendingedge) then
    TryToContinueActiveEdge(pendingedge, chart)
  else
    TryToContinueInactiveEdge(pendingedge, chart)
  ProposeProductions(pendingedge)
  return chart
end
```

図 2.2: アクティブチャート法のアルゴリズム

図 2.2にアクティブチャート法のアルゴリズムの概要を示す。チャート法は、チャートの初期化 (AcpInitializeChart) と、規則の適用の繰り返し (AcpContinue) から構成される。 AcpContinue は、ステップを一つ実行するたびに、解析の中断条件 (suspending condition) を調べる。 AcpOneStep は、規則の適用の一つのステップを実行する。一つのステップは、新しい待機弧 (pending edge) を取り出し (GetPendingEdge)、それをチャートに加え (AddEdge)、活性弧と不活性弧を組み合わせ、新しい弧を提案する (ProposeProductions) という過程から構成されている。パーザは、中断条件が成立した (SatisfySuspendingCondition?) 時に、すなわち、弧の始点と終点が文頭と文末に対応し、弧のラベルが文法の開始記号であるような不活性弧が見つかった時に停止する。

以下では、パーザに関する、次の 5 つの項目に対して評価実験を行なった結果を述べる。

- アジェンダ制御
- 弧の共有
- 素性記述の評価時期
- 形態素解析の負荷
- 辞書の構成

## 2.2 アジェンダ制御

新たに作成された弧は、すぐにチャートに組み込むのではなく、一度、待機弧リスト (pending edge list) に蓄積される。この待機弧リストから弧を選択する方法を変えることによって、解析過程を制御することができる。すなわち、待機弧リストはアジェンダの一種と考えることができる。

### 2.2.1 アジェンダ制御戦略

ここでは、次の5つのアジェンダ制御戦略 (待機弧選択モード) を用意した。

stack 待機弧を LIFO (Last In First Out) で選択する。縦型探索 (depth first search) と類似の動きをする。

queue 待機弧を FIFO (First In First Out) で選択する。横型探索 (breadth first search) と類似の動きをする。

longest 弧が表現する規則の右辺の長さを比較し、長い方を選択する。最長一致法と類似の動きをする。

stack-active 活性弧優先の LIFO で選択する。すなわち、待機弧リストに活性弧が存在すれば、その中で一番最後に格納された弧が選択される。これは、shift-reduce パーザにおいて、shift を優先するのに似ている。

mental-os 左枝分れ構造 (left-branching structure) など、日本語の文の構造の preference にあった解を最初に見つけられるような動きをする[2]。

### 2.2.2 実験と考察

上述の5つのアジェンダ制御戦略に関して、解析時間、チャートパーザのステップ数、出力する構文木の数、作成したノードの数、グラフ単一化関数の呼び出し回数などを調べた。結果を表2.1に示す。

表 2.1: アジェンダ制御 (14文, 平均19.2文字)

| | time (sec) | chart steps | tree | fs | dag nodes | unify-fs | |
|---|---|---|---|---|---|---|---|
| | | | | | | calls | success |
| m-set.qdss.late.longest | 8.9 (1.25) | 939.2 (1.00) | 1.4 | 1.4 | 12238.1 (1.00) | 387.9 | 84.0% |
| m-set.qdss.late.mental-os | 8.1 (1.14) | 901.3 (0.96) | 1.1 | 1.2 | 9324.1 (0.76) | 318.9 | 84.2% |
| m-set.qdss.late.queue | 6.8 (0.96) | 920.9 (0.98) | 1.4 | 1.4 | 12116.0 (0.99) | 384.6 | 83.9% |
| m-set.qdss.late.stack | 7.1 (1.00) | 937.8 (1.00) | 1.4 | 1.4 | 12240.5 (1.00) | 387.9 | 84.0% |
| m-set.qdss.late.stack-active | 5.8 (0.82) | 866.1 (0.92) | 1.1 | 1.2 | 9559.9 (0.78) | 319.4 | 84.7% |

表2.1を見ると、アジェンダ制御の方法を変更することにより、解析時間やステップ数がかなり変化することがわかる。また、アジェンダ制御戦略により、第一解で得られる構文木の数が異なることにも注意が必要である。これは、弧の共有により local ambiguity packing が行なわれているせいである。

[2]島津氏は、Mental OS の立場から、このような preference を提案した[Tomabechi, 91]。我々は、彼に敬意を表して、この待機弧選択モードを mental-os mode と呼んでいる。

- 現在 default で用いている stack の平均的振舞いは、他に比べて遜色ない。
- 第一候補を速く求める目的であれば、stack-active が最も効率的である。
- 少ないスッテプ数で第一候補を得る目的であれば、mental-os が適しているようである。

mental-os は、スッテプ数が少ない割に、解析時間がかかっている。これは、エッジのソートにかかるコストが大きいのだと思われる。ただし、プログラミング・テクニックを駆使すれば、エッジをソートするコストを小さくできる可能性もある。日本語の選好 (preference) に基づいて、最も尤もらしい解を最初に求められるという mental-os の性質を考えると、今後は、これを default にすべきかもしれない。

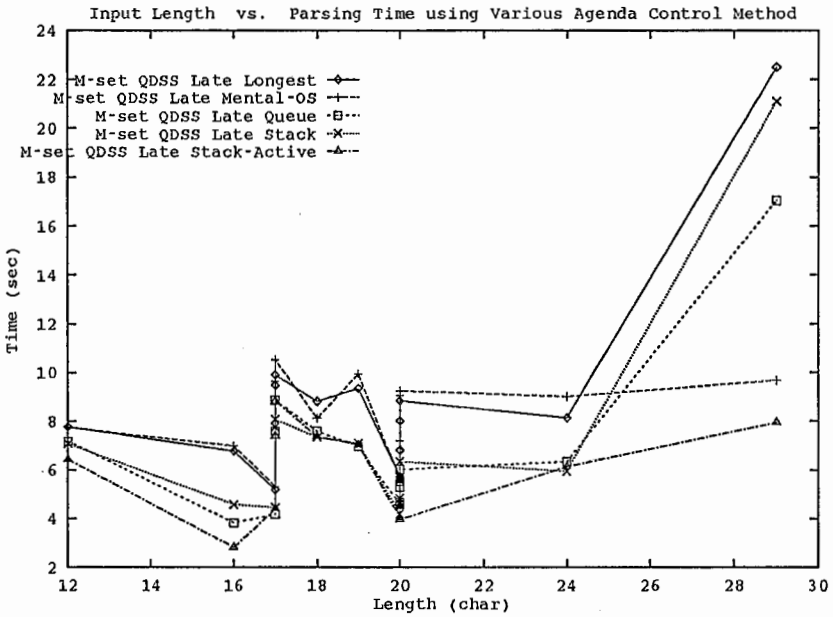


図 2.3: 様々なアジェンダ制御戦略に関する入力文の長さと解析時間の関係

図 2.3に、アジェンダ制御戦略を変えた時の、入力文の長さと解析時間の関係を示す。これを見ると、文が長くなる (ステップ数が大きくなる) ほど、stack-active や mental-os が優れているように思える。恐らく、文法が大きくなった時も同様の傾向が得られるだろう。

構文解析は、組合せ問題の最適解を求める探索問題と考えることができる。従って、長い文や大きな文法 (語彙) を対象にする時には、探索の heuristics が非常に重要になる。

今後は、(1) 解が得られるまでの時間、(2) 解が正解である割合、などを評価尺度として、アジェンダ制御戦略の研究をする必要がある。特に、mental-os のような heuristics に基づく方法と、確率的な手法との兼ね合いが焦点となるだろう。

### 試験文 (14 文, 平均 19.2 文字)

アジェンダ制御戦略に関する実験で使用された試験文を以下に示す。

```
d01-15 住所は大阪市北区茶屋町二十三です
d01-18 登録用紙を至急送らせていただきます
d02-04 今会議に申し込めば参加料はいくらですか
```

d03-08 ところで会議での公式言語は何ですか
d04-13 住所は大阪市東区玉造二丁目二十七の七です
d05-09 既に登録料の八万五千円を振り込まれておられますね
d05-14 案内書にも書いていますが
d05-16 後日プログラムと予稿集をお送りいたします
d07-08 申し訳ありませんがこちらでは専門的な質問にお答えできません
d07-12 それでは早急にその案内書を送ってください
d08-09 六月三十日までに原稿の送付をお願いします
d10-09 二人部屋の値段は九千五百円から六万円です
d10-11 どちらのホテルが会議場に近いですか
d10-14 ホテルの手配もしていただけるのですか

## 2.3 弧の共有

素性記述による制約を扱えるようにチャート法を拡張することは、予想外に難しい問題を含んでいる。少なくとも、これには、(最悪の場合の) 理論的な計算量が異なる二つの方法がある。

### 2.3.1 弧を共有しない拡張

第一の方法は、各々の弧 (edge) が、その弧に対応する句構造規則の素性記述を記録するように拡張 (augment) するものである。これを非弧共有型の拡張 (non-edge-sharing augmentation) と呼ぶことにする。

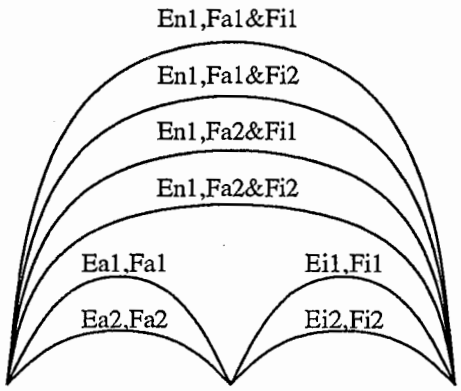


図 2.4: 弧を共有せずに素性記述を格納する方法

文脈自由文法であれば、同じ部分列に広がり、同じカテゴリを既に同定し、これから同定すべきカテゴリも同じである時に、二つの弧が等しいといえる。しかし、素性記述で弧を拡張すると、二つの弧が等しい条件は、上述の他に素性記述が等しいことが必要になる。文脈自由文法の構文解析が $o(n^3)$ であるのは、等しいという条件が構成要素の内部の構造に依存しないからである[3]。異なる娘カテゴリの組み合わせは、異なる素性記述を生じるかもしれないので、必ずしも親の弧をマージすることができない。従って、図 2.4 に示すように、非弧共有型の拡張は、最悪の場合、指数的な計算量を必要とするかもしれない。

活性弧と不活性弧を組み合わせる時に必ず素性記述を評価する (後述する「早期評価戦略」) ようにすれば、素性記述の制約を満足しない弧が原因で、(指数的な数の) 不適切な弧が生することを防げる。これが、単一化に基づく構文解析で、早期評価戦略[4]が何の疑いもなく使われてきた原因だと思われる。しかし、たとえ早期評価戦略を採用しても、非弧共有型の拡張が、最悪の場合、指数的な計算量を必要とすることに変わりはない。

### 2.3.2 弧を共有する拡張

第二の方法は、同じ部分列に広がり、同じカテゴリを既に同定し、これから同定すべきカテゴリも同じである弧は、すべてマージしてしまう方法である。これを弧共有型の拡張 (edge-sharing augmentation) と呼ぶことにする。図 2.5この方法では、異なる娘の組み合わせから生じた素性記述は、リスト形式で親に記録される。この各々の組み合わせ、すなわち、娘の弧へのポインタ

[3]構成要素の内部の構造と、その構成要素の外部の構造 (文脈) が独立なことが context-free と呼ばれる理由である。そして、この性質ゆえに packing が可能となり、効率的な構文解析手法が存在するわけである。

[4]我々が「最終評価戦略」を提案するまで、この部分が議論の対象になることはなかった。

と組み合わせの結果として得られた素性記述は、edge-internal というデータ構造の中に記録されている。

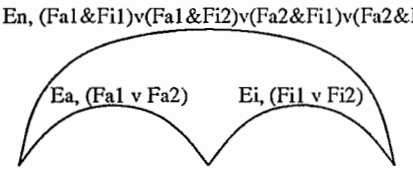


図 2.5: 弧を共有しながら素性記述を格納する方法

このような弧の共有を行なうことにより、単一化に基づく構文解析でも、多項式時間で構文解析できるという CFG の性質を保つことができる[5]。しかし、弧共有型の拡張では、既に存在する弧に対して新しい素性構造を付け加える際に注意が必要になる。もし、その弧を娘とする弧が既に存在していれば、その娘の弧と姉妹の弧との組み合わせを考慮した素性記述を親に付け加えなければならない。そして、この作業は、その弧を使用している親がなくなるまで、再帰的に行なわなければならない。この目的のために、各弧は、その弧の親の弧、および、親の弧を作る時に組み合わされた姉妹の弧へのポインタを持っている。

### 2.3.3 実験と考察

弧共有型パーザ (es) と弧非非共有型パーザ (nes) の 2 つのパーザに関して、素性記述評価戦略を変えながら (early および late)、解析時間などを調べた。結果を表 2.2に示す。

表 2.2: 弧の共有の効果 (9 文, 平均 19.3 文字)

| | time (sec) | chart steps | trees | dag nodes | unify-fs | | unify-desc | |
|---|---|---|---|---|---|---|---|---|
| | | | | | calls | success | calls | success |
| m-set.qdss.early.es | 15.9 (1.00) | 610.6 (1.00) | 6.0 (1.00) | 39950.8 (1.00) | 1457.8 | 78.5% | 143.4 | 75.3% |
| m-set.qdss.early.nes | 15.4 (0.97) | 626.6 (1.03) | 3.3 (0.55) | 39014.2 (0.98) | 1441.1 | 78.5% | 140.6 | 75.0% |
| m-set.qdss.late.es | 7.6 (0.48) | 920.7 (1.51) | 6.0 (1.00) | 13105.4 (0.33) | 413.9 | 83.0% | 43.8 | 92.4% |
| m-set.qdss.late.nes | 7.9 (0.50) | 1227.4 (2.01) | 3.3 (0.55) | 7950.2 (0.20) | 275.2 | 84.0% | 29.7 | 94.1% |

- この例では、弧共有型と弧非共有型のパーザについて、第一候補を見つけるまでの時間はあまり差がない。しかし、弧を共有したパーザは、弧を共有しないパーザの約 2 倍の木を見つけている。我々の経験では、一般に、文が長くなると、弧を共有するパーザの方がかなり速い。

- 弧非共有型パーザと最終評価戦略を組み合わせると、ステップ数が非常に大きくなる。これに対して、弧非共有型パーザと早期評価戦略を組み合わせた時には、弧共有型パーザと比べてもそれほどステップ数が増えない。

