型付きラムダ計算に基づく包括的な文法モデルの構築

Toward a Comprehensive Model of Grammar Based on the Typed Lambda Calculus

金沢　誠　Makoto Kanazawa
吉仲　亮　Ryo Yoshinaka
Sylvain Salvati

何がわかる？

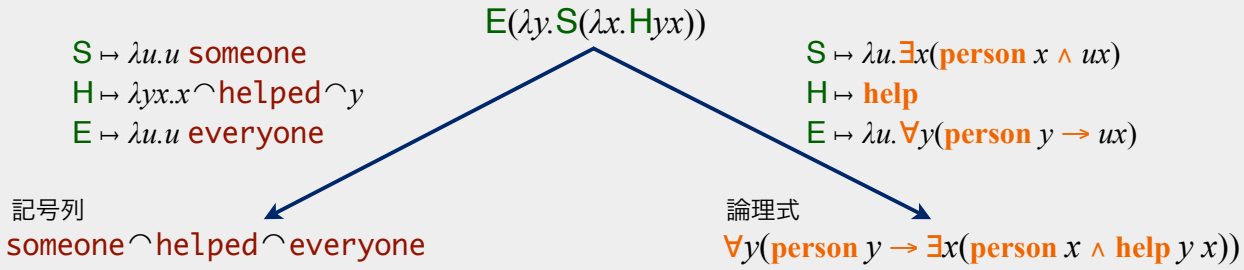
統語論（シンタクス）と意味論（セマンティクス）を統一的に表現できるACG（抽象的範疇文法）という文法フォーマリズムの数学的性質を調べています。ACGの表現力、解析・生成のアルゴリズム、効率的学習可能性の問題をいろいろな制約のもとで明らかにすることによって、人間の言語の文法の多様な側面を捉えた数学的モデルをACGの枠組みの中で構築することを目標としています。

どんな研究？

人間の言語の科学的理解の助けとなるような数学的理論を提供する数理言語学の分野の研究で、理論計算機科学の一分野である形式言語理論と密接な関係にあります。取り上げているACGという文法フォーマリズムは型付きラムダ計算を使って定式化されているため、数理論理学（特に型理論・証明論）の応用という側面も持っています。

題　材

ACGは、型付けられた定項（抽象的定項）から作られるラムダ項（抽象的項）に対して、二種類の代入を施して得られる二つのラムダ項を簡約することによって形（記号列や木）と意味表現（論理式）の対を生成する文法フォーマリズムです。



研究状況

一次元ACGの表現力

形の次元のみに注目して、どんな記号列言語や木言語が生成できるかという問題です。一般の場合にはよくわかっていませんが、抽象的定項の型を２階以下に制限した２階ACGについてはほとんど解決されました。

代入の複雑度	記号列言語	木言語
1		REGT
2	CF	CFT_{sp}
3	$yCFT_{sp}$	MREGT⊊
≥4	MCF=STR(HR)	TR(HR)

ACGの表現できる木変換

二つの次元でどちらも木を生成するように単純化したACGの表現できる関係を調べました。

代入の複雑度	木変換
1, 1	T_{si}⊆
1, 2	$MTT_{si,sp}$⊆

解析・生成

固定した文法のもとで、形からそれを引き出す抽象的項を見つけ意味表現を求める問題が解析で、逆に意味表現からそれを引き出す抽象的項を見つけ形を求める問題が生成です。２階ACGは多項式時間で解析できることがすでにわかっていますが、解析のアルゴリズムについてさらに詳しく調べています。２階ACGに対するEarley型の解析アルゴリズムや、データベース・クエリー言語DATALOGの評価アルゴリズムへの帰着を用いたアルゴリズムを研究しています。

さらに、解析アルゴリズムを一般化する形で生成アルゴリズムを定式化することを試みています。

学習

記号列と意味表現の対からなるデータから文法を学習するアルゴリズムを研究しています。意味の次元である特殊な制限を持ったACGに対して論理学において補完論理式を求めるアルゴリズムと高階マッチングの特殊な場合を解くアルゴリズムを応用した学習アルゴリズムを開発中です。

NII 連絡先：金沢 誠（Makoto KANAZAWA）／ 国立情報学研究所　情報学プリンシプル研究系　助教授
TEL : 03-4212-2651　FAX : 03-3556-1916　Email : kanazawa@nii.ac.jp

