ループを含む素性構造単一化における構造共有手法

2J−5

高橋誠*　松尾秀彦*　鷺恭子*
田代敏久**　永田昌明**
*(株) 東洋情報システム大阪本社 自然言語処理グループ
**(株) ATR 自動翻訳電話研究所

1. はじめに

ATR では、素性構造の複製を逐次的に行なう「非破壊的グラフ単一化アルゴリズム」[1] を元にして、これを遅延複製と構造共有が実現できるように改良した「構造共有型準破壊的グラフ単一化アルゴリズム (Quasi-Destructive Graph Unification with Data Structure Sharing、以下 QDSS)」[2] を開発し、音声言語翻訳実験システム (SL-TRANS) の日本語構文解析部で用いている。

QDSS は、単一化操作を無矛盾性検査と出力素性構造作成の二つの段階に分けることによって過剰複製、早期複製を排除し、さらに、構造共有により冗長複製を回避するアルゴリズムである。

QDSS の問題点は、(1) 素性構造がループを含む場合、ループ上の全てのノードが、たとえ変更されなくても単一化の度に複製されること、(2) Bottom 型 (変数) のノードを構造共有した場合、ある条件下では単一化結果が論理的に不適切なものになること、である。

そこで本稿では、(1)の問題に焦点を当てた QDSS の拡張として、ループを含む素性構造の構造共有を行なう方法について紹介する。

2. 研究の背景

2.1 遅延複製と構造共有の必要性とその効果

単一化文法に基づく構文解析において、素性構造単一化は、時間的にも空間的にも非常にコストの高い処理であり、中でも特に素性構造の複製に処理時間、メモリ空間の多くが費やされることから、単一化時の無駄な複製を削減する様々な手法が考案されている[1][3][4][5][6]。我々は、HPSG/JPSGに基づく日本語文法の構文解析系 [7] を高速化するために、その単一化処理系に遅延複製 [8] や構造共有 [2] の手法を実装した。

表1 に、遅延複製のみを行ない構造共有を行なわない場合 [8] と、遅延複製と構造共有を両方とも行なう場合 [2] について、解析過程で複製される node 数を比較したものを示す。また参考として、一文の解析過程で行なわれる単一化回数とその成功率を示す。処理時間は、構造共有を全く行なわなかった場合を 100 とすると、Atomic 型構造共有の場合 88.8 、Atomic/Complex 型構造共有の場合 84、[1]全て構造共有の場合 62.3 であった。

因みに、国際会議の予約に関する 12 個のモデル会話 (261 文) を対象とした日本語文法 (句構造規則数: 189, 語彙数: 682) において素性構造を表現するのに用いられている node の数は、全体で7,799 個, その内訳は Complex node 53%, Atomic node 29%, Bottom node 18% である。

これらのデータから、構造共有によってメモリ使用量が大幅に減り、処理時間も向上していることがわかる。また、Bottom 型 node は、全体の約 5 分の 1 と比較的多く、しかもそれを構造共有することによる効果が非常に大きいことから、Atomic, Complex 型だけでなく、Bottom 型も安全に構造共有できることが重要であると言える。

2.2 文法記述上の特徴

句構造規則の注釈部に (1) SUBCAT 素性の原理と (2) [2]COH 素性の原理とを同時に使用すると、ループが構成され効率に影響する。SL-TRANS の文法記述では、COH 素性の原理を全ての生成規則の注釈部に使用しているため、素性構造内にループが多発し、かつ、複雑に絡み合う原因となっている。以下に、これらの原理を経路方程式で表したものを示し、これらがループを構成する様子を図 1. に示す。

1) SUBCAT 素性の原理

```
<!m syn subcat> == <!h-dtr syn subcat rest>
<!c-dtr> == <!h-dtr syn subcat first>
```

2) COH 素性の原理

```
<!c-dtrs syn head coh> == <!h-dtr>
```

A Structure sharing Method for Unification of Cyclic Feature Structures
Makoto B. TAKAHASHI, Hidehiko MATSUO, Kyoko SAGI, Toshihisa TASHIRO, Masaaki NAGATA
Toyo Information Systems Co.,LTD. NLP Working Group
ATR Interpreting Telephony Research Laboratories