[5]これは単一化文法の句構造規則に関する部分は多項式時間で解析できるという意味である。そもそも HPSG などの単一化文法は、subcat 素性や slash 素性を用いて局所的な句構造の外部に影響を及ぼすことができるから、CFG の範囲を越えている。従って、そのツケは何処かで払わなければならない。問題は、文脈依存性を導入することによって生じる計算的な負荷を、どこで吸収するのが、全体の効率を考えた時に一番良いかという点にある。

これより、計算コストが小さい edge pruning の方法が利用可能ではない現状では、できる限り弧を共有して、最後に素性構造を評価する方法、すなわち、弧共有型パーザと最終評価戦略の組合せが最も効率が良いことがわかる。

しかし、選択評価戦略 (select) のように素性記述を部分的に評価する方法や、確率に基づいてアジェンダを制御する方法のように、「計算コストが小さい edge pruning の方法」があれば、弧を共有しない方が、速く第一解が求まる可能性がある。ただし、最悪の場合は、理論的には、指数的な計算を必要とすることには変わりない。

**試験文 (9 文, 平均 19.3 文字)**

d01-15 住所は大阪市北区茶屋町二十三です
d01-18 登録用紙を至急送らせていただきます
d02-04 今会議に申し込めば参加料はいくらですか
d03-08 ところで会議での公式言語は何ですか
d04-13 住所は大阪市東区玉造二丁目二十七の七です
d05-09 既に登録料の八万五千円を振り込まれておられますね
d05-14 案内書にも書いていますが
d05-16 後日プログラムと予稿集をお送りいたします
d07-08 申し訳ありませんがこちらでは専門的な質問にお答えできません

## 2.4 素性記述の評価時期

句構造規則の解析に比べ単一化の計算コストは非常に大きいので、句構造規則の注釈や語彙項目が持っている素性記述による制約の評価時期は、全体の処理効率に大きく影響する。

### 2.4.1 素性記述評価戦略

ここでは、次の3つの素性記述評価戦略を用意した。

**早期評価戦略 (early unification)** 規則の適用のたびに素性記述を評価する。これを step-by-step stragety と呼ぶ時もある。これは従来より、単一化に基づく構文解析で採用されてきた方法である。

**最終評価戦略 (late unification)** 構文木が完成した段階で最後にまとめて素性記述を評価する。すなわち、CFG parsing により得られた構文木の候補の妥当性を素性記述を評価することで検証する。これを pipeline strategy と呼ぶ時もある。

**選択評価戦略 (selective unification)** 素性記述を、統語構造を制約するための記述と、意味計算のための記述に分離し、前者は規則適用のたびに評価し、後者は構文木が得られた段階でまとめて評価する[6]。

注釈の評価時期の選択は、句構造規則の粒度 (句構造規則が持つ構造的な制約の強さ) と密接に関連する。文法の粒度が粗い場合には、句構造規則だけでは、統語構造に関する制約が弱く無駄な部分木の発生が避けられないので、素性記述を早期に評価する必要がある (早期評価戦略)。しかし、ある程度、詳細化された句構造規則 (中粒度の文法) を用いれば、句構造規則たけでも、効率良く構文木の候補が発見でき、さらに素性記述の評価を遅延することにより無駄な単一化を回避できると予想される (最終評価戦略)。

しかし、文法の規模が非常に大きくなると、やはり、CFG parsing 途中段階で、ある程度の枝刈り (pruning) をする方が望ましいと予想される。そこで、コストの小さい枝刈りを実現する方法の一つとして、素性記述の一部を規則適用のたびに評価する方法が考えられた。これが最終評価戦略である[田代・永田, 92]。最終評価戦略は、無駄な部分木の発生を抑えながら、無駄な単一化を回避できる可能性がある。

### 2.4.2 実験と考察

上述の3つの素性記述評価戦略に関して、2つの規模の異なる文法 (m-set/e-set) を用いて、解析時間、ステップ数、ノード数、単一化の成功率などを調べた。結果を表2.3に示す。

- 最終評価の解析時間は、早期評価の解析時間の約50%である。
- 最終評価のステップ数は、早期評価のステップ数の約1.5倍である。
- 早期評価のノード数は、最終評価のノード数の約30%である。

これより、最終評価戦略が最も優れていることが分かる。最終評価が早期評価よりも効率的なのは、最終評価が無駄な単一化をほとんどしないためである。このことは、unify-desc の呼び出し回数、および、unify-desc の成功率を比較すれば明らかである。

[6]現在のところ、素性記述の差分、あるいは minimal cover set を計算するアルゴリズムが存在しないので、最後に全ての素性記述を評価するようにしている。そして、これが選択評価戦略の効率を悪くする問題点になっている。

表 2.3: 素性記述評価戦略と解析性能 (21 文, 平均 18.0 文字)

| | time | step | node | unify-f | uf-suc | unify-d | ud-suc |
|---|---|---|---|---|---|---|---|
| m-set.qdss.early | 18.9 (1.00) | 652.8 (1.00) | 53052.9 (1.00) | 1582.9 | 81.2 | 121.5 | 76.9 |
| m-set.qdss.late | 10.5 (0.56) | 879.0 (1.35) | 16098.0 (0.30) | 439.4 | 84.9 | 39.1 | 89.8 |
| m-set.qdss.select | 10.4 (0.55) | 725.8 (1.11) | 24462.7 (0.46) | 606.8 | 86.2 | 279.3 | 76.0 |
| e-set.qdss.early | 186.1 (1.00) | 1250.5 (1.00) | 488557.4 (1.00) | 10775.5 | 86.9 | 354.8 | 78.6 |
| e-set.qdss.late | 73.9 (0.40) | 1914.0 (1.53) | 151639.7 (0.31) | 2982.6 | 84.4 | 81.7 | 82.9 |
| e-set.qdss.select | 89.3 (0.48) | 1714.0 (1.37) | 219968.6 (0.45) | 4206.0 | 85.1 | 1675.1 | 86.5 |

選択評価は、ステップ数を減らすことには成功しているが、 node を消費し過ぎるので最終評価と同等以下の解析時間がかかるという問題がある。これに対しては、(1) 素性構造の「差分」[小暮, 92b]をとることにより、最後に単一化を行なう量を削減するか、または、(2) 素性構造の「一般化 (汎化)」[小暮, 92a]により、グラフの切り取り方を工夫して、もっとステップ数を削減するような工夫が必要だろう。また、e-set 文法を使った場合の選択評価における unify-desc の呼び出し回数が異常に多い。これは原因を追求する必要があるだろう。

**試験文 (21 文, 平均 18.0 文字)**

d01-06 会議に申し込みたいのですが
d01-14 ご住所とお名前をお願いします
d01-18 登録用紙を至急送らせていただきます
d02-04 今会議に申し込めば参加料はいくらですか
d03-08 ところで会議での公式言語は何ですか
d05-05 参加を取り消したいのですが
d05-09 既に登録料の八万五千円を振り込まれておられますね
d05-14 案内書にも書いていますが
d05-16 後日プログラムと予稿集をお送りいたします
d06-12 講演者も参加されるのですか
d06-20 参加料は当日集合場所でお支払いください
d07-03 会議で扱う話題に関して質問したいんですが
d07-08 申し訳ありませんがこちらでは専門的な質問にお答えできません
d08-09 六月三十日までに原稿の送付をお願いします
d09-11 京都駅からですとおよそ六千円かかります
d10-09 二人部屋の値段は九千五百円から六万円です
d10-11 どちらのホテルが会議場に近いですか
d10-12 京都プリンスホテルが会議場には近いんですが
d10-14 ホテルの手配もしていただけるのですか
d10-16 京都ホテルと京都プリンスホテルは予約できます
d10-29 お部屋をお取りできます

## 2.5 入力の形式 (文字と形態素)・辞書の構成

### 2.5.1 文字入力と形態素入力

本システムは、音声入力を考慮して、ラティス形式の入力を受け付けられるようになっている。しかし、通常は、一次元のデータしか用いていない。しかし、ラティス (実際にはグラフ構造である) の弧のラベルの与え方、チャートの初期化の方法、および、文法コンパイラにおける語彙規則の扱い[7]を変更することにより、次の二つの入力形式を扱えるようにしている。

- 文字列: 登 - 録 - 用 - 紙 - を - 至 - 急 - 送 - ら - せ - て - 頂 - き - ま - す
- 形態素列: 登録用紙 - を - 至急 - 送 - ら - せ - て頂 - き - ます

形態素入力の場合には、形態素区切りだけが与えられることを仮定している。品詞情報が利用できれば、より早期に曖昧性が解消できる場合も多いとは思うが、現在はそのようになっていない[8]。

### 2.5.2 主記憶辞書と二次記憶辞書

現在の日本語解析システムの辞書の大きさは、最大で 1500 語程度だが、今後も拡張されることが予想される。そこで、文法中の語彙規則を二次記憶に格納する枠組を用意した [松尾 ほか, 91b]。従って、現在の日本語解析システムは、次の二つの辞書構成が可能になっている。

- 句構造規則: 主記憶常駐, 語彙規則: 主記憶常駐
- 句構造規則: 主記憶常駐, 語彙規則: 二次記憶格納

なお、二次記憶版のシステムでは、一度、主記憶に読み込まれた語彙規則はキャッシュするようにしている。また、予め指定した語彙 (例えば、全ての助詞) を主記憶常駐にすることもできる。

### 2.5.3 実験と考察

次の 3 つのパーザに関して、2 つの規模の異なる文法を用いて、解析時間、ステップ数、ノード数、単一化の成功率などを調べた。結果を表 2.4に示す。

- 文字入力 (char)・二次記憶辞書 (off)
- 文字入力 (char)・主記憶辞書 (off)
- 形態素入力 (morph)・二次記憶辞書 (off)

表 2.4より次のようなことが分かる。

---

[7]文字列入力の時は、文字単位で予測テーブルを作成し、形態素入力の時は、形態素単位で予測テーブルを作成する。

[8]形態素入力モードは、元々、音声認識装置および形態素解析プログラムとの結合を前提として作られた[永田, 92] [松尾 ほか, 91b]。その際には、「形態素調整」と呼ばれるインタフェースの存在を仮定している。しかし、形態素区切りと品詞を調整する枠組を作ることはそれほど難しくないが、データの整合性を保守し続けるのは非常に困難である。従って、構文解析部が用いている文法を使って専用の形態素を作成することが今後必要になるだろう。

表 2.4: 文字入力と形態素入力 (15 文, 平均 18.8 文字)

| | time | step | node | unify-fs |
|---|---|---|---|---|
| m-set.qdss.late.off.char | 8.2 (1.00) | 922.3 (1.00) | 15815.6 (1.00) | 469.1 |
| m-set.qdss.late.on.char | 8.7 (1.06) | 922.3 (1.00) | 14939.4 (0.94) | 469.1 |
| m-set.qdss.late.off.morph | 6.4 (0.78) | 512.7 (0.55) | 15668.5 (0.99) | 462.5 |
| e-set.qdss.late.off.char | 82.0 (1.00) | 2012.7 (1.00) | 198172.4 (1.00) | 4087.0 |
| e-set.qdss.late.on.char | 85.9 (1.04) | 2012.7 (1.00) | 196465.8 (0.99) | 4087.0 |
| e-set.qdss.late.off.morph | 63.3 (0.77) | 884.9 (0.44) | 166459.3 (0.84) | 3455.7 |

- 形態素列入力の解析時間およびステップ数は、文字列入力の場合いに比べて、それぞれ、約 80% および約 50% に削減されている。しかし、単一化の回数および生成されたノード数にはあまり大きな変化がない。これは、中粒度文法では、形態論的な情報が句構造規則の中に記述されているので、形態素解析を省略することの効果は、主に CFG parsing の処理量の減少として現れるためだと思われる。
- m-set 文法と e-set 文法では、数字 (月日を除く) や住所 (行政区画名) の扱いが異なっている。また、一般に、e-set 文法の方がはるかに自由度が大きい。そのために、形態素入力とすることによる構文解析の負荷の軽減の度合は、e-set の方が m-set より大きいように見える。
- 辞書主記憶常駐版と辞書二次記憶配置版では、二次記憶版の方がわずかに速い。これは、単一化構文解析プログラムの効率が使用可能な実記憶の量に強く依存するので、主記憶が空いている分だけ二次記憶版の方が有利なのだと思われる。

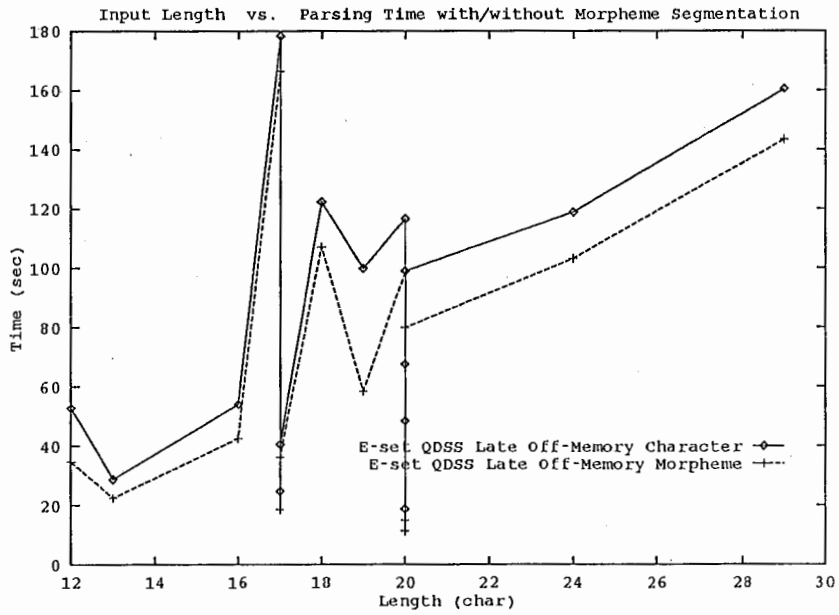


図 2.6: 入力文の長さと解析時間 (e-set)

次に、文字入力と形態素入力に関して、e-set 文法を用いた場合の入力文の長さと解析時間の関係を図 2.6に示す。図 2.6より次のようなことが分かる。

- 入力文が長くなるほど、形態素入力とすることによる構文解析の負荷の削減の度合が大きいように見える。

試験文 (15 文, 平均 18.8 文字)

```
d01-15  (m) 住所 は 大阪市 北区 茶屋町 二十三 です
        (e) 住所 は 大阪 市 北 区 茶屋町 二 十 三 です
d01-18  (m) 登録用紙 を 至急 送 ら せ ていただ き ます
