5C−7

# 論理型言語上の単一化文法のための並列単一化

菅井 猛　　溝口 文雄

東京理科大学　　理工学部

## 1　はじめに

単一化に基づいた文法を計算機で作ろうとすると、様々な言語現象を素性構造として、表現しようとするため、素性構造が大きくなる。この素性構造間の unification により、効率が悪くなる。その解決策として、並列処理を行なって、全体の効率を挙げることが考えられる。本論文は、論理型言語上の単一化に基づく文法の並列処理について考察する。

## 2　単一化文法のための並列単一化

単一化に基づいた文法を並列論理型言語上 (GHC,KL1) で行なってきた。単一化に基づいた文法を作る上での重要なことは、単一化の操作とその効率である。この記述方法には、次の2つのアプローチが考えられる。

1. トランスレータによって、feature structure を変換する方法[1]
2. feature structure の unification をそのまま記述する方法

1. をコンパイラー方式 [1] と呼ぶ。この時、文法の中の reentrant structure は、トランストレータによって、基底項表現されるまた、disjunction を含んだ feature structure の単一化が、Kasper[3] によって提案されているが、1. のアプローチは disjunction や negation の記述はトランスレータに任せることによって、これらを扱えることができる。トランスレータによって、否定は選言の形に変換され、その選言を AND 並列に実行して、効率を挙げている。また、ここで扱っている否定や選言は、atomic value の否定と選言に限っている。それに対して、2では、graph unification[5] の中で、feature structure 内の否定や選言を扱わなければならない。また、disjunction を含んだ構造を扱うことは、組合せ的な曖昧性を1つのデータ構造に圧縮できるという利点がある。

## 3　term unification と feature structure の unification

定理証明で使われている term unification と計算言語学で用いられている feature structure の unification の違いを簡単に述べる。素性構造は、arity の概念がないことと、引数の数の概念がない。例えば、次のような論理プログラミングの項は、unify することができない。

```
head(agreement(number(plural),person(third)))
head(agreement(person(third),number(plural)))
```

これは、素性構造としてみた場合、属性／属性値の間に順序がなく、任意である。これは、論理プログラムの unification の操作を、素性構造の扱いにおいて使うことができないことを示している。

- 論理プログラミングの unification では、引数の順番が関係してくるのに対して、feature structure の単一化は、引数の数、順序は自由である。
- reentrant structure を論理プログラミングのオブジェクト変数として記述することはできない。

また、reentrant structure は、それ自体構造を持っているのと、構造をオブジェクト変数として共有するものの2種類がある。一方、論理プログラミングの変数は、共有変数として作用する。

論理プログラミングの unification のインプリメンテーションは、occur check を省略することによって、効率を挙げている。一方、grath unification でも、ループ構造をどのように扱うかが効率化と結び付いてくる。

## 4　データ構造

データ構造は次のような順序付き二分木を用いる。

obt(頂点のノード、頂点のノードよりもパスが小さいノード、頂点のノードよりもパスが大きいノード)

のようになる。KL1 では、大きさ4のベクタを用いて表現される。この時、feature structure は、

$$FS_1 \sqsubseteq FS_2 \sqsubseteq \ldots \sqsubseteq FS_n$$

という subsumption の関係にあるので、ベクタのデータの割り当ては、1つのセルが成長していくことになる。つまり、unification によって、feature structure が減ることはない。このデータ構造を作り出すのは、次のような前処理が行なわれる。

1. path expansion への変換

---

Parallel Unification for Unification-Based Grammar on Logic Programming

Takeshi SUGAI , Fumio MIZOGUCHI

Science University of Tokyo

2. バランスのよい順序付き二分木への変換

3. feature structure のノードの数の記述

文法記述形式は、Kasper の論理の中で使われている path expansion に変換されたものを用いる。2. は、path を key とすると、取り出しまでの時間は、$\log(n)$ となる。2 分探索木に対する探索は、データ量に対して、記憶場所の使用効率がよいこと、key になるパスについて順次アクセスすることが可能な点があげられる。3. は、句構造規則内の unification では、feature strcuture が小さいものと、大きいものとの単一化が起こることが多いので、大きな構造に小さな構造を付け加えることによって、unification を行なっている。

## 5 並列化の利点

Commint Choice Language のプログラミング方法であるショートサーキット法によって早期に unification の失敗が見つかる。これは、逐次的な処理だと最後のノードに失敗が見つかった時には、その前の unification の操作が無駄になってしまう。逐次的な方法だと、このような失敗を見つけるために、feature structure の並びかえを行なわなければならない。unification の失敗の早期発見が、パーザの効率化に影響してくる。我々の文法では、単一化文法におけるパーザの unification の約 9 割が失敗に終っている。graph のトップのノードが 10、path の長さが 10 の feature structure で、最後の path で失敗する場合、並列化すると約 $\frac{1}{4}$ のリダクション数で済んでいる。このことを考えると、ショートサーキット法によるプログラミングは非常に有効であることがわかる。

## 6 考察

Karttunen の方法 [2] は、データ構造に二分木することと、データ構造の copy を最小にするための lazy copy の方法を採用している。この 2 分木のデータ構造では，できるだけこの構造を平衡 (balance tree) になるようにして，それぞれの素性のアクセス時間が一定になるようにしている。しかし、Karttunen の方法は、句構造規則との関係が明らかではない。

Pereira の方法 [4] では、Prolog のインプリメンテーションによって実現されている structure sharing の方法を feature structure に適用している。この方法では、ガーベージコレクションがしにくくなる。

Wroblewski の方法 [5] は、再帰的なユニフィケーションがすべて成功したものみ、コピーノードに書き込みを行なう。graph unification をインプリメントする問題点として、次のことを挙げている。

1. over copying

2. early copying

これらの方法は、feature structure の unification を行なう時、もとの feature structure を unification によって破壊してしまうため、元の feature structure のコピーを保存している。このコピーにかかるコストを削減、省略することによって、feature structure の効率化をはかっている。

1. と 2. は、論理型言語のパーザの場合、データの共有は、句構造規則が、成功する時に、論理変数によって feature structure のプロセスが生まれる。つまり、unification の失敗を早期に見つけることによって解決される。論理型言語のパーザでは、文法の曖昧さごとにプロセスが分かれるので、そのプロセスごとに、素性構造を持ち歩くことになる。そして、もし、そのプロセスが失敗した時、素性構造を格納してあったセルはガーベージコレクションの時に、回収される。

## 7 おわりに

単一化に基づいた文法の枠組みでは、積極的に reentrant structure を取り込もうとしており、これらの unification を考察することは意義あることである。また、選言を直接扱うインプリメンテーションは、今後の課題である。

## 参考文献

[1] Susan Hirsh. P-PATR: A Compiler for Unification Based Grammars. In Veronica Dahl and Patrick Saint Dizier, editors, *Natural Language Understanding and Logic Programmnig II*. Elsevier Science, 1987.

[2] L. Karttunen and M. Kay. Structure Sharing with Binary Tree. In *Proceedings of ACL-85*, 1985.

[3] Robert T. Kasper. A Unification Method for Disjunctive Feature Desciptions. In *Proceedings of ACL-87*, 1987.

[4] Fenando C. N. Pereira. A Strcuture-Sharing Representation for Unificaton Grammar Formalism. In *Proceedings of ACL-85*, 1985.

[5] D. Wroblewski. Nondestructive Graph Unification. In *AAAI 87*, 1987.