CFG フィルタリングを用いた高効率な LTAG パーザの構築

大内田賢太 † 吉永直樹 † 辻井潤一 ‡*
† 東京大学大学院情報理工学系研究科コンピュータ科学専攻
‡ 東京大学大学院情報学環 * CREST, 科学技術振興機構
{oouchida, yoshinag, tsujii}@is.s.u-tokyo.ac.jp

1 はじめに

本研究では、CFG フィルタリング [1, 2, 3, 4, 5, 6, 7] を用いた LTAG[8] パーザの実現への第一段階として、LTAG から CFG への近似手法を提案する。

LTAG や HPSG[9] のような語彙化文法の枠組みでは、CFG などの単純な文法枠組みに比べ、文に項構造などの豊かな意味表現を与えることができるため、近年、高度な自然言語処理アプリケーションで利用されることが期待されている [10, 11]。しかしながら、語彙化文法の枠組みでは、構文的な制約や意味的な制約などの情報を単語の語彙項目に押し込めるため、木構造や素性構造などの複雑なデータ構造を必要とし、またそのデータ構造を扱う構文解析の速度が問題になる。

この問題に対し、語彙化文法に対する構文解析の高速化手法として、CFG フィルタリング [1, 2, 3, 4, 5, 6, 7] が提案されている。CFG フィルタリングでは、与えられた語彙化文法から近似的な CFG を抽出し、得られた CFG で構文解析の探索空間を絞ることで高効率な構文解析を実現する。しかしながら、HPSG に対する CFG フィルタリング [7] が大規模な文法を用いた際にも高速化が得られている一方で、LTAG に対する既存の CFG フィルタリング [1, 3, 5] は、大規模文法を用いた実験結果が今のところ報告されていない。一方、吉永ら [12] は、LTAG を HPSG スタイルの文法に変換し、得られた文法に対して HPSG の CFG フィルタリングを適用することで、大規模な LTAG 文法に対して高速化を達成している。さらに、LTAG に対する既存の CFG フィルタリングよりも、LTAG を変換した HPSG に対する CFG フィルタリングの方がより効果的であることも示している。

我々の提案する LTAG からの CFG への近似手法では、吉永らにより示された基本的なアイデアに従って、まず LTAG の語彙項目に、LTAG の文法規則である代入と接合をオフラインで適用することにより、構文解析中に生成され得る部分構文木を数え上げる。この際に、CFG を越える記述力を持つ接合の適用回数を制限することで、有限サイズの CFG に近似することを可能とする。次に、それらの部分構文木間の導出関係を CFG の規則とすることで、与えられた LTAG から CFG を獲得する。

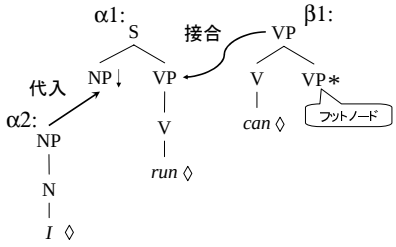

図 1: Elementary tree と、代入、接合操作

本稿で提案した近似手法の有効性を確かめるために、宮尾ら [13] による手法により、Penn Treebank コーパス [14] から自動的に獲得された LTAG を使い、CFG への近似の実験を行った。

2 背景

2.1 Lexicalized Tree-Adjoining Grammar (LTAG)

LTAG では、語彙項目が **elementary tree** と呼ばれる木構造で表現される。文を解析する時は、**代入**と**接合**と呼ばれる文法規則により elementary tree を組み合わせることで、構文木を導出する。図 1 は、“I”, “run”, “can” に対する elementary tree α_1, α_2, β_1 とそれらに対する代入および接合操作を示している。

代入は、木構造の葉ノードを、その葉ノードと同じラベルを根ノードに持つ他の木構造で置き換える操作である。図 1 では、α_1 の ラベル NP の葉ノード を、同じラベル NP の根ノードを持つ α_2 により置き換えている。

接合は、木構造の中間ノードを、その中間ノードと同じラベルを根ノードと葉ノードに持つ他の木構造で置き換える操作である。図 1 では α_1 のラベル VP の中間ノードを、根ノードと一つの葉ノードが同じラベル VP を持つ β_1 で置き換えている。このような、根ノードと同じラベルを持ち、接合のときに使われる特別な葉ノードのことを**フットノード**と呼ぶ。