        (e) 登録用紙 を 至急 送 ら せ ていただ き ます
d02-04  (m) 今 会議 に 申し込 め ば 参加料 は いくら です か
        (e) 今 会議 に 申し込 め ば 参加料 は いくら です か
d03-08  (m) ところで 会議 で の 公式言語 は 何 です か
        (e) ところで 会議 で の 公式言語 は 何 です か
d04-13  (m) 住所 は 大阪市 東区 玉造 二丁目 二十七 の 七 です
        (e) 住所 は 大阪 市 東 区 玉造 二 丁目 二 十 七 の 七 です
d05-09  (m) 既に 登録料 の 八万五千 円 を 振り込 ま れ ておられ ます ね
        (e) 既に 登録料 の 八 万 五 千 円 を 振り込 ま れ ておられ ます ね
d05-14  (m) 案内書 に も 書 い て い ます が
        (e) 案内書 に も 書 い て い ます が
d05-16  (m) 後日 プログラム と 予稿集 を お送り いた し ます
        (e) 後日 プログラム と 予稿集 を お送り いた し ます
d06-12  (m) 講演者 も 参加 さ れ る の です か
        (e) 講演者 も 参加 さ れ る の です か
d07-08  (m) 申し訳ありませんが こちら で は 専門的 な 質問 に お答え でき ません
        (e) 申し訳ありませんが こちら で は 専門的 な 質問 に お答え でき ません
d07-12  (m) それでは 早急に その 案内書 を 送 っ てください
        (e) それでは 早急に その 案内書 を 送 っ てください
d08-09  (m) 六月 三十日 までに 原稿 の 送付 を お願い し ます
        (e) 六月 三十日 までに 原稿 の 送付 を お願い し ます
d10-09  (m) 二人部屋 の 値段 は 九千五百 円 から 六万 円 です
        (e) 二人部屋 の 値段 は 九 千 五 百 円 から 六 万 円 です
d10-11  (m) どちら の ホテル が 会議場 に 近 い です か
        (e) どちら の ホテル が 会議場 に 近 い です か
d10-14  (m) ホテル の 手配 も し ていただけ る の です か
        (e) ホテル の 手配 も し ていただけ る の です か
```

#### 2.5.3.1 形態素入力による単一化の負荷の軽減

表 2.4に示したデータには、unify-desc の呼び出し回数および成功率がなかった。そこで、3つの文に関して、データを取り直した結果を表 2.5に示す。

表 2.5: unify-desc の呼び出し回数と成功率 (3 文, 平均 17.0 文字)

| | time (sec) | chart steps | dag nodes | unify-fs | | unify-desc | |
|---|---|---|---|---|---|---|---|
| | | | | calls | success | calls | success |
| m-set.qdss.late.char | 6.8 (1.00) | 1021.3 (1.00) | 10217.0 (1.00) | 378.7 | 81.3% | 50.3 | 85.6% |
| m-set.qdss.late.morph | 4.9 (0.72) | 581.0 (0.57) | 10846.7 (1.06) | 397.3 | 80.9% | 50.3 | 85.6% |
| e-set.qdss.late.char | 83.8 (1.00) | 2190.7 (1.00) | 237636.0 (1.00) | 4289.0 | 90.5% | 154.7 | 74.9% |
| e-set.qdss.late.morph | 57.6 (0.69) | 1038.0 (0.47) | 169229.0 (0.71) | 3166.0 | 89.5% | 112.0 | 83.9% |

m-set の場合には形態素入力と文字入力で単一化の成功率に差はないが、e-set の場合には、形態素入力とする (形態素解析を分離する) ことにより、unify-desc の成功率が約 10% 向上している。従って、形態素解析を分離することにより、無駄な単一化を削減することができることがわかる。

この傾向は、日本語解析システムが大規模になればなるほど顕著になると予想される。従って、精度が高く、適用範囲が広い、高速な形態素解析プログラムを別途開発することが望まれる。また、今後、対象範囲を広げても、形態素解析が分離されていれば、現在の e-set よりも大きく性能が劣ることはないのではないかと、報告者 (永田) は、個人的に思っている。

**試験文 (3 文, 平均 17.0 文字)**

d02-04 今会議に申し込めば参加料はいくらですか
d04-13 住所は大阪市東区玉造二丁目二十七の七です
d05-14 案内書にも書いていますが

# 第 3 章

# 単一化プログラム (ユニファイア)

## 3.1 単一化プログラムの概要

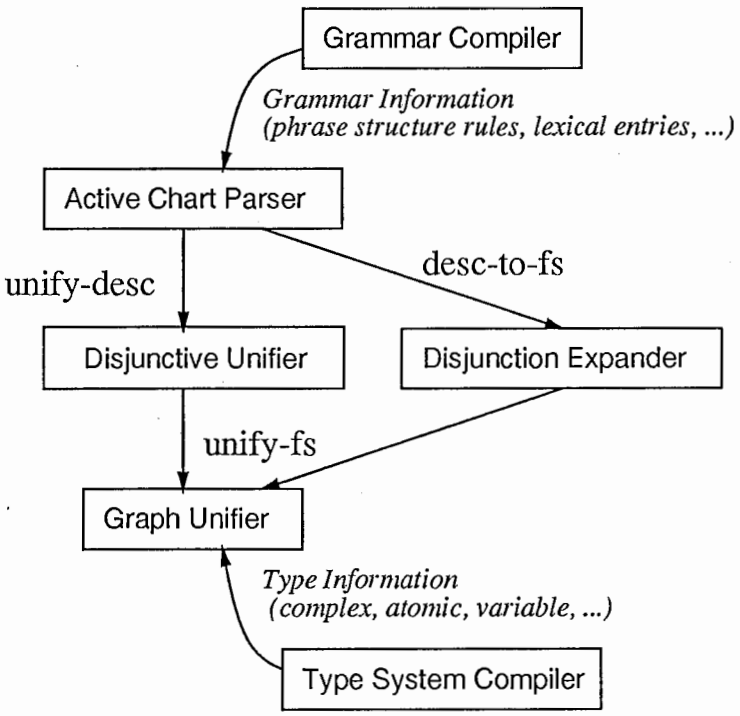


図 3.1: ユニファイアの構成

図 3.1に、日本語解析システムで用いられている単一化プログラム (ユニファイア) の構成を示す。ユニファイア (unifier) は、基本的に、次の二つのモジュールから構成されている。

素性構造単一化 (feature structure unifier) 素性構造の単一化を行なう。グラフ単一化 (graph unifier) とも呼ばれる。

選言的素性記述単一化 (disjunctive unifier) 選言を含む素性記述の単一化を行なう。

その他、ユニファイアに関連するモジュールとしては、次の二つが重要である。

選言的素性構造展開 (disjunction expander) 選言を含む素性記述を、選言を含まない素性構造の集合に展開する。このモジュールは、構文解析終了後に用いられる。内部では、素性構造単一化プログラムを利用する。

**タイプシステムコンパイラ (type system compiler)** タイプ記述言語で定義されたタイプの階層と、各タイプごとに定義された手続きの情報を、グラフ単一化プログラムの中に埋め込む。

日本語解析システムで用いる基本的なデータ構造は、タイプ付き素性構造[Aït-Kaci, 1986]である。また、日本語解析システムの基本的な演算である単一化は、タイプ付き素性構造の選言、および否定が扱えるように拡張されている[小暮・加藤, 88] [小暮, 91]。また、タイプ付き素性構造が循環を含むことも許している[1]。

ただし、素性構造に否定を導入することは、実は理論的には非常に難しい。現在実装されているのは、token-identity (Lisp でいう eq) の否定であって、type-identity (Lisp でいう equal) の否定ではない。ところが、この token-identity の否定は、文法記述ではあまり使い道がない。また、diff スロットを使った実装はあまり効率が良いとは言えない。この反省から、タイプ付き素性構造の否定の新たな取り扱い[Kogure, 1992]が提案されている。

タイプ・循環・否定に関する拡張は、グラフ単一化プログラムの中で行なわれている。これに対して、選言に関する拡張は、グラフ単一化プログラムの外側に、選言的な制約の充足問題を解くモジュール (disjunctive constraint solver) を作ることで実現している。

以下では、素性構造 (グラフ) 単一化アルゴリズムに関する評価実験の結果を述べる。選言的素性構造の単一化アルゴリズムについては、何も改良が行なわれていないので、本報告では説明しない。しかし、選言的素性構造の複雑さは、実行効率に大きく影響するので、選言的素性構造の単一化アルゴリズムで用いられているデータ構造については、付録Aで述べる。

[1]一般的に、素性構造は有向非循環グラフ (DAG, Directed Acyclic Graph) で表せるので、素性構造と DAG は同義語として使われている。しかし、我々が用いる素性構造は循環構造を持つ可能性があるので、本当は DAG ではない。

## 3.2 選言を含まない素性構造の単一化

### 3.2.1 素性構造単一化アルゴリズム

ここでは、次の三つの素性構造単一化アルゴリズムを実装した。アルゴリズムの詳細については、[永田, 92]または各文献を参照して欲しい。

**非破壊的グラフ単一化 (Non-Destructive Graph Unification)** 非破壊的グラフ単一化アルゴリズムでは、素性構造を逐次的 (incremental) に複製することによって、早期複製および過剰複製を減少させる (逐次複製) [Wroblewski, 87]。

**準破壊的グラフ単一化 (Quasi-Destructive Graph Unification)** 単一化処理を、無矛盾性検査過程と出力グラフ作成過程に分けることによって、誤りを早期に発見できた場合には、無駄な複製を回避する (逐次複製 + 遅延複製)。[Tomabechi, 91]。

**構造共有型の準破壊的グラフ単一化 (QDGU with Structure Sharing)** 出力グラフ作成過程において、あるノードから下の部分グラフに変更が加えられていなければ、これをそのまま出力グラフに利用 (構造共有) することによって、無駄な複製を回避する (逐次複製 + 遅延複製 + 構造共有)[Tomabechi, 92]。

### 3.2.2 実験と考察

上の 3 つの単一化アルゴリズムを用いたパーザについて、 m-set 文法を用いて、解析時間、ステップ数、ノード数などを調査した。結果を表 3.1に示す。

表 3.1: 単一化アルゴリズムの違いによる解析性能の差異 (40 文, 平均 20.3 文字)

| | time (sec) | step | dag nodes |
|---|---|---|---|
| m-set.ndgu.late | 27.1 (1.00) | 1002.1 | 200581.2 (1.00) |
| m-set.qdgu.late | 21.5 (0.79) | 991.4 | 187717.0 (0.94) |
| m-set.qdss.late | 13.5 (0.50) | 991.4 | 30801.2 (0.15) |

表 3.1より次のことが分かる。

- ndgu を基準に考えると、 qdgu は遅延複製により 79% に、 qdss は遅延複製と構造共有により 50% にまで、解析時間が短くなっている
- qdgu や qdss における遅延複製の効果は、使用ノード数の減少と、 unify-internal-1 (unify-fs の内部で再帰的に呼び出される関数) の呼び出し数の減少により確認できる (3.2.2.1節参照)。

構造共有型の準破壊的グラフ単一化アルゴリズム (qdss) が、解析時間およびメモリ使用量の観点から見て、最も優れている。しかし、 qdgu や qdss における遅延複製の効果は、単一化の成功率、すなわち記述された文法の性質に大きく依存する。単一化の成功率が低い粗粒度文法では、遅延複製の効果が大きいのに対して、単一化の成功率が高い中粒度文法では、遅延複製の効果が相対的に低くなる (5章参照)。

また、qdss では、構造共有により作成されるノード数が劇的に減少する (元の 15%)。それに比べて、解析時間が 50% にしかならないのは、グラフを 2 回巡回することのオーバーヘッドが相当大きいことを示している[2]。

ndgu と qdss (qdgu) で、ステップ数が違う原因は良く分らない。本来、unifier を置き換えても、parser の動作は変わらないはずである。しかし、これまでのテストでは解析結果が異なっていた事例がないのと、ステップ数は少ないに越したことはないという楽観的理由で、この件に関しては、深く追求していない。これは、将来問題を起こすかもしれない。

**試験文 (40 文, 平均 20.3 文字)**

d01-06 会議に申し込みたいのですが
d01-07 どのような手続きをすればよろしいのでしょうか
d01-13 それでは登録用紙をお送りいたします
d01-14 ご住所とお名前をお願いします
d01-15 住所は大阪市北区茶屋町二十三です
d01-18 登録用紙を至急送らせていただきます
d02-03 会議の参加料について教えていただきたいのですが
d02-04 今会議に申し込めば参加料はいくらですか
d02-07 来月お申し込みになりますと四万円です
d02-08 参加料には予稿集代と歓迎会費が含まれています
d02-11 今回は割引を行なっておりません
d02-13 参加料はどのようにお支払いしたらよいのですか
d02-15 案内書に記載されている口座番号に振り込んでください
d02-20 分からない点がございましたらいつでもお聞きください
d03-03 会議に論文を発表したいと思っているのですが
d03-05 今回の会議は通訳電話に関連する広範な研究分野を含んでいます
d03-06 言語学や心理学を専攻する方にも参加していただく予定です
d03-08 ところで会議での公式言語は何ですか
d03-10 わたしは日本語が全然分からないのですが
d03-11 発表が日本語で行なわれる場合英語への同時通訳はあるのですか
d04-02 会議について詳しいことを教えてください
d04-07 会議は八月二十二日から二十五日まで京都国際会議場で開催されます
d04-09 発表を希望されるのでしたら三月二十日までに要約を提出してください
d04-10 会議の案内書をお送りいたしますのでそれをご覧ください
d04-11 失礼ですがお名前とご住所をお願いいたします
d04-13 住所は大阪市東区玉造二丁目二十七の七です
d04-15 電話番号もお聞きしたいのですが
d04-21 それではよろしくお願いします
d05-04 わたしは会議に申し込みをした者です
d05-05 参加を取り消したいのですが
d05-06 お名前をお伺いできますでしょうか
d05-09 既に登録料の八万五千円を振り込まれておられますね
d05-11 そうです