表 1. 構造共有の効果

	コピーされたノードの数(個)				Unify 成功率	Unify 回数
	構造共有	構造共有の対象ノード型				
文 ID	なし	Atomic	Bottom以外	すべて	(%)	(回)
1	22,431	15,394	12,673	3,074	88.61	196
2	7,283	4,940	4,558	1,238	76.06	71
3	29,211	18,719	16,563	3,848	85.49	195
4	154,240	104,769	93,934	24,187	91.90	661
5	86,028	62,561	55,395	13,803	85.22	410
6	270,616	186,055	159,516	40,828	85.95	1398
7	190,903	129,561	115,692	29,647	83.21	1080
8	595,279	409,096	360,541	91,451	94.12	1909
9	1,488,208	1,033,637	887,176	233,003	93.01	4781
10	251,859	166,004	142,578	34,768	89.96	1262
total	3096058	2130736	1848626	475847		
比率	100%	68.8%	59.7%	15.4%		

●比率は, 構造共有をしない場合に対する比率を示す。

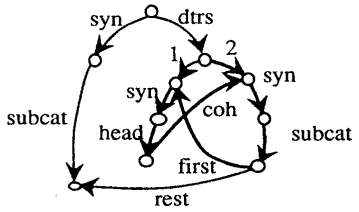

図 1. ループを生じる文法記述

3 QDSS の概要

3.1 アルゴリズム

QDSS での単一化操作は、無矛盾性検査、変更検査、出力素性構造作成の三つの基本処理に分けることができる。[2][8] では、これらを 1) 無矛盾性検査部、および、2) 出力素性構造作成部の二つのモジュールで分業させている。全体の処理は概ね以下に示す通りである。

0) 単一化のメイン (unify-dg)

まず、単一化の世代管理のためのグローバルカウンタをインクリメントする。次に、無矛盾性検査部において、入力素性構造 (dg1, dg2 とする) の単一化成功の可能性を検査する。最後に、成功する入力に対して出力素性構造作成部で出力構造を作る。

1) 無矛盾性検査部 (unify-consistency-check)

dg1、dg2 の矛盾 を探索しながら、dg1 と dg2 の相補的かつ非共通なアーク を dg1 に書き込み、処理過程で等値と判断された node 間に Forward Link を張る。矛盾が発見された場合、単一化は失敗であるとしてすべての単一化処理を終了する。さもなければ、単一化は成功であるとして 2) の処理を行なう。

2) 出力素性構造作成部 (duplicate-dg-share)

dg1 に書き込まれた情報をもとに、変更検査および出力素性構造作成を行なう。dg1 を再帰的に探索しながら再帰の行きがけに変更検査を行ない、戻りがけに変更検査の結果を返しつつその値により出力構

1. Bottom 型の構造共有に問題があることを上で述べた。この問題は、1 文の解析中に同じ生成規則が適用される場合に生じる。ここでは、この場合に限って 2 度目に適用する生成規則の注釈 (素性構造) を複製するという暫定的な手段をとっている。この問題が解消されれば、処理時間、構造共有率ともにさらに向上するはずである。紙面の都合上詳述は控える。
2. [9] ではADJUNCT 素性と呼んでいる。

造を決定する。すなわち、変更があった場合 "変更あり" が返され入力 node の複製を出力構造とし、変更がなかった場合 "変更なし" が返され入力 node を出力構造とする。もし変更検査の対象となる node が Forwarding されていた場合、この node に到達する arc を複製する。

3. 2 効率上の問題

上の説明からも分かるように、QDSS では三つの基本処理を、素性構造に対する二回の走査の間に行なっている (2 パスアルゴリズム)。このため変更検査が未確定なループは変更があったものとしてすべてコピーせざるを得ない。

それぞれの基本処理につき一回の走査を行なう 3 パスアルゴリズムによるのが、遅延複製と構造共有を同時に行なう最も自然な方法であると考えられる。しかし、効率の面からすると走査回数が増えることは好ましいことではない。

本質的に構造共有の効率は、node 複製の空間的コストと 1) 素性構造の走査回数, 2) 変更情報管理のための時間的コスト (ポインタを辿る回数), 3) 変更情報管理のための空間的コスト(node のスロット数, 保持すべき大域的情報), 4) 冗長複製の許容度 (どの程度冗長な複製を許すか), などの諸要因との trade-off となるため、実装方法や環境にかなり依存する。

4. ループのデータ構造共有

ここでは、本研究の焦点となるループのデータ構造共有の手法を説明する。この手法は、QDSS の 無矛盾検査部をそのまま使用し、出力素性構造作成部に変更を加えることで実現される。

4. 1 変更検査の保留

ループ上の node の場合、変更の有無は、その上位構造にも依存するため、戻りがけに必ずしも決定できるとは限らない。そこでループ上の node は、変更が確認されないかぎり変更検査を保留し"変更保留"という値を返し、node には変更検査保留印を残しておく。この時の出力構造には入力構造を暫定的に使用する。これは、後に変更が確認された時点で再構成される。

