複雑な文法を効率的に使う

プログラム変換を用いた多重文脈自由言語の効率的解析

金沢　誠（情報学プリンシプル研究系）

何が分かる？

自然言語にまれに見られる交差する依存関係を含む構文の記述や疑似ノットを含むRNAの二次構造の予測などに使われている多重文脈自由文法の構文解析のための効率的アルゴリズムを簡単に導く方法がわかります。

どんな研究？

形式言語理論という理論計算機科学の分野と計算言語学に属する研究です。論理プログラミング、データベース理論とも関係があります。

人間の言語やプログラミング言語には、主語や目的語と動詞、**if**と**else**や**begin**と**end**など、呼応する要素の対が入れ子状に配置される構造がごく普通に見られます(①, ②)。プログラミング言語では、**if ... begin ... else ... end**のように呼応する要素の対が交差するような構文は許されません。しかし、人間の言語には、オランダ語やドイツ語のスイス方言の従属節に見られるように、まれに呼応する要素の対が交差する構造が見られます(③)。同様に、RNAの二次構造においては結合する塩基対が入れ子状に配置される場合がほとんどですが(④)、**疑似ノット**という塩基対が交差する構造も存在します(⑤)。

入れ子状の構造は**文脈自由文法** (CFG) によって表現することができます。例えば、$\mathrm{a}^m\mathrm{b}^n\mathrm{c}^n\mathrm{d}^m$というパターンは、aとd、およびbとcのあいだに依存関係がある入れ子状の構造を持ちますが、次の CFG で表現できます。

$$S(\mathrm{a}x\mathrm{d}) \leftarrow S(x). \quad S(x) \leftarrow T(x). \quad T(\mathrm{b}x\mathrm{c}) \leftarrow T(x). \quad T(\epsilon).$$

これに対し、$\mathrm{a}^m\mathrm{b}^n\mathrm{c}^m\mathrm{d}^n$のような依存関係が交差するパターンは、CFG で表現することはできず、**多重文脈自由文法** (MCFG) のようなより表現力の強い文法形式を必要とします。CFGの規則が１つの文字列の作り方を規定するのに対して、MCFGの規則はいくつかの文字列からなる組の作り方を規定します。

$$S(x_1y_1x_2y_2) \leftarrow P(x_1, x_2), Q(y_1, y_2).$$
$$P(\mathrm{a}x_1, \mathrm{c}x_2) \leftarrow P(x_1, x_2). \qquad P(\epsilon, \epsilon).$$
$$Q(\mathrm{b}y_1, \mathrm{d}y_2) \leftarrow Q(y_1, y_2). \qquad Q(\epsilon, \epsilon).$$

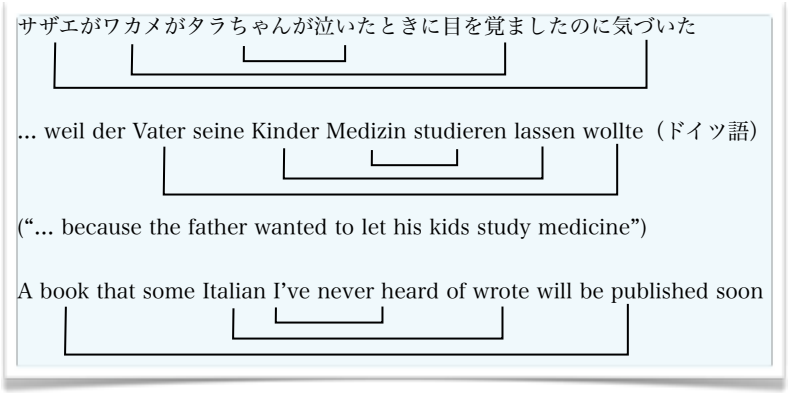
上のMCFGでは、$(\mathrm{a}^m, \mathrm{c}^m)$と$(\mathrm{b}^n, \mathrm{d}^n)$という文字列の対２つを独立に構成した上で最後に$\mathrm{a}^m\mathrm{b}^n\mathrm{c}^m\mathrm{d}^n$という１つの文字列にまとめる方法を規定することによって、$\mathrm{a}^m\mathrm{b}^n\mathrm{c}^m\mathrm{d}^n$というパターンを表現しています。

人間の言語の文やプログラムを解析するには、入力文字列が文法にのっとってどのような過程で導出されたかを示す**導出木**(⑥)という構造を求める必要があります。同様に、RNAの一次配列から二次構造を予測するには、文字列で表される与えられた配列の導出木の中から最も確率の高いものを求める必要があります。入力文字列から導出木を求めることを**構文解析**と呼びます。