d05-12 登録料を払い戻していただけますか
d05-14 案内書にも書いていますが
d05-15 九月二十七日以後の取り消しに対する払い戻しはできません
d05-16 後日プログラムと予稿集をお送りいたします
d05-17 では誰かがわたしの代わりに参加することはできますか

[2]qdss は循環構造に関与するノードを全て複製してしまうという欠点を持っている。これを改良するために、(1) 無矛盾性検査、(2) 循環検査、(3) 出力グラフ生成の 3 つのパスから構成されるアルゴリズムを実装したが、グラフを巡回することのオーバヘッドのために高速化につながらなかった[高橋 ほか, 92]。

d05-19 代理人が参加する場合はあらかじめこちらまでお知らせください
d05-21 代理人が決まりましたらお知らせいたします

#### 3.2.2.1 遅延複製の効果

グラフを巡回するために unify-fs の内部で再帰的呼び出される unify-internal-1 という関数の呼び出し回数を調べれば、遅延複製の効果が分かる。しかし、表 tab: 単一化アルゴリズムの実験では、これを調べていなかったので、3 つの文に関してデータを取り直した。結果を表 3.2に示す。

表 3.2: 非破壊的グラフ単一化と準破壊的グラフ単一化 (3 文, 平均 15.7 文字)

| | time | step | node | unify-fs | | unify-desc | | unify-fs |
|---|---|---|---|---|---|---|---|---|
| | | | | calls | success | calls | success | internal calls |
| coarse.qdss.early | 13.3 | 154.0 | 40295.7 | 1701.0 | 84.4% | 72.7 | 66.7% | 13174.0 |
| coarse.ndgu.early | 31.9 | 160.0 | 337384.7 | 1701.0 | 84.4% | 72.7 | 66.7% | 14582.0 |
| coarse.qdss.late | 17.9 | 300.3 | 52473.3 | 2072.7 | 84.4% | 74.3 | 78.2% | 15693.7 |
| coarse.ndgu.late | 41.0 | 307.0 | 406775.0 | 2072.7 | 84.4% | 74.3 | 78.2% | 17127.7 |
| medium.qdss.early | 16.1 | 502.0 | 44105.0 | 1521.3 | 88.9% | 67.7 | 83.4% | 16187.0 |
| medium.ndgu.early | 41.9 | 511.3 | 371974.7 | 1521.3 | 88.9% | 67.7 | 83.4% | 17730.3 |
| medium.qdss.late | 9.1 | 604.3 | 21739.3 | 664.3 | 90.8% | 32.0 | 88.3% | 7652.0 |
| medium.ndgu.late | 18.3 | 615.7 | 150248.3 | 664.3 | 90.8% | 32.0 | 88.3% | 8475.3 |

この表より、グラフ単一化プログラムが unify-fs の内部で再帰的に呼ばれる回数は、qdss では ndgu の 90.3% になっていることが分かる。これは、無矛盾性検査過程を持つ qdss の方が、グラフを巡回する量が少ないことを表す。

試験文 (3 文, 平均 15.7 文字)

d04-15 電話番号もお聞きしたいのですが
d05-14 案内書にも書いていますが
d05-21 代理人が決まりましたらお知らせいたします

# 第 4 章

# 文法の規模と粒度

## 4.1 文法の粒度と素性記述評価戦略

### 4.1.1 句構造規則の粒度

単一化文法は記述の自由度が高いので、文法的な制約は、1) 句構造規則、2) 句構造規則の注釈, 3) 語彙項目、のいずれにも記述できる。特に、単一化はパターン照合のために多用されるが、パターン照合は非終端記号を適当に設定すれば句構造規則でも記述できる。従って、単一化に基づく文法を設計する際には、文法の保守性と解析の計算効率のトレードオフを考慮して、句構造規則の詳細化の度合 (「句構造規則の粒度」と呼ぶことにする) を決めることが重要である。また、句構造規則を詳細化すれば、句構造規則の解析に要する計算コストが増加するから、単一化の計算コストの減少分がこれを上回るように句構造規則の粒度を設定しなければ、全体の解析効率は向上しない。

ここでは、粒度の異なる次の二つの文法を用意した。文法に関する様々な数値に関しては、付録 Cを参照して欲しい。

**粗粒度文法 (coarse)** 一般的な句構造規則と詳細な注釈から構成される (句構造規則 22 個, 語彙項目 438 個)。

**中粒度文法 (medium)** 詳細な句構造規則から構成される (句構造規則 163 個, 語彙項目 504 個)。

### 4.1.2 実験と考察

粒度の異なる 2 つの文法 (coarse/medium) と、2 つの異なる素性記述評価戦略 (early/late) の組み合わせについて、28 個の試験文を用いて、解析時間、ステップ数、ノード数、単一化の成功率などを調べた。結果を表 4.1に示す。

表 4.1より次のようなことが分かる。

- 粗粒度文法では、early unification の方が解析時間が短い。
- 中粒度文法では、late unification の方が解析時間が短い。
- 中粒度文法と最終評価の組み合わせの解析時間は、粗粒度文法と早期評価の組み合わせの解析時間の半分以下である。

従って、中粒度文法 (medium) と最終評価戦略 (late) の組合せが最も効率が良いと言える。単一化 (unify-desc) の呼び出し回数と成功率を見れば分かるように、medium と late の組合せ

表 4.1: 文法の粒度と素性記述評価戦略 (28 文, 平均 18.1 文字)

| | time (sec) | chart steps | dag nodes | unify-fs | | unify-desc | |
|---|---|---|---|---|---|---|---|
| | | | | calls | success | calls | success |
| coarse.qdss.early | 16.6 (1.00) | 179.8 (1.00) | 51046.6 (1.00) | 1881.3 | 81.1% | 103.1 | 65.3% |
| coarse.qdss.late | 29.1 (1.75) | 338.4 (1.88) | 47791.8 (0.94) | 1607.0 | 81.9% | 75.6 | 73.0% |
| medium.qdss.early | 18.0 (1.08) | 638.5 (3.55) | 48082.8 (0.94) | 1628.8 | 82.5% | 107.0 | 83.7% |
| medium.qdss.late | 7.8 (0.47) | 792.2 (4.41) | 16128.3 (0.32) | 495.7 | 83.2% | 34.4 | 91.0% |

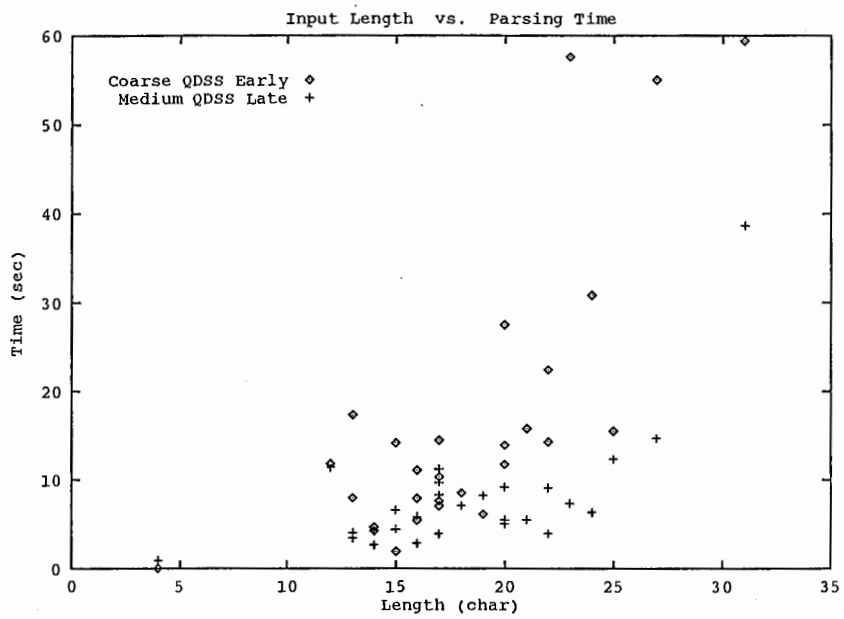


図 4.1: coarse.early.qdss と medium.late.qdss の解析時間の比較

は、成功率が約 90% あり、ほとんど無駄な単一化が行なわれていない。これが medium と late の組合せの効率が良い理由である。

句構造規則に基づいた構文解析の計算量と、素性構造の単一化の計算量には、トレードオフの関係がある。最終評価戦略は、チャートパーザのステップ数は多いが、作成するノード数が少ない。現在の計算機では、ノードを作成する処理、すなわち、メモリを割り当てる処理のオーバヘッドが非常に大きいので、ノード数を減らすことが処理の効率化につながっている。

図 4.1に、coarse と early の組み合わせと、medium と late の組み合わせに関して、入力文の長さと解析時間の関係を示す。これを見ると、coarse と early の組み合わせは、同じ長さの文に対する解析時間のバラツキが大きいのに対して、medium と late の組み合わせの解析時間は、入力文の長さとの相関が大きい。これは、medium と late の組み合わせでは、CFG parsing が中心的な計算になるのに対して、coarse と early の組み合わせは、unification が中心的な計算になっているためだと思われる。

文脈自由言語の範囲で記述された句構造規則は、入力長 $N$ に対して $o(N^3)$ の効率的な解析アルゴリズムが存在するのに対して、選言的素性構造の単一化の計算量は、最悪の場合、選言の数の指数に比例する。従って、今後の文法の大規模化を考えると、medium と late の組み合わ

せを用いて、CFG Parsing の効率性を利用する方が有利であると予想される。

試験文 (28 文, 平均 18.1 文字)

d01-06 会議に申し込みたいのですが
d01-07 どのような手続きをすればよろしいのでしょうか
d01-13 それでは登録用紙をお送りいたします
d01-14 ご住所とお名前をお願いします
d01-15 住所は大阪市北区茶屋町二十三です
d01-18 登録用紙を至急送らせていただきます
d02-03 会議の参加料について教えていただきたいのですが
d02-07 来月お申し込みになりますと四万円です
d02-11 今回は割引を行なっておりません
d02-13 参加料はどのようにお支払いしたらよいのですか
d03-08 ところで会議での公式言語は何ですか
d04-02 会議について詳しいことを教えてください
d04-07 会議は八月二十二日から二十五日まで京都国際会議場で開催されます
d04-11 失礼ですがお名前とご住所をお願いいたします
d04-13 住所は大阪市東区玉造二丁目二十七の七です
d04-15 電話番号もお聞きしたいのですが
d04-21 それではよろしくお願いします
d05-04 わたしは会議に申し込みをした者です
d05-05 参加を取り消したいのですが
d05-06 お名前をお伺いできますでしょうか
d05-09 既に登録料の八万五千円を振り込まれておられますね
d05-11 そうです
d05-12 登録料を払い戻していただけますか
d05-14 案内書にも書いていますが
d05-15 九月二十七日以後の取り消しに対する払い戻しはできません
d05-16 後日プログラムと予稿集をお送りいたします
d05-17 では誰かがわたしの代わりに参加することはできますか
d05-21 代理人が決まりましたらお知らせいたします

## 4.2 文法の規模と素性記述評価戦略

### 4.2.1 文法の規模

ここでは、規模の異なる。次の 3 つの文法を用意した。文法に関する様々な数値に関しては、付録 Cを参照して欲しい。

- モデル会話 AB1-5 の文法 (medium) (句構造規則 163 個, 語彙項目 504 個)
- モデル会話 AB1-10 の文法 (m-set) (句構造規則 194 個, 語彙項目 687 個)
- 機能試験文 (600 個) の文法 (e-set) (句構造規則 223 個, 語彙項目 1776 個)

### 4.2.2 実験と考察

上述の 3 つの文法 (medium/m-set/e-set) と、2 つの素性記述戦略 (early/late) について、7 個の試験文を用いて、解析時間、ステップ数、ノード数、単一化成功率などを調べた。結果を表 4.2に示す。

表 4.2: 文法の規模と素性記述評価戦略 (7 文, 平均 15.1 文字)

| | time (sec) | | chart steps | | dag nodes | | unify-fs | | unify-desc | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | calls | success | calls | success |
| medium.qdss.early | 12.3 | (2.12) | 567.0 | (0.81) | 32323.6 | (2.97) | 1186.6 | 83.4% | 78.6 | 84.7% |
| medium.qdss.late | 5.8 | (1.00) | 699.6 | (1.00) | 10882.3 | (1.00) | 358.0 | 85.0% | 27.9 | 91.1% |
| m-set.qdss.early | 13.5 | (2.32) | 586.6 | (0.84) | 35264.7 | (3.24) | 1226.0 | 82.7% | 92.6 | 76.6% |
| m-set.qdss.late | 8.5 | (1.47) | 835.6 | (1.19) | 9608.0 | (0.88) | 315.6 | 86.6% | 30.1 | 87.6% |
| e-set.qdss.early | 192.5 | (33.18) | 1177.6 | (1.68) | 458014.7 | (42.09) | 11058.9 | 87.6% | 304.6 | 77.5% |
| e-set.qdss.late | 64.5 | (11.12) | 1776.6 | (2.54) | 117418.4 | (10.79) | 3107.3 | 82.9% | 76.7 | 80.6% |

文法の規模が大きくなっても、late の方が early よりも解析時間が短い。m-set と e-set を比較すると、文法が大きくなればなるほど late の方が有利なように見える。従って、文法の規模が大きくなっても、最終評価戦略の方が早期評価戦略よりも効率が良いことがわかる。

文法の規模の拡大に伴って、解析時間が急激に増大する。これは、組み合わせ爆発 (combinatorial explosion) によるステップ数の増加が原因であると思われる。また、このことは、e-set のグラフ単一化 (unify-fs) の成功率がそれほど低下していないのに比べて、e-set の選言的単一化 (unify-desc) の成功率が大きく低下していることによっても裏付けられる。この組み合わせ爆発に対処するために、アジェンダ制御戦略の研究をさらに進める必要がある。

