様式 C-19

科学研究費補助金研究成果報告書

平成　２１年　６月　１０日現在

研究種目：　基盤研究（C）
研究期間：　２００７～２００８
課題番号：　１９５００１３１
研究課題名（和文）　拡張文脈自由文法の漸次学習方式とその応用

研究課題名（英文）　Incremental Learning of Extended Context Free Grammars and Its Applications

研究代表者
中村克彦　(NAKAMURA KATSUHIKO)
東京電機大学・理工学部・教授
研究者番号：９００５７２４０

研究成果の概要: 文脈自由文法を正負の記号列例から合成する漸次学習方式を改良すると共に，これを確定節文法(DCG)と構文的翻訳図式（SDTS）の学習などに拡張した。さらにこれらの応用として，プログラム言語の文法とコンパイラを機械学習する方式，および幾何学図形および輪郭図形の構文的パタン認識，特にパタンの学習方式，言語認識のためのセルオートマトンの合成について検討し，実験によって有効性を確認した。

交付額

（金額単位：円）

	直接経費	間接経費	合　計
2007 年度	1,200,000	360,000	1,560,000
2008 年度	400,000	120,000	520,000
年度			
年度			
年度			
総　計	1,600,000	480,000	2,080,000

研究分野：　総合領域
科研費の分科・細目：　情報学・知能情報学
キーワード：文法推論，機械学習，確定節文法，漸次学習，セルオートマトン，構文的翻訳図式, SDTS

１．研究開始当初の背景

与えられた正負の記号列の例からこれを満足する形式文法を自動的に合成する文法推論は，帰納推論および機械学習の基礎として，また認知科学における幼児の言語獲得のモデルとして重要な研究テーマであり，近年世界中で研究がさかんになっている。しかし，この分野の多くの研究は，制限の強い言語について計算量の観点から理論的な限界を明らかにすることが目的であり，実際に意味のある文法を学習する方式およびその応用についての研究は現在もあまりみられない。

われわれは初め帰納的 CYK アルゴリズムと呼ばれる規則合成方式と最小規則集合探索法と組み合わせて文脈自由文法(CFG)の漸次学習(incremental learning)システム Synapse を作成してその有効性を確認した。この規則合成法では，上向き構文解析のための CYK アルゴリズムにもとづいて，正例の記号列に対する構文解析が成功するように規則が生成され追加される。その後の研究によって，Synapse にはブリッジ方式と呼ばれる新しい規則合成方式と，最小規則集合探索方式に加えて準最適規則集合探索と呼ばれ

る探索方式が組み込まれた。準最小規則集合探索は最小な規則集合の探索を目標とせずに，より短い時間で最小に近い規則集合を求める探索方式であり，これには直列探索と呼ばれる探索法のほかに，最小ではない非終端記号集合を許容すること，規則の形式の制限などによって実現される。

Synapse はあいまいな文法の他に非あいまいな文法を合成できること，および最小または最小に近い規則集合を合成できることを特徴としている。

２．研究の目的

本研究の目的はこれまで進めてきたCFG の漸次学習方式を基本として，自動合成可能な文法の範囲をCFG 以外に拡張し，この文法推論を構文的パタン認識などのいくつかの分野の他の機械学習へ応用することである。本研究で拡張CFG として主に扱った文法は，確定節文法（DCG: definite clause grammar）および構文的翻訳図式（SDTS: syntax- directed translation scheme）である。SDTS は同期CFG とも呼ばれるように二つのCFG の規則を関連付けることによって二つの言語の間の翻訳規則を記述できる。

３．研究の方法

本研究は理論的な検討とこれを実証する文法推論システムを作成して文法の学習の実験を行うことによって進められる。実際にはこれまで改良を続けてきた Synapse システムを拡張して，DCG および SDTS の学習機能を中心とする各種の機能を組み込むことが基本となる。Synapse は，正例を上向き構文解析した結果（不完全な導出木）から導出木を完成させるため規則を生成する bridging と呼ばれる規則合成方式と，与えられた例を満足する最小または準最小の規則集合を探索する機能から構成されている。

(1) 文脈自由文法は Chomsky 標準形と呼ばれる `p → a` および `p → q r` （`a` は終端記号，`p, q, r` は非終端記号）の形式の規則で表すことができることが知られている。Synapse は次の形式の規則を合成する。

① 変形 Chomsky 標準形 $p \rightarrow \beta\ \gamma$

② 拡張 Chomsky 標準形

$p \rightarrow \beta$ または $p \rightarrow \beta\ \gamma$

ここで，p は非終端記号，β および γ は終端または非終端記号である。

(2) 確定節文法（DCG）は論理プログラミングに基礎をもち，CFG の生成規則と類似した形式の規則（文法規則と呼ばれる）を用いる。この規則に含まれる非終端記号に変数を含む項（DCG 項）を付加することによって文脈自由言語を超える言語を導出し，また構文解析することができる。DCG の例を次に示す。ここでは `1`, `N` および `t(N)` が DCG 項である。

```
p(1) --> a.   p(t(N)) --> a, p(N).
q(1) --> b.   q(t(N)) --> b, q(N).
r(1) --> c.   r(t(N)) --> c, r(N).
s(N) --> p(N),q(N),r(N).
```