本研究では、代入と接合を CFG の文法規則に変換することで、LTAG から CFG を抽出する。CFG

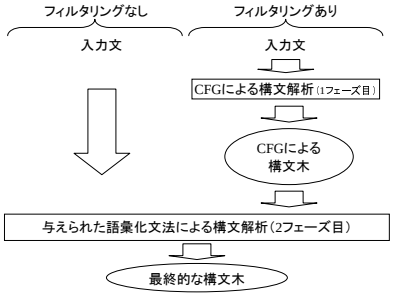

図 2: CFG フィルタリング

に変換するとき、代入は CFG の文法規則と等価な記述力なので問題はないが、接合は CFG の文法規則以上の記述力を持つため、接合をいかに扱うかが問題になる。

2.2 CFG フィルタリング

CFG フィルタリング [1, 2, 4] とは、与えられた文法から CFG を抽出し、その CFG を使って、元の文法で得られるはずの構文木を効率的に絞り込む手法である。CFG フィルタリングでは、まず、与えられた文法からオフラインで CFG を抽出する (以下 **CFG 近似**と呼ぶ)。抽出された CFG は、構文木を過剰生成してもかまわないが、元の文法によって作られる構文木を全て含まなければならない。得られた CFG により、元の文法で得られるはずの構文木が満たすべき必要条件を効率的に計算することができる。

CFG フィルタリングを用いた構文解析は、2 つのフェーズで行われる (図 2)。1 フェーズ目では、オフラインで抽出された CFG を用いて構文解析を行う。このとき得られた構文木は、元の文法の制約の元では生成されない構文木を含みうる。2 フェーズ目では、元の文法の制約を全て使って CFG 文法で得られた構文木を調べ、過剰に生成された構文木を除く。

CFG フィルタリングの性能は、CFG の近似の良さに依存する [12]。CFG が与えられた文法に対する良い近似となっていれば、過剰生成する構文木の数は抑えられ、フェーズ 2 でチェックする構文木の数も抑えられる。従って、いかに元の文法によって与えられた文法的制約を近似の CFG に保存するかが問題となる。

LTAG の既存の CFG フィルタリング [1, 3, 5] では、LTAG の elementary tree を各段ごとに切り離し、それをそのまま CFG 規則として用いている（図 3）。このようにして得られた CFG は、elementary tree に記述された多段の構文的制約が捉えられていないため、フェーズ 2 でそれらの構文的制約をチェックをしなければならない。

一方、吉永らは、LTAG を一旦等価な HPSG に変換し、得られた HPSG 文法に HPSG の CFG 近似手法 [6, 7] を適用することで LTAG から CFG を獲得している。HPSG を CFG に近似する手法の概要は、次の通りである。まず、素性構造で記述された語彙項目に対して文法規則を順次適用し、文法が生成しうる全ての素性構造 (HPSG の場合 sign) を列挙する。次に、この数え上げられた sign に非終端記号を割り当て、sign 間の文法規則で結ばれた導出関係を CFG の規則とする (図 4)。

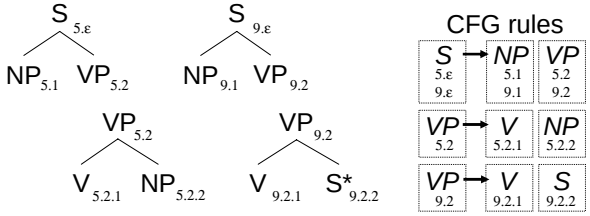

図 3: LTAG から CFG を抽出する既存手法

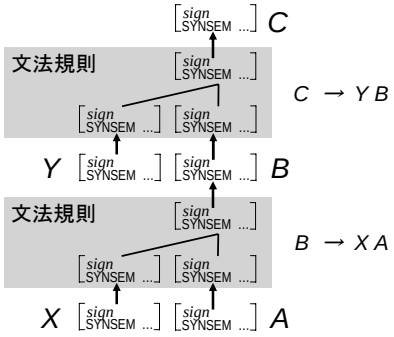

図 4: HPSG から CFG を抽出する既存手法

吉永ら [12] はさらに、LTAG から変換された HPSG にどのように CFG 近似の手法が適用されるかを考察することで、LTAG に対する新しい CFG 近似の方法について言及している。

3 LTAG の CFG への近似アルゴリズム

吉永ら [12] の LTAG の CFG 近似の基本的なアイデアは、2.2 で説明した HPSG の CFG 近似と同様に、「構文解析時に可能な部分構造を数え上げる」というものである。彼らは、LTAG の語彙項目の elementary tree に逐次的に代入・接合を適用することで、可能な部分構文木を数え上げる。文法規則を elementary tree に逐次的に適用する既存の方式としては、例えば head corner traversal [13] が提案されている。head corner traversal では、elementary tree に対し、主辞から根ノードにいたる一意なボトムアップのパスをまず定義する。次に、パス上のノードを順に辿って、適用可能な文法規則を適用していく (図 5)。このとき、生成される部分構文木について、着目しているノードより「下の」部分木は既に文法適用を行った部分、「上の」部分木は、これから文法適用を行う部分であることに注意され