試験文 (7 文, 平均 15.1 文字)

d01-06 会議に申し込みたいのですが
d01-14 ご住所とお名前をお願いします
d01-18 登録用紙を至急送らせていただきます
d03-08 ところで会議での公式言語は何ですか
d05-05 参加を取り消したいのですが
d05-14 案内書にも書いていますが
d05-16 後日プログラムと予稿集をお送りいたします

# 第 5 章

# 単一化アルゴリズムと素性記述評価戦略の組み合わせ

## 5.1 文法粒度・単一化アルゴリズム・素性記述評価戦略の相関

### 5.1.1 遅延複製と最終評価

準破壊的グラフ単一化アルゴリズム (qdgu/qdss) は、二つの素性構造の非共通部分を遅延複製することにより、素性構造の無駄な複製を削減する。一方、素性記述の最終評価は、構文木が完成した時にだけ素性記述を評価することにより、素性構造の無駄な複製を削減する。両者は、無駄な複製を削減するという点では同じだが、その実現方法が異なるし、削減している対象も異なると予想される。

また、準破壊的グラフ単一化および素性記述最終評価の有効性は、対象とする文法の性質にも大きく依存する。準破壊的グラフ単一化は、単一化の成功率が低い場合に有効であるのに対して、素性記述最終評価は、単一化の成功率が高い場合に有効である。

文法の粒度、単一化アルゴリズム、素性記述評価戦略の 3 項目を変えることによる実行効率の改善の試みは、相補的な側面と競合する側面が存在するので、これらの間の関係を明らかにするために、以下のような実験を行なった。

### 5.1.2 実験と考察

文法の粒度 (coarse/medium)、単一化アルゴリズム (ndgu/qdss)、素性記述評価戦略 (early/late) を、それぞれ変えたパーザについて、解析時間、ステップ数、ノード数を調べた。その結果を表 5.1に示す。

表 5.1: 文法粒度・単一化アルゴリズム・素性記述評価戦略 (28 文, 平均 18.1 文字)

| | time (sec) | chart steps | dag nodes |
|---|---|---|---|
| coarse.ndgu.early | 46.0 (1.00) | 189.3 (1.00) | 465211.1 (1.00) |
| coarse.ndgu.late | 54.0 (1.17) | 341.6 (1.80) | 351168.6 (0.75) |
| medium.ndgu.early | 41.8 (0.91) | 645.1 (3.41) | 359607.6 (0.77) |
| medium.ndgu.late | 15.8 (0.34) | 801.1 (4.23) | 114315.0 (0.25) |
| coarse.qdss.early | 15.2 (0.33) | 179.8 (0.94) | 51046.6 (0.11) |
| coarse.qdss.late | 28.0 (0.61) | 338.4 (1.79) | 47791.8 (0.10) |
| medium.qdss.early | 16.7 (0.36) | 638.5 (3.37) | 48082.8 (0.10) |
| medium.qdss.late | 7.2 (0.16) | 792.2 (4.18) | 16128.3 (0.03) |

- 単一化アルゴリズムとして ndgu を用いた場合には、coarse. と early の組み合わせから medium と late の組み合わせにすることによって、2.91 倍速くなる。
- 単一化アルゴリズムとして qdss を用いた場合には、coarse と early の組み合わせから medium と late の組み合わせにすることによって、2.11 倍速くなる。

各条件下で作成されたノード数を比べると、準破壊的グラフ単一化 (qdgu/qdss) が削減するノードと、素性記述最終評価戦略が削減するノードには、共通部分と非共通部分があり、両者を組み合わせると、各々の削減効果の積にはならないが、かなり大きな削減効果がある。

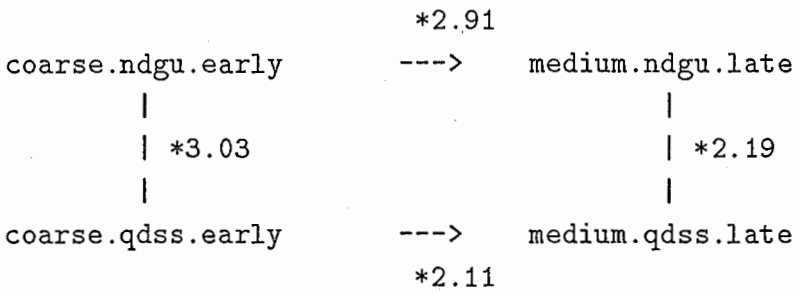


図 5.1: 句構造規則詳細化・複製遅延・最終評価の組合せの効果

図 5.1 に、句構造規則の詳細化 (coarse → medium)、素性構造の複製遅延と構造共有 (ndgu → qdss)、および、素性記述の最終評価 (early → late) によって、解析速度 (解析時間の逆数) がどのように改善されたかを示す。

coarse.ndgu.early の組み合わせにおいて、単一化アルゴリズムを ndgu から qdss にすると 3.03 倍速くなる。また、coarse.ndgu.early の組み合わせから、単一化アルゴリズムは ndgu のままで、medium.ndgu.late の組み合わせにすると、2.91 倍速くなる。そして、最終的には、従来のシステム (coarse.ndgu.early) から、現在のシステム (medium.qdss.late) に至るまでの改良により、6.39 倍の高速化が達成されたことがわかる。

**試験文 (28 文, 平均 18.1 文字)**

d01-06 会議に申し込みたいのですが
d01-07 どのような手続きをすればよろしいのでしょうか
d01-13 それでは登録用紙をお送りいたします
d01-14 ご住所とお名前をお願いします
d01-15 住所は大阪市北区茶屋町二十三です
d01-18 登録用紙を至急送らせていただきます
d02-03 会議の参加料について教えていただきたいのですが
d02-07 来月お申し込みになりますと四万円です
d02-11 今回は割引を行なっておりません
d02-13 参加料はどのようにお支払いしたらよいのですか
d03-08 ところで会議での公式言語は何ですか
d04-02 会議について詳しいことを教えてください
d04-07 会議は八月二十二日から二十五日まで京都国際会議場で開催されます
d04-11 失礼ですがお名前とご住所をお願いいたします
d04-13 住所は大阪市東区玉造二丁目二十七の七です
d04-15 電話番号もお聞きしたいのですが
d04-21 それではよろしくお願いします
d05-04 わたしは会議に申し込みをした者です
d05-05 参加を取り消したいのですが
d05-06 お名前をお伺いできますでしょうか

d05-09 既に登録料の八万五千円を振り込まれておられますね
d05-11 そうです
d05-12 登録料を払い戻していただけますか
d05-14 案内書にも書いていますが
d05-15 九月二十七日以後の取り消しに対する払い戻しはできません
d05-16 後日プログラムと予稿集をお送りいたします
d05-17 では誰かがわたしの代わりに参加することはできますか
d05-21 代理人が決まりましたらお知らせいたします

# 第 6 章

# おわりに

## 6.1 結論

パーザに関して

- アジェンダ制御戦略では、stack-active が一番効率が良い。しかし、ヒューリスティクスな手法と確率的な手法の優劣については、解析時間と精度を評価尺度として、さらに研究する必要がある。
- 弧を共有するパーザの方が、弧を共有しないパーザより効率が良い。しかし、文脈依存の要素を扱う場合に関しては、さらに研究する必要がある。
- 素性記述の評価時期に関しては、最終評価戦略が最も効率が良い。しかし、選択評価戦略が、今後の改良次第では、最終評価戦略に優る可能性がある。
- 文法が大規模化するほど、形態素入力による構文解析の負荷の軽減は大きくなる。

ユニファイアに関して

- 構造共有型の準破壊的グラフ単一化アルゴリズムは、非破壊的グラフ単一化アルゴリズムよりも、高速、かつ、メモリの消費量が少ない。

文法に関して

- 中粒度文法と素性記述最終評価戦略の組み合わせは、その他のどの粒度および素性記述評価戦略の組合せよりも効率が良い。

パーザ・ユニファイア・文法の組み合わせについて

- 素性記述最終評価、構造共有型準破壊的グラフ単一化、中粒度文法の各々の効率の改善効果は組み合わせることができる。これら 3 つの改良を組み合わせた時、改良後のパーザは、改良前のパーザに比べて 6 倍以上速い。

ひとこと

P. Norvig の Lisp の教科書[Norvig, 92]では、プログラムを高速化する手法として、次のような項目を挙げている。

- 以前の計算結果を覚えておく (Memoization)。
- 先に計算できることは計算しておく (Compiling)

- 計算を遅延する (Delaying Computation)。
- 適切なデータ構造とアルゴリズムを選ぶ (Indexing)。
- プロファイルを取って何を最適化すべきかを見極める。

今にして思えば、単一化結果のキャッシュや素性構造の構造共有は memoization の例であるし、素性構造の遅延複製や素性記述の最終評価は delaying の例である。従って、我々が試行錯誤の末に辿り着いたのは、実は、非常に基本的な「ごくあたりまえ」の方法だったのかもしれない。研究とは、そういうものなのだろう。

## 6.2 今後の課題

日本語解析システムが、現在、抱えている問題点と、今後、取り組むべき課題について、効率・精度・頑健さという観点からまとめてみる。

効率 (efficiency)

- 確率的なアジェンダ制御。これはヒューリスティクスを用いる方法 (mental-os) よりも優れているか?
- 効率の良い弧の枝刈り方法 (edge pruning)。素性構造の汎化と差分を利用して、選択評価戦略 (制約を部分的に適用するアプローチ) を改良する案が有力。
- 選言的素性構造の単一化アルゴリズムの改良。制約が不足した記述 (under-specified constraitns) に対して解を求めようとする場合に問題が多い。独立性の利用が鍵だろう。成功すれば、大幅な効率改善が達成できるだろう。
- 選言的素性構造から素性構造を一つずつ読み出すアルゴリズムの改良。素性構造レベルの曖昧性の解消の問題も含んでいる。
- 精度が良く、効率の良い、形態素解析プログラムの実装。日本語解析が利用可能なインタフェースが必要。
- 冗長な記述を削除したり、効率の良い記述に変換する最適化文法コンパイラ。素性構造の汎化と差分などの演算と密接な関係がある。

精度 (accuracy)

- 句構造レベルの曖昧性の解消 and/or 音声認識のための「文脈依存の言語モデル」。これと弧の共有をどうやって両立させるか?
- 素性構造レベルの曖昧性の解消 (選言の扱い)。確率的な優先制御は可能か?

頑健さ (robustness)

- 入力文をカバーする文法的部分列の出力。自由発話を扱うために基礎となるだろう。

効率・精度・頑健さのいずれの問題も、確率的な言語モデルを (半) 自動的に作成できるかどうかが鍵だろう。今後、コーパスに基づく統計的なアプローチなどにより、これらの問題が形式化され、解決されることを切に望む。

# 参考文献


[茨木, 92] 茨木俊秀: 組合せ問題とその複雑さ, 人工知能学会研究会資料, SIG-F/H/K/S/I-9202-3, 1992.

[小倉ほか, 89] 小倉健太郎, 坂野俊哉, 保坂順子, 森元逞: 音声言語日英翻訳実験システム (SL-TRANS), ATR テクニカルレポート TR-I-0102, 1989.

[竹沢ほか, 93] 竹沢寿幸ほか (14 名): ATR 音声翻訳システム ASURA の実装, ATR テクニカルレポート TR-I-0303, 1993.

[小暮・加藤, 88] 小暮潔, 加藤進: 素性構造とその単一化アルゴリズムに関する検討, ATR テクニカルレポート TR-I-0032, 1988.

[小暮, 88] 小暮潔: 解析過程の制御を考慮した句構造文法解析機構の検討, ATR テクニカルレポート TR-I-0064, 1988.

[小暮, 91] 小暮潔: 型付き素性構造の実装手法, NLC91-3, 1992.

[小暮, 92a] 小暮潔: 増進的グラフ複製を用いた型付き素性構造汎化手法, NLC92-8, 1992.

[小暮, 92b] 小暮潔: 型付き素性構造の差分, NL 研資 92-5, 1992.

[高橋 ほか, 92] 高橋誠, 松尾秀彦, 鷺恭子, 田代敏久, 永田昌明: ループを含む素性構造単一化における構造共有手法, 情報処理学会第 45 回全国大会 2J-4, 1992.

[竹沢 ほか, 93] 竹沢寿幸, 森元逞, 谷戸文廣, 鈴木雅実, 嵯峨山茂樹, 椚松明: ATR 音声言語翻訳実験システム ASURA, 情報処理学会第 46 回全国大会 6B-5, 1993.

[田代・永田, 92] 田代敏久, 永田昌明: 単一化に基づく構文解析における制約の選択的適用, 情報処理学会第 45 回全国大会 4F-9, 1992.

[中村ほか, 91] 中村佳介, 松本裕治, 長尾真: 選言的素性構造を用いた自然言語の曖昧性の表現とその解消法, 情報処理学会第 43 回全国大会 3G-2, 1991.

[永田 ほか, 90] 永田昌明, 久米雅子, 小暮潔: 単一化に基づく枠組みにおける日本語対話文解析用文法の記述とその計算的側面, NL 研資, 90-NL-76-1, 1990.

[永田・小暮, 90] 永田昌明, 小暮潔: 音声言語日英翻訳実験システム SL-TRANS における日本語解析, NL 研資, 90-NL-78-20, 1990.

[永田, 91] 永田昌明: 単一化文法の効率的な解析手法, 情報処理学会第 42 回全国大会 5C-8, 1991.

[永田 ほか, 91] 永田昌明, 竹沢寿幸, 森元逞: 疎結合かつ階層的な音声言語インタフェース: 音声認識用文法と言語処理用文法の段階的な統合を目指して, 情報処理学会第 43 回全国大会 6V-1, 1991.

[永田, 92] 永田昌明: 単一化に基づく構文解析: 入門編, ATR テクニカルレポート TR-I-0288, 1992.

[永田・久米, 90] 永田昌明, 久米雅子: SL-TRANS における日本語文法の概要, ATR テクニカルレポート TR-I-0156, 1990.