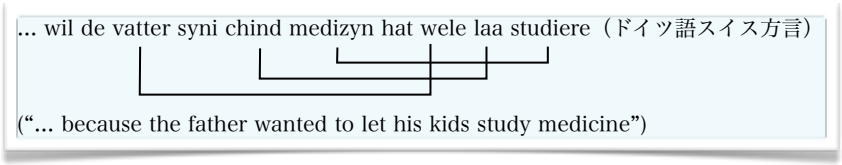
CFGに対する最も効率的な構文解析アルゴリズムのひとつに**Earleyのアルゴリズム**があります。Earleyのアルゴリズムは、論理プログラミングにおける**マジックセット書き換え**の応用と理解することができます。CFGをDatalogという

```
for j ← 2 to length[A] do
  begin
    key ← A[j]
    i ← j − 1
    while i > 0 and A[i] > key do
      begin
        A[i + 1] ← A[i]
        i ← i − 1
      end
    A[i + 1] ← key
  end
```

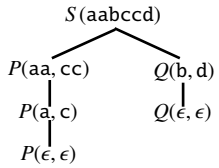
①プログラムの入れ子構造



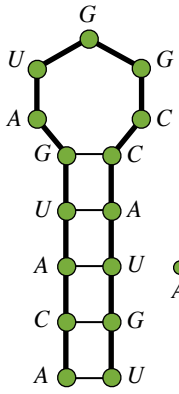
②入れ子状の依存関係を含む構文

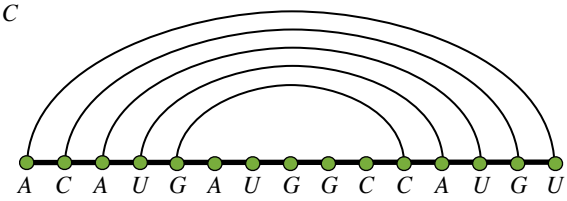


③交差する依存関係を含む構文

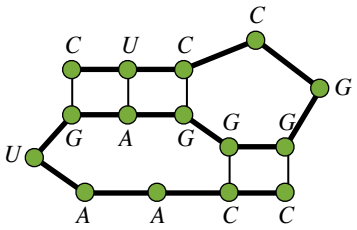


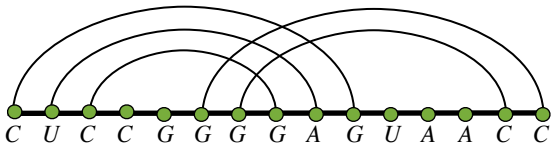
⑥MCFGの導出木



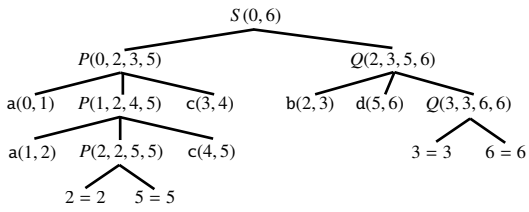


④疑似ノットを含まないRNA二次構造





⑤疑似ノットを含むRNA二次構造



⑦MCFGに対応するDatalogプログラムの導出木

制限された論理プログラミングのプログラムで表し、マジックセット書き換えを適用するとEarleyのアルゴリズムの手続きを得ることができます。

MCFGもCFGと同様、Datalogで表現することができます。

$$S(i,m) \leftarrow P(i,j,k,l), Q(j,k,l,m).$$
$$P(i,k,l,n) \leftarrow \mathsf{a}(i,j), P(j,k,m,n), \mathsf{c}(l,m).$$
$$P(i,j,k,l) \leftarrow i = j, k = l.$$
$$Q(i,k,l,n) \leftarrow \mathsf{b}(i,j), \mathsf{d}(l,m), Q(j,k,m,n).$$
$$Q(i,j,k,l) \leftarrow i = j, k = l.$$

したがって、マジックセット書き換えを使ってMCFGに対してもEarley流のアルゴリズムを得ることが可能です。しかし、こうして得られたアルゴリズムは、correct prefix propertyという、入力文字列を左から右に処理し、誤りを即座に検出するというEarleyのアルゴリズムの利点を失ってしまいます。マジックセット書き換えは、元のプログラムの導出木(⑦)を深さ優先で左から右にたどってゆくことに対応し、MCFGの場合は、導出木における文字の出現順序が入力文字列における出現順序と一致しないためです。

この問題を解決するために、マジックセット書き換えの前にもう1つの簡単な書き換えを適用し、導出木における文字の出現順序が入力文字列における文字の出現順序に一致するようにします。こうしておいてマジックセット書き換えを適用した結果得られるアルゴリズムは、効率性を損なうことなくcorrect prefix propertyを満たします。