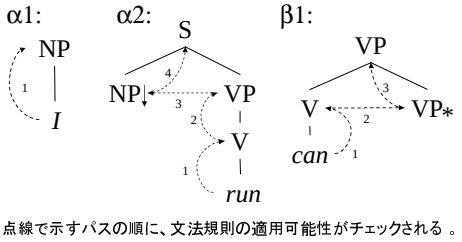

図 5: head-corner traversal

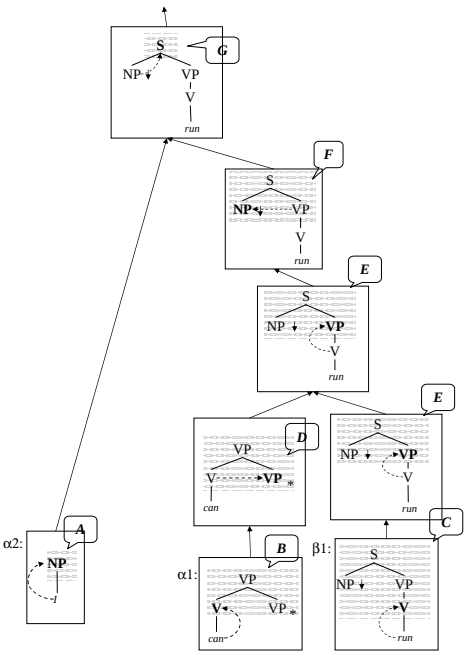

図 6: LTAG から CFG の抽出手法

たい。構文解析に必要となる情報は、着目しているノードより「上の」部分木に全て含まれる。本論文ではこの部分木を**活性部分木** と呼ぶ。吉永らのアイデアでは、このようにして得られる活性部分木を CFG の非終端記号と考え、活性部分木間の導出関係を CFG 規則とすることで CFG を獲得する。

図 6 は図 1 の文法に対し、head-corner traversal に従い、文法適用を行った例である。このとき、網掛けの部分が活性部分木である。この活性部分木に対し、図 6 中の A〜G の様に非終端記号を割り当てる。

次に、CFG の文法規則を抽出する。図 6 を例に、代入・接合を CFG の文法規則として抽出すると、以下のように表せる。

$G \rightarrow A\ F$
$F \rightarrow E$
$E \rightarrow D\ E$
$D \rightarrow B$
$E \rightarrow C$

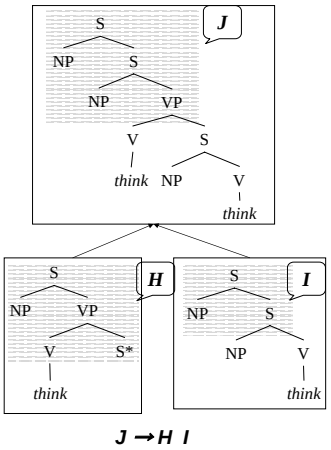

図 7: CFG 抽出:接合

以下で、これらの CFG 規則が LTAG のどの文法規則に対応しているかをみていき、実際に LTAG を有限サイズの CFG に近似する際に問題となるケースについて考察する。

まず、図 6 で上の $G \rightarrow A\ F$ は、LTAG の代入に対応する CFG 規則である。代入が行われる時、親の活性部分木が子供の活性部分木よりも小さくなるこのことから新しく生成される活性部分木の数が有限であることが分かる。活性部分木の数が有限ならば、割り当てる非終端記号も有限になる。つまり、代入のみを文法規則を用いる LTAG であれば、有限の CFG に変換することができる。

また、$F \rightarrow E$、$D \rightarrow B$、$E \rightarrow C$ のように、代入や接合を行わない場合も存在する。代入の場合と同様、親の活性部分木が小さくなるので、有限の CFG に変換することができる。

図 6 で中ほどの $E \rightarrow D\ E$ は、LTAG の接合に対応する CFG 規則である。代入の場合と同様に、活性部分木に非終端記号を割り当てる。このように、フットノードが木構造の深さ 1 段目にある活性部分木の接合の時は、生成される活性部分木のサイズは増大しない。しかし、図 7 の例のように接合する活性部分木のフットノードの深さが木構造の 2 段目以下の位置にあるとき、親の活性部分木が複数の活性部分木の組み合わせになる。これにより、活性部分木のサイズが増えるため、部分活性木が無限に生成されうる。

本手法では、活性部分木ごとに、何度このような活性部分木のサイズが増える接合を適用して得られたのかを記録しておき、閾値以上適用して得られた活性部分木には、全て同じ非終端記号 “*” を割り当てることとする。これにより、活性部分木の最大の深さを制限することができるので、得られる CFG のサイズも有限に抑えることができる。また、得られた CFG が元の文法で得られるはずの構文木を全