[永田 ほか, 93a] 永田昌明, 田代敏久, 衛藤純司, 坂口明子: 日本語解析文法の開発の軌跡, ATR テクニカルレポート TR-I-0334, 1993.

[永田 ほか, 93b] 永田昌明, 田代敏久, 衛藤純司, 坂口明子: 日本語解析文法 解説書, ATR テクニカルレポート TR-I-0335, 1993.

[松尾 ほか, 91a] 松尾秀彦, 谷田康郎, 永田昌明: 日本語対話文解析文法の開発環境, 情報処理学会第 42 回全国大会 7E-2, 1991.

[松尾 ほか, 91b] 松尾秀彦, 谷田康郎, 永田昌明, 竹沢寿幸: 日本語対話文解析部における計算コストの削減方法 (形態素解析部と構文解析部の分離とインタフェース), 情報処理学会第 43 回全国大会 3G-10, 1991.

[横田, 88] 横田一正: 新しいプログラミング・パラダイム 14, レコードプログラミング, bit Vol.20, No.12, 1988.

[吉本, 88] 吉本啓: 句構造文法にもとづく日本語文の解析, ATR テクニカルレポート TR-I-0049, 1988.

[Aït-Kaci, 1986] Aït-Kaci, H.: "An Algebraic Semantics Approach to the Effective Resolution of Type Equations," *Journal of Theoretical Computer Science 45*, pp.293-351, North-Holland, 1986.

[Gunji, 87] Gunji, T.: *Japanese Phrase Structure Grammar — A Unification-Based Approach*, Dordrecht, Reidel, 1987.

[Kasper, 87] Kasper, R.: "A Unification Method for Disjunctive Feature Descriptions," *Proc. of the 25th ACL*, 1987.

[Kay, 1980] Kay, M.: *Algorithm Schemata and Data Structures in Syntactic Processing*, Technical Report CSL-80-12, Xerox PARC, 1980.

[Kogure, 89] Kogure, K.: "Parsing Japanese Spoken Sentences based on HPSG," *Proc. of IWPT-89*, 1989.

[Kogure et. al, 90] K. Kogure, H. Iida, T. Hasegawa, and K. Ogura: "NADINE: An Experimental Dialogue Translation System from Japanese to English", Proc. Info Japan'90, 1990.

[Kogure, 90] Kogure, K.: "Strategic Lazy Incremental Copy Graph Unification", *Proc. of COLING-90*, 1990.

[Kogure, 1992] Kogure, K.: *A Treatment of Negative Descriptions of Typed Feature Structures*, COLING-92, 1992.

[Maxwell and Kaplan, 91] Maxwell, J. and Kaplan, R.: "A Method for Disjunctive Constraint Satisfaction", *Current Issues in Parsing Technology*, Kluer, 1991.

[Maxwell and Kaplan, 92] Maxwell, J. and Kaplan, R.: "The Interface between Phrasal and Functional Constraints", Unpublished manuscript, 1992.

[Mitchell et. al, 92] Mitchell, D., Selman, B., and Levesque, H.: "Hard and Easy Distributions of SAT Problems", *Proc. of AAAI-92*, 1992.

[Nagata, 92] Nagata, M.: "An Empirical Study on Rule Granularity and Unification Interleaving Toward an Efficient Unification-Based Parsing System," *Proc. of COLING-92*, 1992.

[Nagata, 93] Nagata, M. and Morimoto, T.: "A Unification-Based Japanese Parser for Speech-to-Speech Translation", IEICE Trans., Vol E76-D, No.1, 1993.

[Norvig, 92] Norvig, P.: *Paradigms of Artificial Intelligence Programming: Case Studies in Common Lisp*, Morgan Kaufmann, 1991.

[Pollard, 87] Pollard, C. and Sag, I.: *Information-based Syntax and Semantics —Volume 1 Fundamentals*, CSLI Lecture Notes, No.13, 1987.

[Tomabechi, 91] Shimazu, A.: "Scheduling of Parsing Processes – From the viewpoint of a Mental OS –," *Proc. of JAJSNLP-91*, 1991.

[Tomabechi, 91] Tomabechi, H.: "Quasi-Destructive Graph Unification," *Proc. of 29th ACL*, 1991.

[Tomabechi, 92] Tomabechi, H.: "Quasi-Destructive Graph Unification with Structure-Sharing," *Proc. of COLING-92*, 1992.

[Wroblewski, 87] Wroblewski, D.: "Nondestructive Graph Unification," *Proc. of the 6th AAAI*, 1987.

# 付録 A

# 選言的素性構造の単一化

選言的な素性構造の単一化アルゴリズムとして、ここでは、漸近法 (連続的近似法 successive approximation) [Kasper, 87]を用いた。以下では、漸近法で用いられているデータ構造の概要を説明する。

無条件連言 (unconditional conjunct) を選言を含まない式の連言と定義する。選言を含む素性記述は、選言を含む経路を展開し、交換律により選言を含まない素性記述を前に出すと、次のような式で表せる。

$$uconj \wedge disj_1 \wedge disj_2 \wedge \ldots \wedge disj_m \tag{A.1}$$

ここで、$uconj$ は無条件連言で、$disj_i$ は二個以上の要素を含む選言である。また、選言中の各要素 (disjunct) も上のような形式に変換できるから、選言 (disjunction) は次のような形式に変換できる。

$$(uconj_1 \wedge disj_{1,1} \wedge disj_{1,2} \wedge \ldots \wedge disj_{1,x}) \vee \ldots \vee (uconj_n \wedge disj_{n,1} \wedge disj_{n,2} \wedge \ldots \wedge disj_{n,x}) \tag{A.2}$$

そこで、一般選言を含む素性構造を表すために、定部と不定部からなる次のようなデータ構造 (feature-description 構造または desc 構造と呼ぶ。) を定義する。

定部 (definite part): 選言を含まない素性構造。従って DAG で表せる。

不定部 (indefinite part): 選言の集合。各選言は feature-description 構造の集合である。

例えば、$\phi_i$ を dag 構造とすれば、desc 構造は、以下のような、dag を連言 (conjunction) と選言 (disjunction) を用いて組み合わせた論理式に相当する。

$$\phi_0 \wedge (\phi_1 \vee \phi_2) \wedge (\phi_3 \vee \phi_4 \vee (\phi_5 \wedge (\phi_6 \vee \phi_7))) \tag{A.3}$$

ここでは、$\phi_0$ が定部で、残りが不定部である。不定部は二つの選言 (disjunction) から構成され、一つは $\phi_1$ と $\phi_2$ からなる選言、もう一つは $\phi_3$ と $\phi_4$ と $\phi_5 \wedge (\phi_6 \vee \phi_7)$ からなる選言である。選言の各要素 (disjunct) は feature-description 構造であり、DAG は定部のみの feature-description 構造と解釈できる。特に、最後の要素 (disjunct) は、定部 $\phi_5$ と不定部 $\phi_6 \vee \phi_7$ から構成されている。

また、これは図 A.1のような and-or 木で表現できる。図 A.1において、黒丸は、and 節点 (conjunction) であり、白丸は or 節点 (disjunction) である。and 節点は desc 構造であり、or 節点は、desc 構造の不定部に対応している。この and-or 木の葉は、desc 構造の定部 (つまり

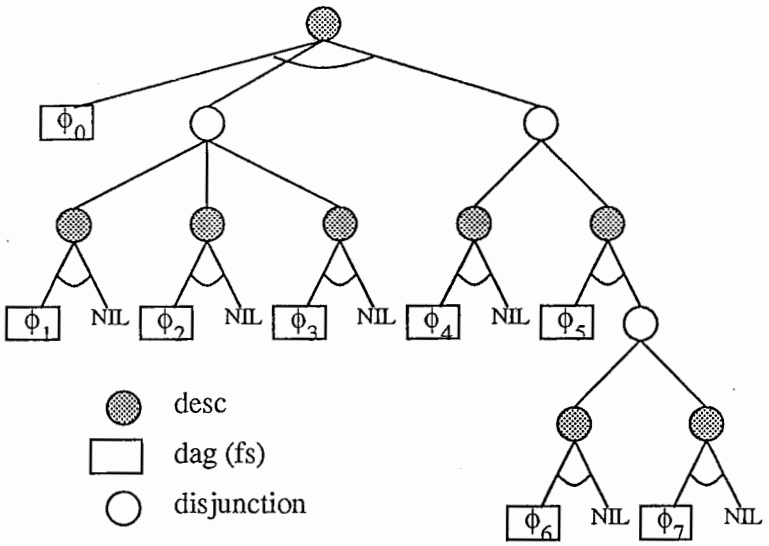


図 A.1: desc 構造の and-or 木による表現

dag) か、nil である不定部となる。すなわち、この and-or 木において、desc 構造の数は、conjunction の数に等しく、さらに、これは dag の数 (nil でない葉の数) と等しい。

$$N_{desc} = N_{conjunction} = N_{dag} = N_{leaf} \tag{A.4}$$

また、nil である葉節点は除外することにすると、この and-or 木の節点の数に関して次の関係がある。

$$N_{tree-node} = N_{conjunction} + N_{disjunction} + N_{dag} \tag{A.5}$$

$$= N_{disjunction} + 2 \times N_{dag} \tag{A.6}$$

選言的素性構造の単一化は、充足可能性 (satisfiability) 問題[茨木, 92]と捉えることができる。制約充足問題 (SAT) では、変数の数、節の数、節の中のリテラルの数などが計算の複雑さの尺度になる[Mitchell et. al, 92]。これと同様に考えれば、選言的素性構造の単一化においては、dag の数や disjunction の数が問題の複雑さのある程度の目安になると、報告者 (永田) は考えている。

# 付録 B

# 文法コンパイラ

## B.1 文法コンパイラの概要

ここで通称「文法コンパイラ」と呼んでいるモジュールは、文法記述言語で記述された文法(句構造規則と語彙項目)の字句解析・構文解析を行ない、パーザおよびユニファイアの動作に必要な内部データを作成する。

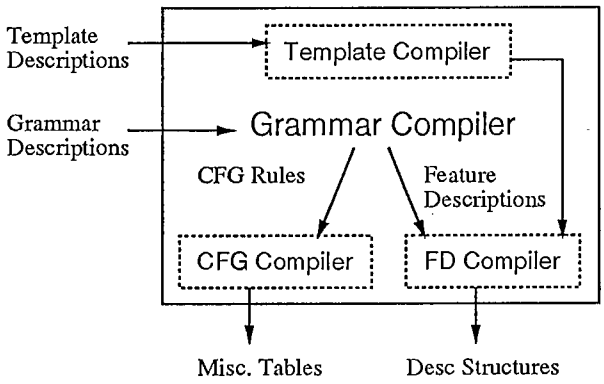


図 B.1: 文法コンパイラの構成

図 B.1に、文法コンパイラの構成を示す。文法コンパイラは、次のようなモジュールから構成されている。ただし、タイプシステムコンパイラは、便宜上、文法コンパイラの中に含めているが、本当は、ユニファイアの一部と考えた方が良いかも知れない。

- CFG 規則のコンパイラ
- 素性記述言語のコンパイラ
- テンプレート記述言語のコンパイラ
- タイプシステム記述言語のコンパイラ

一般的に使われる「コンパイラ」という用語が、字句解析、構文解析、中間語生成、最適化、目的コード生成などのフェーズから成り立つものであるとすれば、上述の 4 つのコンパイラの中で、最適化を行なっているのは CFG 規則のコンパイラだけである。他の 3 つは、記述言語を目的コードに変換しているだけである。

以下では、CFG 規則と素性記述言語のコンパイラについて簡単な説明する。

## B.2 CFG 規則のコンパイル

構文解析において、句構造規則から前もって計算できる情報を最大限に利用するために、CFG で記述された句構造規則から、いくつかの予測表を計算する。

- 下降型 (top-down) の予測表の作成
- 上昇型 (bottom-up) の予測表の作成
- 第一記号表の作成

## B.3 素性構造記述言語のコンパイル

素性構造記述言語のコンパイラが行なっている処理は、次のようなものである。

- 素性構造記述言語の字句解析・構文解析
- テンプレート (マクロ) の展開
- 素性記述 (経路方程式と論理演算子からなる式) への変換
- desc 構造 (dag を葉とする特別な and-or 木) の生成

素性記述言語から選言的な素性構造を生成する過程は次のようなものである。まず、素性記述言語の syntactic sugar を展開して S 式による内部表現を得た後に、素性記述言語の中のテンプレートを展開する。その後、タグで示された構造共有の扱い、否定や選言の扱いなど、幾つかの処理を経て、最終的に dfs-spec 構造と呼ばれる形式に変換される。

- dfs-spec 構造: 定部と不定部を持ち、t-ident-spec とその否定から構成される。
- t-ident-spec 構造: 経路方程式の内部表現。左辺と右辺が token-identity であることを表現する。キーワード :t-ident で始まる (:t-ident 左辺 右辺) という形の S 式である。左辺および右辺は、経路 (path-spec)、または、素性構造 (fs-spec) である。
- path-spec 構造: 経路の内部表現。キーワード :path で始まる (:path 素性名の列) という形の S 式である。
- fs-spec 構造: 素性構造の内部表現。キーワード :fs で始まる (:fs タイプ名 値) という形の S 式である。

この dfs-spec 構造を元にして、素性構造を作成し、これを desc 構造に格納する。

# 付録 C

# 文法に関する数値

ここでは、4章で使用された文法に関する様々な数値を示す。対象となるのは、次の4つの文法である。それぞれの文法の中に記述されている内容については、次の文献を参照して欲しい。

coarse モデル会話 AB1-5 の文法 (粗粒度)[吉本, 88][永田・久米, 90]