`s(X)` を開始項とするとき，この DCG は非文脈自由言語 $\{a^n b^n c^n \mid n \geq 1\}$ の記号列を導出し，受理する。これらの文法規則は簡単に Horn 節に変換して，構文解析用の Prolog のプログラムとして実行することができる。

(3) Synapse システム は 正例の順序集合と負例の集合，また初期規則の集合を入力として，すべての正例を導出し，負例を導出しない規則集合 P を探索する。正例と負例は，文字列と開始記号の対として与えられる。

開始記号が DCG 項をもつ場合には，DCG 項として記号列の構文解析の結果得られる項またはその一般化した項と開始項の対を与える。上記の言語に対する正例の一部を次に示す。

```
[a,b,c] - s(1).
[a,a,b,b,c,c] - s(t(1)).
[a,a,a,b,b,b,c,c,c] - s(t(t(1))).
[a,a,a,a,b,b,b,b,c,c,c,c]       -
                    s(t(t(t(1)))).
```

(4) Synapse における bridge 法による規則合成手順の概要は次のような非決定的な手続きとして記述される。

① 正例の記号列に対して，それまでに得られた規則集合を用いて上向きの構文解析を行う。構文解析の結果が成功ならば規則合成は終了（成功）。

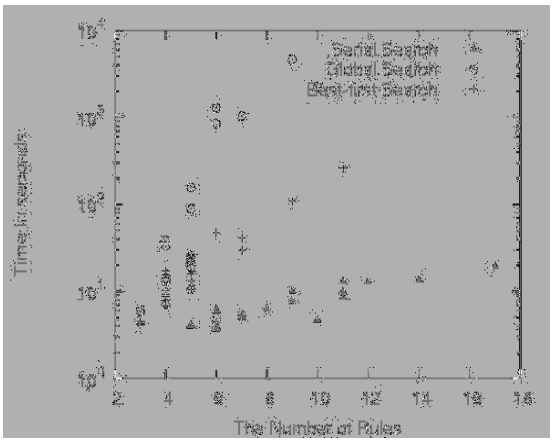
② 構文解析が失敗したとき，構文解析の結果である不完全な導出木に対して，欠けた部分を探索し，導出木を完成するような規則を生成する。この探索には，それまでに得られた規則も用いられる。

③ DCG 規則の合成において生成される規則が非終端記号を含む場合，正例がもつ DCG 項に含まれる部分項を一般化したものを非

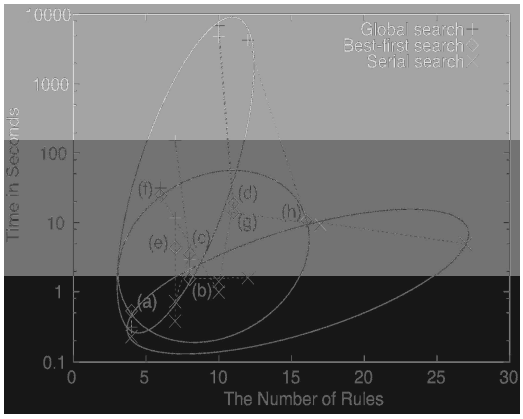
決定的に選択して，これを DCG 項として付加する。ある項の一般化とは，その部分項を変数に置き換えることである。
④ 生成された規則を含む規則集合が負例を導出しないことをテストする。どの負例も導出しないなら終了（成功）。そうでなければ失敗して終了（前の分岐点にバックトラックする）。
(5) Synapse は次のいずれかの探索方式によって，すべての正例を導出し，負例を導出しない規則集合を出力する。探索の様子は，各節点が合成された規則集合のラベルをもち，木の根から深さが k の節点のラベルが，最初から第 k 番目までの正例によって合成された規則集合であるような探索木によって表される。
① 最小規則集合探索(global search)：与えられたすべての正例を導出し，すべての負例を導出しない最小の規則集合を規則数についての反復深化によって求める。
② 直列探索(serial search)：各正例について順にこの記号列を導出し，すべての負例を導出しない最小の規則を反復深化によって求め，規則集合に漸次追加する。これは探索木の一つの経路についてのみ規則集合を求めている。
③ 最良優先探索(best-first search)： 探索木の各節点を評価し，もっとも最終的な規則集合に近いとみられる節点から探索木を拡張する。この方式は，探索を進める候補の節点をある個数に限定するため，ビーム探索とも呼ばれる。
これまでの Synapse は最小規則集合探索方式と準最小規則探索方式として直列探索を採用していた。最近になって最良優先(ビーム)探索方式が組み込まれた。

４．研究成果

(1) ランダムに発生させた規則数10以内の約1000個のCFG に対してその言語の例から，3種類の探索方式を用いてCFGを学習によって合成した。この実験によって，次のような結果が得られた。
① すべての言語について，最小規則集合探索によって最初に発生させたDCGより少ない規則数の文法が合成された。
②直列探索によって約43%のCFGについて最小の規則集合が合成され，約85%のCFGに対して規則数の増加は3以内であった。ただし，3個のCFGが直列探索によっては合成できなかった。
③ 最良優先探索は最小規則集合探索と直列探索の中間的な結果を示す。
次の図はこの実験で得られたCFG の規則数と合成時間との関係を示している。