表 1: WSJ section 23 から抽出された LTAG

単語数	1,569
非終端記号の数	60
elementary tree の数	466

表 2: LTAG から抽出された CFG

終端記号数	3,001
非終端記号数	1,318
CFG 規則数	19,155

て生成することを保証するために、非終端記号 “*” に対して、全ての非終端記号 X において、$* \rightarrow X\ *$ あるいは $* \rightarrow *\ X$ となる CFG の規則を作る。得られる CFG は、この規則によって、過剰生成する可能性があるが、与えられた文法が出力する全ての構文木を出力することが保証される。

このように、適用する文法規則の回数を制限することで語彙化文法を CFG に変換する手法は Kiefer ら [6] により HPSG の枠組で提案されている。我々の LTAG に対する近似手法は、LTAG の文法規則の性質に着目し、実際に問題となるケースに限って適用回数を制限しており、単純に文法規則自体の適用回数を制限するよりも良い近似を得ることができると思われる。

4 実験

本論文で提案した CFG の近似手法の有効性を確かめるために、コーパスから獲得した LTAG に CFG の近似手法適用し、得られる CFG の性質を調べる。

我々は、本稿で提案した CFG への近似手法を実装し、宮尾ら [14] の文法抽出プログラムを使用し、Penn Treebank Wall Street Journal (WSJ) の section 23 から抽出した LTAG(表 1) に適用した。今回のコーパスから自動獲得した LTAG 文法では、活性部分木のサイズを増やす接合回数を制限すること無しに、LTAG を CFG に近似することに成功した。これは、吉永らの HPSG 経由の CFG コンパイルの [12] の実験結果と一致する。

表 2 に近似の結果得られた CFG のスペックを示す。近似にかかった時間は、約 30 分である。単純に elementary tree の各段を切り離すことで CFG を得る既存の手法の場合、得られる CFG の非終端記号の数は元の LTAG の非終端記号と同じ 60 個となることを考えると、我々の手法で得られる CFG はより深い制約を考慮していると言えるだろう。

5 まとめ

本論文では、CFG フィルタリングを用いた LTAG パーザの実現するため、LTAG から CFG への近似手法を提案した。吉永ら [12] の手法に基づき、構文解析時に可能な部分構造を数え上げるという近似手法を、活性部分木のサイズが増える接合の回数を制限するという制約を取り入れることで実現した。

今後の課題として、まず、制限が必要であると思われる XTAG 英文法などの人手で書かれた文法の CFG への近似実験があげられる。また、LTAG パーザに CFG フィルタリングを実装し、本手法によって得られた CFG による CFG フィルタリングの有効性を検証したい。

参考文献

[1] K. Harbusch. An efficient parsing algorithm for Tree Adjoining Grammars. In *Proc. of the 28th ACL*, pages 284–291, 1990.

[2] J. T. Maxwell III and R. M. Kaplan. The interface between phrasal and functional constraints. *Computational Linguistics*, 19(4):571–590, 1993.

[3] P. Poller. Incremental parsing with LD/TLP-TAGs. *Computational Intelligence*, 10(4):549–562, November 1994.

[4] K. Torisawa and J. Tsujii. Computing phrasal-signs in HPSG prior to parsing. In *Proc. of the 16th COLING*, pages 949–955, 1996.

[5] P. Poller and T. Becker. Two-step TAG parsing revisited. In *Proc. of TAG+4*, pages 143–146, 1998.

[6] B. Kiefer and H.-U. Krieger. A context-free approximation of Head-Driven Phrase Structure Grammar. In *Proc. of the sixth IWPT*, pages 135–146, 2000.

[7] K. Torisawa, K. Nishida, Y. Miyao, and J. Tsujii. An HPSG parser with CFG filtering. *Natural Language Engineering*, 6(1):63–80, 2000.

[8] Y. Schabes, A. Abeillé, and A. K. Joshi. Parsing strategies with ‘lexicalized’ grammars: application to Tree Adjoining Grammars. In *Proc. of the 12th COLING*, pages 578–583, 1988.

[9] C. Pollard and I. A. Sag. *Head-Driven Phrase Structure Grammar*. CSLI Publications, 1994.

[10] Deep Thought Project. http://www.project-deepthought.net/.

[11] Kototoi Project. http://www-tsujii.is.s.u-tokyo.ac.jp/kototoi/.

[12] N. Yoshinaga, T. Kentaro, and J. Tsujii. Comparison between CFG filtering techniques for LTAG and HPSG. In *Proc. of the 41st ACL companion volume*, pages 185–188, 2003.

[13] G. van Noord. Head corner parsing for TAG. *Computational Intelligence*, 10(4):525–534, 1994.

[14] Y. Miyao, T. Ninomiya, and J. Tsujii. Lexicalized grammar acquisition. In *Proc. of the 10th EACL companion volume*, pages 127–130, 2003.