medium モデル会話 AB1-5 の文法 (中粒度)[永田 ほか, 93a]

m-set モデル会話 AB1-10 の文法 (中粒度)[永田 ほか, 93a]

e-set 機能試験文の文法 (中粒度)[永田 ほか, 93b]

## C.1 文法が使用する記憶資源に関する尺度

「文法が使用する記憶資源」と「文法に記述されている情報量」の尺度として、次のようなものを考える。

**素性構造の大きさの尺度** 素性構造の大きさの尺度として、これをグラフ表現した際の、ノード数 (dag-nodes) とアーク数 (dag-arcs) と用いる。大まかに言って、グラフのノード数が素性構造の「大きさ」に対応し、ノード数とアーク数の比が素性構造の「複雑さ」に対応する。

**選言的素性記述の大きさの尺度** 選言的素性記述の大きさの尺度として、AND-OR 木において素性構造が格納されているノードの数 (dags) と、OR ノードの数 (disjunctions) を用いる。詳しくは、Aを参照して欲しい。

**素性記述の中の情報量の尺度** 素性記述の中の情報量の尺度として、文法中の異なり素性名の数 (feature-names)、文法中で定義された素性値の延べ数 (atomic-values)、文法中で定義された変数の延べ数 (variables) を用いる。変数は相互参照 (co-reference) を表現するのに用いられるので、素性値と変数の和は、文法が規定している情報量の尺度になる。従って、この情報量の尺度をグラフの大きさで割ったものは、記憶資源の利用効率の尺度になる。

表 C.1に、4種類の文法に対する上述の尺度の値を示す。これらの値は、文法が使用している記憶資源の総和を表す。この表より次のことが分かる。

- e-set は m-set の約 2.5 倍の記憶資源を必要とする。
- m-set と e-set の素性値の異なり数はほぼ同じである。
- 素性値以外の項目では、m-set と e-set の数値の割合がほぼ一定である。

表 C.1: 文法 (句構造規則および辞書) が使用する記憶資源

| | dag-nodes | dag-arcs | dags | disjunctions | feature-names | atomic-values | variables |
|---|---|---|---|---|---|---|---|
| coarse | 42903 | 47066 | 2538 | 893 | 144 | 12839 | 6212 |
| medium | 52354 | 60877 | 6577 | 915 | 157 | 11946 | 10501 |
| m-set | 63803 | 73286 | 3617 | 1170 | 215 | 15841 | 12056 |
| e-set | 156481 | 173891 | 9535 | 3276 | 213 | 43071 | 25246 |

## C.2 文法の規模と記述の様態に関する尺度

文法の粒度、より一般的には、文法記述の様態 (manner) に関する尺度として、次のようなものを考える。

**文法の粒度の尺度** 文法の粒度の尺度として、句構造規則の総数 (count)、および、規則当たりの平均ノード数 (node) と平均アーク数 (arc) を考える。前者は句構造規則の詳細度を表し、後者は素性記述の大きさを表すと考えられる。

**選言的素性記述の大きさの尺度** 選言的素性記述の大きさの尺度として、and-or 木において素性構造が格納されているノードの数 (dag) と、or ノードの数 (disj) を、規則当たりで平均したものを用いる。

**素性記述の中の情報量の尺度** 素性記述の中の情報量の尺度として、一つの規則の中に現れる異なり素性名の平均 (f-nam)、一つの規則の中で定義される素性値の延べ数の平均 (f-val)、一つの規則の中で定義される変数の延べ数の平均 (var) を用いる。

### C.2.1 句構造規則に関する比較

表 C.2: 文法に関する統計量 (句構造規則)

| | count | node | arc | dag | disj | f-nam | f-val | var |
|---|---|---|---|---|---|---|---|---|
| coarse | 22 | 91.1 | 113.9 | 6.5 | 1.9 | 26.1 | 8.7 | 25.2 |
| medium | 163 | 63.2 | 88.7 | 2.3 | 0.5 | 22.7 | 3.3 | 24.7 |
| m-set | 194 | 60.9 | 85.9 | 2.1 | 0.4 | 22.5 | 3.3 | 24.2 |
| e-set | 223 | 64.7 | 90.4 | 2.5 | 0.5 | 23.8 | 3.8 | 24.8 |

上述の尺度について、句構造規則の間で比較した結果を表 C.2に示す。この表より、同程度の適用範囲を持つ文法で比較した場合、文法の粒度に関して次のようなことが分かる。

- 中粒度文法 (medium) は粗粒度文法 (coarse) より句構造規則の数が一桁多い。
- これに対して、medium の素性記述の量は coarse の約 70% である。
- medium の選言の数は coarse の約 1/3 である。
- medium において平均的に定義されている素性値の数は、coarse の半分以下である。

- coarse と medium の変数の数は同程度である。これは、変数が主に素性伝播の原則の実現に用いられているためだろう。

また、同じ粒度の文法について比較した場合、文法の規模に関して次のようなことが分かる。

- medium, m-set, e-set を比較すると、文法の規模は、句構造規則の数に反映されている。
- 文法記述の様態は、文法の規模が変わっても、あまり変化がない。

### C.2.2 語彙項目に関する比較

表 C.3: 文法に関する統計量 (語彙項目)

| | count | node | arc | dag | disj | f-nam | f-val | var |
|---|---|---|---|---|---|---|---|---|
| coarse | 438 | 93.4 | 101.8 | 5.5 | 1.9 | 36.0 | 28.9 | 12.9 |
| medium | 504 | 83.5 | 92.1 | 4.9 | 1.7 | 31.3 | 22.7 | 12.8 |
| m-set | 687 | 75.7 | 82.4 | 4.7 | 1.6 | 31.3 | 22.1 | 10.7 |
| e-set | 1776 | 80.0 | 86.6 | 5.1 | 1.8 | 33.4 | 23.8 | 11.1 |

さらに、前述の尺度について、語彙項目の間で比較した結果を表 C.3に示す。この表より次のようなことがわかる。

- 全般的に、語彙項目は句構造規則に比べて素性記述の量が多い。特に、選言の数、素性値の数が多く、変数の数が少ない。
- coarse と medium を比較すると、文法の粒度の差は、ノード数、アーク数などに反映されている。しかし、素性記述の量については、語彙項目には句構造規則ほど顕著な差が現れない。
- medium, m-set, e-set を比較すると、文法の規模は、語彙項目の数に反映されている。

### C.2.3 品詞に関する比較

e-set の句構造規則および語彙項目について、前述の尺度を品詞別に求めた結果を、表 C.4および表 C.5に示す。e-set で実際に使われている品詞は非常に細かく分かれているので、平均を求める際には、品詞の大分類を用いた。また、表 C.4の品詞は、句構造規則の左辺の品詞である。

これらより次のことが分かる

- 素性記述の量は、品詞により大きく異なる。
- 句構造規則・語彙項目共にでは、動詞、助動詞、助詞、名詞などの素性記述が大きく、選言の数も多い。

表 C.4: 品詞ごとの統計量 (e-set の句構造規則)

| lhs | count | d-node | d-arc | dag | disj | f-nam | f-val | var |
|---|---|---|---|---|---|---|---|---|
| ADV | 173 | 47.8 | 53.0 | 3.2 | 1.1 | 23.7 | 8.8 | 7.0 |
| ATT | 33 | 37.8 | 44.2 | 1.8 | 0.2 | 20.4 | 8.4 | 6.8 |
| AUXV | 207 | 83.1 | 91.1 | 4.7 | 1.6 | 32.0 | 22.4 | 10.5 |
| FADV | 1 | 57.0 | 64.0 | 3.0 | 1.0 | 26.0 | 12.0 | 9.0 |
| INFL | 72 | 32.8 | 34.7 | 2.2 | 0.6 | 16.8 | 10.2 | 4.1 |
| N | 865 | 68.9 | 74.4 | 3.8 | 1.2 | 36.0 | 25.6 | 9.3 |
| POSTP | 63 | 70.7 | 73.0 | 6.1 | 1.9 | 21.3 | 15.7 | 7.2 |
| PREFIX | 12 | 35.0 | 41.0 | 2.3 | 0.7 | 20.2 | 4.8 | 6.6 |
| SUFFIX | 57 | 45.5 | 50.5 | 1.7 | 0.3 | 30.7 | 17.6 | 6.5 |
| V | 293 | 156.4 | 168.5 | 11.7 | 4.7 | 41.6 | 37.2 | 23.5 |
| Ave. | 1776 | 80.0 | 86.6 | 5.1 | 1.8 | 33.4 | 23.8 | 11.1 |

表 C.5: 品詞ごとの統計量 (e-set の辞書)

| lhs | count | d-node | d-arc | dag | disj | f-nam | f-val | var |
|---|---|---|---|---|---|---|---|---|
| ADV | 11 | 57.5 | 81.3 | 1.9 | 0.5 | 23.0 | 2.8 | 22.7 |
| ATT | 5 | 40.2 | 61.4 | 1.4 | 0.2 | 26.0 | 1.6 | 18.8 |
| AUXV | 24 | 35.6 | 51.5 | 1.2 | 0.0 | 18.7 | 1.1 | 15.7 |
| N | 97 | 61.7 | 89.1 | 1.8 | 0.3 | 23.7 | 3.2 | 25.2 |
| P | 4 | 49.2 | 76.0 | 1.0 | 0.0 | 27.2 | 1.2 | 23.8 |
| POSTP | 4 | 90.5 | 124.8 | 2.5 | 0.8 | 27.5 | 5.8 | 33.0 |
| SENT | 3 | 9.0 | 14.0 | 1.0 | 0.0 | 8.0 | 0.0 | 6.0 |
| START | 1 | 13.0 | 10.0 | 3.0 | 1.0 | 6.0 | 4.0 | 0.0 |
| V | 74 | 83.3 | 111.0 | 4.1 | 1.1 | 26.1 | 6.0 | 28.6 |
| Ave. | 223 | 64.7 | 90.4 | 2.5 | 0.5 | 23.8 | 3.8 | 24.8 |

# 付録 D

# 日本語解析システムの開発の経緯

日本語解析システムの基本的な部分は、元々、小暮氏により作成された[小暮, 88] [Kogure, 89]。このシステムは、その後、報告者らによって、継続的かつ精力的に様々な改良が加えられ [永田, 91] [松尾 ほか, 91b][Nagata, 92] [高橋 ほか, 92][田代・永田, 92]、ようやく現在の日本語解析システムが完成した。

日本語解析部には、実は、Lisp 版と C 版がある。C 版は、Lisp 版の約半分の時間で解析が終了する。しかし、C 版には文法コンパイラが存在しないので、Lisp 版の文法コンパイラで作成した文法・辞書情報を、インタフェースプログラムを用いて C 版にロードする構成になっている。Lisp 版の解析時間が遅いために、デモンストレーションにはなどに耐えなかった頃は、アルゴリズムの開発は Lisp 版で行ない、実行は C 版という役割分担があった。

以下では、Lisp 版と C 版の日本語解析システムの開発の経緯を述べる。ここでは、報告者らが開発にあたった 89 年 3 月から 93 年 3 月までの間、どのような改良が試みられたかを、時間的な経過に沿って述べる。また、成果が得られた研究項目と同時に、成果が得られなかった研究項目も列挙する。次の世代の研究者の何かの参考になれば幸いである。

## D.1 Lisp 版の日本語解析システム

表 D.1 に、Lisp 版の日本語解析システムの開発の進捗状況を示す。表の左端は開発期間を示し、パーザ・ユニファイア・文法コンパイラの各々について、バージョン名と改良項目を示す。また、表の右端には、開発支援環境を整備するために行なわれた作業を示す。以下、特に断らない限り、日本語解析システムは Lisp による実装を指すものとする。

### D.1.1 89 年 2 月以前

日本語解析システムの最初のバージョン (1.0) は、アクティブ・チャート・パーザ (ACP) と非破壊的グラフ単一化 (NDGU) を用いていた[小暮, 88] [Kogure, 89]。このパーザは、通称「小暮パーザ」と呼ばれ、Symbolics および Explorer の上で動いていた。なお、後に、句構造規則の粒度との兼ね合いが議論される[永田, 91] [Nagata, 92] 各種の素性記述評価モード (early/late mode) も、若干のバグは残っていたが、既に「小暮パーザ」の中で実装されていた。

### D.1.2 89 年 3 月～90 年 3 月

89 年 5 月頃、永田は、この「小暮パーザ」を Sun の上に移植し、89 年 3 月～90 年 3 月にかけて、音声言語日英翻訳実験システム SL-TRANS [小倉 ほか, 89] の日本語解析文法の開発に用いた。漢字を扱うために、Symbolics 版と Sun 版では、終端記号の扱いが変更されている。そこで Symbolics 版を Version 1.0 とし、Sun 版を Version 2.0 と呼ぶことにした。

表 D.1: 日本語解析システムの開発の進捗状況

| 期間 | 版 | パーザ | 版 | ユニファイア | 版 | 文法コンパイラ | 開発支援環境 |
|---|---|---|---|---|---|---|---|
| ～89.2 | 1.0 | ACP | 1.0 | NDGU(SLING) | 1.0 | | |
| ～90.3 | 2.0 | Symbolics から Sun へ | 2.0 | 同左 | 2.0 | 同左 | |
| ～90.9 | 2.1 | 形態素解析との接続<br>辞書二次記憶化 | | | 2.1 | 二次記憶化辞書 | 会話型文法デバッガ |
| ～91.3 | 2.2 | チャート初期化方式変更<br>形態素解析ユーザ辞書<br>構文解析前処理 | | | 2.2 | 辞書実装方式変更 | テンプレートの<br>動的修正機能 |
| ～91.9 | | | 3.0 | QDGU<br>否定の扱いの変更<br>メモリ管理改良の試み | | | |
| ～92.3 | | | 4.0 | QDSS<br>dtrs 切断モード実装<br>desc-to-fs の近似計算<br>unify-list 実装の試み | 2.3 | 簡易登録削除機能 | |