次の図はいくつかの基本的な言語の CFG を 3 種類の探索方式によって合成させたときの，規則数と計算時間との関係を示している。点線は同じ言語に対する 3 種類の文法を結んでいる。



ここで，記号は (a) 対応の取れた括弧からなる記号列の集合，(b) a と b からなる回文の集合，(c) 正則表現の集合，(d) a と a の数の 2 倍の b からなる記号列の集合，(e) ww の形をしていない a と b からなる記号列の集合，(f) $\{a^i b^j c^k \mid i = j$ または $j = k\}$， などの言語を表している。
(2) DCG および DCG にもとづく SDTS の自動合成を含む次のようないくつかの例題についてテストし，この方式の有効性を確認した。なお，GR は生成された全規則数であり，記号列例や使用コンピュータに依存しない

計算量を表わす指標として用いられている。
① 言語 $\{a^n b^n c^n \mid n \geq 1\}$：上記3(2) にあげた DCG はこの言語の文法である。全域探索によって 3243 秒, GR =7×10^5 で３規則が合成された。
② Fibonacci 数列を1進法で表した記号列の集合： 最小の規則集合は次の３規則である。

```
s(1) → [a].
s(t(1)) → [a].
s(t(t(X))) → s(X), s(t(X)).
```

正例の一部を次に示す。

```
[a] - s(1).          [a] - s(t(1)).
[a,a] - s(t(t(1))).
[a,a,a] - s(t(t(t(1)))).
[a,a,a,a,a] - s(t(t(t(t(1))))).
```

上の3規則が全域探索によって0.95秒, GR = 254，最良優先探索によって5秒，GR =831で合成された。また，直列探索によって5つの規則集合のDCG が2.1秒，GR =1289で合成された。
③ 合成数（素数でない数）を1進法で表した記号列の集合：次の *印をつけた規則を初期規則として与えたとき，最後の２規則が合成された。

```
*unit(1) → [a].
*count(t(X)) → count(X), unit(1).
*count(t(1)) → unit(1), unit(1).
 s(t(X)) → count(t(X)), s(t(X)).
 s(t(X)) → count(t(X)), count(t(X)).
```

正例の一部を次に示す。

```
[a,a,a,a] - s(t(1)).
[a,a,a,a,a,a] - s(t(1)).
[a,a,a,a,a,a,a,a] - s(t(1)).
[a,a,a,a,a,a,a,a,a] - s(t(t(1))).
[a,a,a,a,a,a,a,a,a,a] - s(t(1)).
```

この言語に対するDCG規則が，全域探索によって195秒, GR =6782，また最良優先探索によって90秒, GR= 271で合成された。
(3) DCG およびSDTSの漸次学習の応用としてプログラム言語の文法とコンパイラをソースプログラムとこれに対応する中間言語のコードの対の例から学習する新しい方式について検討し，実験を行った。この結果，代入演算および関数呼び出しなどを含むように拡張した算術式について，非あいまいな文脈自由文法とProlog言語のコンパイラを自動合成することができた。
(4) 負例を実際の記号列だけでなく，変数を含む記号列のパタンで表わす機能，および合成された文法から長さの順に記号列を生成させる機能が付加された。これらの機能によってよって，文法をより少ない数の負例で学習させ，また，合成された文法の正しさを容易に検証できるようになった。
(5) 文脈自由文法および確定節文法とその規則合成方式を，幾何学図形および輪郭図形の構文的パタン認識，特にパタンの学習に応用するための検討と実験を行った。さらに，輪郭図形を扱うために，循環記号列の集合を言語とする循環CFG の構文解析と自動合成法について調べた。
(6) 本研究の学習方式を，形式言語を受理する1次元セルオートマトンの学習に応用するための検討と予備的な実験を行った。理論的な検討として，1次元セルオートマトンが実時間認識できる言語の限界について調べた。

５．主な発表論文等
（研究代表者、研究分担者及び連携研究者には下線）

〔雑誌論文〕（計　２件）
① Katsuhiko Nakamura, Language not recognizable in real time by one-dimensional cellular automata, Jour. of Computer and System Sciences, Vo. 74, 1095-1102, 2008, 査読有.
② Katsuhiko Nakamura, Towards a Basis for Parallel Language Recognition by Cellular Automata, Journal of Cellular Automata, Vol. 4, 201-211, 2009, 査読有.

〔学会発表〕（計　１件）
Keita Imada and Katsuhiko Nakamura, Towards Machine Learning of Grammars and Compilers of Programming Language, Machine Learning and Knowledge Discovery in Database (ECML PKDD 2008), Antwerp, Belgium, Part II, LNAI 5212, 98-112, 2008, 査読有.

６．研究組織
(1)研究代表者
中村克彦（NAKAMURA KATSUHIKO）
東京電機大学・理工学部・教授
研究者番号:90057240

(2)研究分担者　なし

(3)連携研究者　なし