4.2 変更発見時の再構成

Complex 型 node の変更の有無は、node がもつアークをそれぞれ再帰的に変更検査し、再帰が返す値を総合することによって判定される。(但し、node が差分アークを持つ場合、あるいは Forwarding のターゲットとなっている場合は最初から"変更あり"と判断できる。)

一般的に変更検査過程の状態は、判断過程の変更検査状態と新たに返えされた下位構造の変更検査結果によって、図 2. のような状態遷移ダイアグラムで表すことができる。

もし図 2. に A. B. で示した状態遷移があれば、再構成フラグが立つ。下位構造の検査後このフラグが立っていれば、そのノードを複製した後、変更検査保留マークの付与されたノードを探索しながら再帰的に変更を保留され出力構造を複製する。

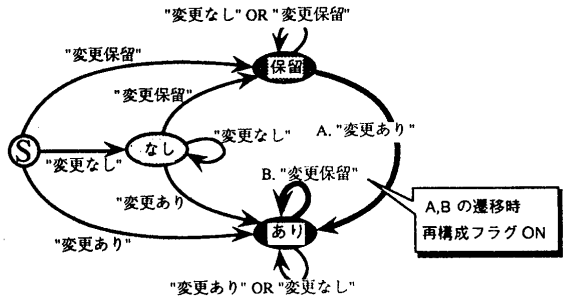

図 2. 変更検査過程における変更検査状態の遷移

5. 実験と考察

5.1 実験結果

4. で述べたループの構造共有手法を実装し、表 1. で示した実験と同じ環境下で同じ文に対する解析実験を行なった。表 2. には、従来の QDSS を使用した場合、および、ループを構造共有する手法を使用した場合について、一文の解析過程で複製された node の数を示している。また、参考として、ループ上の node の個数とそのうちで構造共有された node の個数を示す。

実験の結果、ループ上の node の構造共有率は 3.15% と、構造共有による効果がみられた。解析中複製される node 個数全体に対しては、わずか 1% 未満の向上率であった。また、処理速度は、ループをコピーする場合を 100 とすると、ループを構造共有した場合は 103.53 と、ほぼ同等の性能を保った。

表 2. 実験結果

	コピーされたノードの数 (個)		ループ上のノード (個)	
文 ID	QDSS	LoopShare	全体	構造共有個数
1	3,074	3,016	600	58
2	1,409	1,371	282	38
3	7,612	7,405	2,313	207
4	25,386	25,322	7,112	64
5	16,383	16,206	1,847	177
6	59,125	58,129	21,821	996
7	47,814	47,104	14,500	710
8	114,869	114,110	29,860	759
9	237,837	236,886	44,590	951
10	48,338	48,064	11,338	274
total	561,847	557,613	134,263	4,234
比率	100%	99.25%	100%	3.15%

●比率は, 構造共有をしない場合に対する比率を示す。

5. 2 考察

処理速度は、QDSS に対してほぼ同等の性能であった。これは実装環境 (ポインタを辿る CPU の速さ) に依存するものである。しかも、複製される nodeは必ず減少し、空間的コストが削減できることから、ループ上の node が変更される率の少ない場合には、QDSS と比較して本手法が有利であり、さらに性能向上が見込める。

このことは、ループの使用頻度や形状、使用コストなどを考慮に入れた文法記述の検討の必要性を示している。

また、本研究により、Wroblewski [1] のデータ構造 (arc に 世代管理機構を持たない構造) を使用した単一化において、理論上可能な構造共有を全て行なったことになるが、依然として root node から変更箇所までがコピーされてしまう。今後は、この根本的な問題点を如何に解消していくかを検討すべきであろう。

6. 今後の課題

今後の課題として、素性ごとの単一化成功率をヒューリスティクスとした早期失敗発見による高速化や、選言的素性構造単一化における素性構造の等値性の検査履歴の記憶 (cache) による高速化などを検討している。

謝辞

本研究の機会をくださるとともに適切な助言を頂いたATR自動翻訳電話研究所森元室長に感謝致します。議論に参加して下さったデータ処理研究室に所属する研究員の方々に感謝致します。

参考文献

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

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

[3] Karttunen, L. and Kay, M., "Structure Sharing with Binary Trees", Proc. of the 23rd ACL, 1985.

[4] Pereira, F., "A Structure-Sharing Representation for Unification-Based Formalisms", Proc. of the 23rd ACL, 1985.

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

[6] Emele, M., "Unification with Lazy Non-Redundant Copying", Proc. of the 29th ACL, 1991.

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

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

[9] Gunji, T., Japanese Phrase Structure Grammar --- A Unification-Based Approach, Dordrecht, Reidel, 1987.