| ～92.9 | 2.3 | 二次記憶辞書形式の変更<br>素性記述の選択的適用 | 4.1 | equal-fs のキャッシュ<br>単一化失敗点の調査<br>循環構造の扱いの改良 | 3.0 | 構成の見直し | |
| ～93.3 | 3.0 | 弧非共有型の ACP<br>確率的アジェンダ制御 | | | | | 実験管理ツール |

なお、小暮氏は、89 年後半に、戦略的遅延単一化アルゴリズム (SLING)[Kogure, 90]を提案した。しかし、彼が作ったプログラムは、否定・循環・タイプなどに関する拡張がなされておらず、結局、音声翻訳システムには使われなかった。

## D.1.3 90 年 4 月～90 年 9 月

### D.1.3.1 形態素解析プログラムと構文解析プログラムの接続

日本語解析部の形態素解析相当部分を専用の形態素解析プログラムで置き換えることにより、構文解析部の負担を軽減しようとした。形態素解析プログラムは、ATR 言語データベースの形態素付与作業用に作成されたものを転用した。しかし、形態素解析部と構文解析部の間で、品詞および語彙区切りが異なることが多いので、このプログラムはテストデータ以外では動作しなかった。

### D.1.3.2 辞書の二次記憶化

これまで主記憶常駐だった日本語解析辞書を二次記憶に格納するようにした[松尾 ほか, 91b]。しかし、二次記憶上の辞書形式として、文法記述言語をそのまま用いので、読み込み、および、内部形式への展開に時間がかかるという問題点があった。

### D.1.3.3 会話型デバッガの開発

チャートパーザに対して、ステップ・トレースなどの様々なデバッグ機能を付加した[松尾 ほか, 91a]。「小暮パーザ」もある程度のデバッグ機能は備えていたが、これはそれより強力である。

## D.1.4 90 年 10 月～91 年 3 月

### D.1.4.1 チャート初期化方式および二次記憶辞書実装方式の変更

これは、形態素解析部と構文解析部の接続方式を改良するものである。前期の作業 (D.1.3.1節) において、仕様の基本的な部分がうまく実装されていなかった (理解されていなかった) ので、再度、実装したというのが実態である。

教訓 1 仕様書は、できる限り細かく書くこと。その方が、結局は時間の節約になる。 □

### D.1.4.2 形態素解析ユーザ辞書の実現

言語データベース用の形態素解析プログラムは、言語データベースをバッチ処理することしか考慮していないので、語彙の登録・修正・削除が簡単にはできなかった。そこで、新しい単語を登録する機能を付加した。

### D.1.4.3 構文解析前処理部 (形態素調整部)

複合名詞や数量名詞の解析を専用の前処理プログラムで行なうことにより、構文解析部の負担を軽減しようとした。この前処理プログラムを、「構文解析前処理部」または「形態素調整部」と呼んでいる[松尾 ほか, 91b] [永田 ほか, 91]。しかし、この試みは、構文解析前処理規則を開発・保守することの難しさから、実質的に放棄された。

教訓 2 よっぽど大きな性能の差がない限り、複数の文法記述枠組を導入してはいけない。それは、文法の開発現場に混乱を生じさせるだけである。 □

構文解析前処理部は、構文解析の負荷を軽減する役割と、品詞・語彙区切りの違いを吸収するという役割があった。音声認識部と構文解析部の間で、後者の役割を実現する「形態素調整部」は、92 年 3 月頃に谷戸さんによって C 言語で再実装された。より正確には、形態素調整部は仮名漢字変換部と統合されて HMM-LR (SSS-LR) のプログラムの中に取り込まれた。しかし、形態素調整規則の開発・保守が困難であるという点は、現在も問題として残っている。

### D.1.4.4 テンプレートの動的修正機能の実現

主記憶常駐型の文法・辞書において、テンプレートの修正が自動的に反映されるようにするものである。この頃から、主記憶常駐型と二次記憶格納型のどちらの文法開発環境を整備するかで議論となった。現在では、二次記憶格納型の方を重視する方向に傾いている。

## D.1.5 91 年 4 月～91 年 9 月

### D.1.5.1 準破壊的グラフ単一化 (QDGU) の実装

遅延複製を実現した単一化アルゴリズム。詳しくは、本編参照。

### D.1.5.2 否定の扱いの変更

素性構造の否定に関する従来の扱いは、(1) 実行効率が悪い、(2) そこで定義されている否定の意味論で記述できるような文法現象が少ない、という問題点があった。そこで、文法記述の中で否定的素性記述がただ一つ使用されている例である差分リストの empty-check を、unify-dlist の中で行なうように変更した。

#### D.1.5.3 メモリ管理の改良の試み

単一化に基づくパーザが memory-bound なプロセスである。そこで、素性構造を作成するためのメモリの管理を、Lisp システムの Garbage Collector に任せるのではなく、ユーザプログラムの中で行なうようした。その他にも、メモリ管理に関して様々な施策を試みた。しかし、結局、あまり効果がなかった。

> 教訓 3 プログラムを高速化したいのならば、小手先のテクニックを駆使するよりも、新しいアルゴリズムを考えた方が良い。 □

### D.1.6 91 年 10 月〜92 年 3 月

#### D.1.6.1 構造共有型の準破壊的グラフ単一化 (QDSS) の実装

遅延複製と構造共有を実現した単一化アルゴリズム。詳しくは、本編参照。

#### D.1.6.2 dtrs 切断モードの実装

「dtrs 切断モード」では、不活性弧の素性構造において選言が解消されると、dtrs 素性から下が抹消される。これは既に「小暮パーザ」に実装されていた。ここでは、このモードよりも早期に記憶領域を解放できるような方法を実験的に探索した。しかし、常に、正しい解を保証し、かつ、元の「dtrs 切断モード」よりも効率がよい方法はなかった。

#### D.1.6.3 desc-to-fs の近似計算

「desc-to-fs」は、選言を含む素性構造を、選言を含まない素性構造に変換する関数である。この関数は、指数的な計算量を必要とする場合があるので、近似計算によりその効率の改良を試みた。その結果、ほとんどの場合、正しい解を出力し、かつ、効率の良い近似解法を実装した。しかし、理論的な解の保証がないので、正しい解が求まらない場合の原因の究明に時間がかかるという問題がある。従って、より根本的な解決が望まれている。

#### D.1.6.4 unify-list 実装の試み

subcat-slash scrambling の負荷を軽減するために、集合を値として扱えるユニファイアの実装を試みた。しかし、pop, append, union などの集合演算を直接実装しても、より低レベルの機能を用いて実装されていることを除けば、内部で行なわれている処理に大差がないので、あまり大きな効果がなかった。さらに、この方法は、文法記述の大幅な変更を要求するので、結局、この路線は放棄した。

> 教訓 4 理論的またはアルゴリズム的な基盤を持たないアプローチは、結局、うまく行かないことが多い。 □

#### D.1.6.5 簡易登録削除機能

従来の文法コンパイラは、句構造規則と語彙項目を一緒にコンパイルして、各種の予測表を作成していた。このために、語彙の登録・削除に際しては、文法を再コンパイルする必要があり、時間がかかった。これに対して、辞書を検索する時期を変更し、句構造規則だけから予測表をコンパイルすることにより、二次記憶版の辞書の登録・削除が容易に行なえるようにした。

## D.1.7 92年4月～92年9月

### D.1.7.1 二次記憶辞書形式の変更

以前の二次記憶辞書の形式は、文法記述言語をそのまま用いていた。これは、テンプレート定義を利用可能にするためである。しかし、そのために、語彙項目の読み込みの際に、文法記述言語を解釈するオーバヘッドが生じていた。そこで、二次記憶辞書の形式を、素性構造を直接表現するような Lisp の S 式に変更し、高速化を達成した。また、辞書の一部を主記憶に常駐する機能や、一度読み込まれた規則をキャッシュする機能も付加した。これらの作業により、二次記憶版は主記憶版とほぼ同程度の性能を持つようになった。

### D.1.7.2 素性記述の選択的適用

句構造規則の注釈の素性記述の一部を選択的に適用するモード (selective mode) を新たに加えた[田代・永田, 92]。詳しくは、本編参照。

### D.1.7.3 equal-fs のキャッシュと単一化失敗点の調査

素性構造の同一性の検査の結果をキャッシュすることにより、効率の改善を図った。その他にも、様々な関数の結果をキャッシュしてみたが、それ程大きな効果はなかった。また、素性構造の単一化において、素性ごとの単一化の失敗回数などを調べた。その結果、統語的な素性、特に、形態素に関する素性の失敗率が高いことが判った。そこで、とりあえず、形態素に関する素性の単一化を優先する方法を試してみたが、あまり効果がなかった。しかし、研究の余地は、まだ十分あると思う。

### D.1.7.4 循環構造の扱いの改良

構造共有型の準破壊的グラフ単一化アルゴリズムは、循環構造に関わるノードを全て複製してしまうという欠点がある。この改良を試み、アルゴリズムは完成した。しかし、実際に用いられている素性構造は、何重にも絡みあった循環構造をしているために、グラフを巡回するオーバヘッドが問題となり、このアルゴリズムは、元のアルゴリズムと性能的に大差がなかった[高橋ほか, 92]。

### D.1.7.5 文法コンパイラの構成の見直し

以前の文法コンパイラの内部構成は、素性構造記述言語の処理、テンプレート記述言語の処理、文法記述言語の処理などが複雑に絡み合っていて、開発・保守が難しかった。そこで、モジュール構成と中間データの形式を整理し、各モジュールが独立したツールとしても使えるようにした。

## D.1.8 92年10月～93年3月

### D.1.8.1 弧非共有型の ACP

確率的な構文解析を行なったり、音声認識のスコアを優先するような解析を行なうなど、文脈自由文法の範囲を越えるような処理を行なう場合には、弧を共有している「小暮パーザ」は、不都合な点が多かった[1]。そこで、弧を共有しないアクティブチャートパーザを作成した。この「弧を共有しないパーザ」と「弧を共有するパーザ」のとちらが今後の主流となるかは、これからの研究次第である。

[1]実は、HPSG などの単一化文法も CFG の範囲を越えている。これを句構造規則に関しては、多項式時間で扱えるようにするために、「小暮パーザ」は、弧の共有と伝搬など、かなり複雑な機構を用意している。

#### D.1.8.2 確率的アジェンダ制御

弧を共有しないパーザにおいて、句構造規則の確率に基づいてアジェンダ制御を行なうモードを作成した。

#### D.1.8.3 実験管理ツール

音声翻訳システム ASURA の開発の効率化のために、音声認識結果・構文解析結果などを管理するデータベースと、このデータベースから、正解率などの統計情報を抽出するプログラム群を作成した。これらをまとめて、「音声翻訳実験管理ツール」と呼んでいる。

## D.2 C 版の日本語解析システム

表 D.2に、C 版のパーザおよびユニファイアの開発の進捗状況を示す。

表 D.2: C 版のパーザおよびユニファイアの開発の進捗状況

| 期間 | パーザ | ユニファイア | 文法情報インタフェース |
|---|---|---|---|
| ～90 年 3 月 | | NDGU | 第 1 版 |
| ～91 年 3 月 | ACP | | |
| ～91 年 9 月 | ラティス入力 | | |
| ～92 年 3 月 | 二次記憶化辞書 | | |
| ～92 年 9 月 | LR パーザとの接続 | QDSS | 第 2 版 (高速化版) |

報告者 (永田) は、C 版のパーザの実装がいつ頃から始まったのかは良く知らない。報告者がこれに関係するようになったのは、90 年の前半からである。90 年 3 月頃には、非破壊的グラフ単一化アルゴリズム (NDGU) に基づく C 版のユニファイアが完成していた。

初期の C 版パーザは、Lisp 版のアクティブチャートパーザ (ACP) から C 版のユニファイアを呼び出すという構成だった。しかし、この構成は、Lisp と C の環境の間で大きなグラフの受け渡しをすることのオーバヘッドのために、あまり速くなかった。そこで、C 版のアクティブチャートパーザの実装を始め、90 年の後半には、C の環境だけで動くパーザが完成した。この C 版のパーザは、Lisp 版より約 2 倍速かった。

> 教訓 5 アルゴリズムが同じであれば、プログラミング言語を変えても、それほど大きく実行効率が変わるわけではない。 □

91 年 4 月頃から、音声認識部の出力を扱えるように C 版のパーザにラティス構造の入力を扱う機能を付加した。さらに、92 年の後半には、辞書を二次記憶に置くように改造された。二次記憶化に際しては、C-ISAM と呼ばれる UNIX 上の I-SAM ファイルユーティリティを使用した。

92 年 5 月頃に、Lisp 版の文法コンパイラと C 版パーザとの新しいインタフェースプログラムが完成した。これにより、C 版パーザ用の文法情報を作成する時間が大幅に短縮された。また、92 年 7 月頃には、構造共有型の準破壊的グラフ単一化アルゴリズム (QDSS) に基づく C 版のユニファイアが完成した。

なお、92 年半ばには、この C 版のユニファイアと、音声認識部 (HMM-LR) で用いられている C 版の LR パーザを組み合わせたプログラムも完成している。

## D.3 音声翻訳実験システムと日本語解析システムの関係

最後に、ATR の音声翻訳実験システムと日本語解析システムのバージョンとの関係について述べておく。日本語解析システムは、次の三つの (音声) 翻訳実験システムの中で使用された。

- 対話翻訳実験システム NADINE (88 年 11 月〜)
- 音声言語日英翻訳実験システム SL-TRANS (89 年 5 月〜)
- ATR 音声翻訳システム ASURA (92 年 11 月〜)

NADINE[Kogure et. al, 90] は、端末間の対話のように、キーボード入力による対話文の翻訳システムである。このシステムには、Version 1.0 の日本語解析システムが使われた。

SL-TRANS[小倉ほか, 89] は、89 年秋に、ATR が初めて公開した音声翻訳実験システムである。このシステムには、Version 2.0 の日本語解析システムが使われた。

91 年 5 月に、天皇陛下が ATR を御訪問になった。それ以前のシステムは、予め録音された音声を用いていたが、この時に初めて、マイクによる音声入力が可能になり、真に「音声翻訳システム」と呼べるものが出来上がった。これ以降のシステムを、SL-TRANS 2 と呼ぶこともある。このシステムには、C 版の日本語解析システムが使われた。

ASURA[竹沢 ほか, 93] は、ATR の自動翻訳電話プロジェクトの研究成果を結集して作られた、世界初の自動翻訳電話システムである。このシステムには、幾つかのバージョンがあるが、日本語解析システムに関しては、Version 2.3 のパーザ、Version 4.0 のユニファイア、Version 3.0 の文法コンパイラが用いられている。なお、93 年 1 月 28 日の自動翻訳電話国際共同実験でも、日本語解析システムは、上述の ASURA と同じものが使われた。