Q & A

Q. 形式文法とは何ですか？

A. 一口に言うと、人間の言語の文法の数学的モデルのことです。言語学では文法によってその言語の単語から作られた「文法的な文」からなる無限集合が定まると考えますが、この点に着目すると、ある有限個の記号（単語）を組み合わせて作られる整合的な表現（文）の集合を帰納的に定義するような形式的体系を文法のモデルと考えることができます。このような形式的体系が形式文法です。

Q. 文法フォーマリズムとは何ですか？

A. 個々の文法の記述の仕方と、文法と文の集合のあいだの対応を厳密に定める規定のことです。何をもって「文法」と呼ぶか、文法が文の集合をどのように定めるかは、文法フォーマリズムによって異なります。

Q. 形式文法の研究は人間の言語の理解に役立つのですか？

A. 一般に、ある自然現象の数学的モデルを数学の対象として研究することがその自然現象の科学的理解に役立つ可能性は大いにあります。人間の言語の研究に限って言えば、人間が脳に内在化している文法とはどのようなものなのか、人間が文を発話したり文を理解する過程で従っているアルゴリズムはどのようなものか、そして人間が母国語を習得する過程で従っているアルゴリズムはどのようなものかという疑問に科学的な答えを与えようとする研究にとって、文法フォーマリズムの記述力、解析や生成のアルゴリズム、学習のアルゴリズムに関する数学的研究は強力な道具となります。

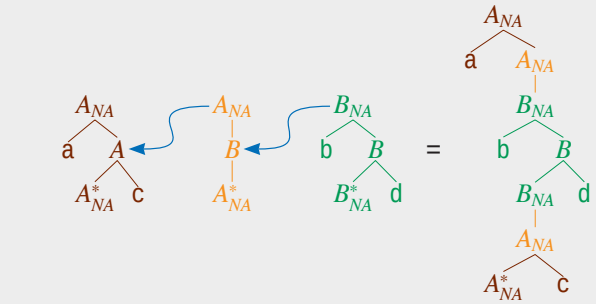
現実には、理論言語学・心理言語学の研究はまだ形式文法の研究からさほど大きな影響を受けてはいませんが、将来これらの分野が成熟するためにはより数学的な理論化が必要です。

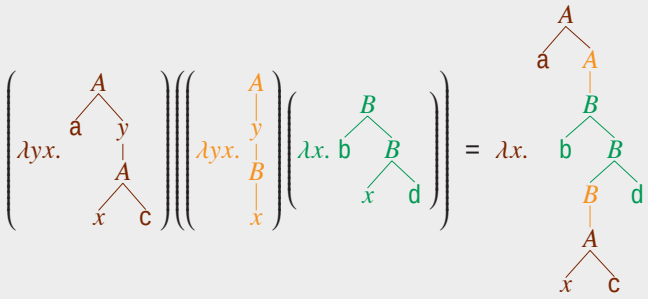
Q. 数理言語学でよく研究されている文法フォーマリズムにはどのようなものがありますか？

A. 人間の言語の記述のために考えられた代表的な文法フォーマリズムに、プログラミング言語の記述にも使われる文脈自由文法があります。文脈自由文法には人間の言語の記述には不十分な点もあるため、より記述力が高いが文脈自由文法の持つ多くの望ましい性質を引き継いでいるいくつかの文法フォーマリズムが1980年代以降よく研究されています。そのひとつが木接合文法です。木接合文法は、文脈自由文法がとらえることのできる入れ子上の依存関係（たとえば、$\{\mathrm{a}^m\mathrm{b}^n\mathrm{c}^n\mathrm{d}^m \mid m,n \geq 0\}$ に見られるような依存関係）のみならず、文脈自由文法によってはとらえることができない交差する依存関係（たとえば、$\{\mathrm{a}^m\mathrm{b}^n\mathrm{c}^m\mathrm{d}^n \mid m,n \geq 0\}$）をもとらえることができます。交差する依存関係を示す構文は、オランダ語やドイツ語のスイス方言などいくつかの言語に存在することが知られています。

Q. ACGとはどんな文法フォーマリズムですか？

A. これまでに研究されて来た文法フォーマリズムには、大きく分けて記号列を操作するもの（記号列文法）と、木を操作するもの（木文法）の二種類があります。木文法は一義的には木の集合を定め、二義的に記号列の集合を定めます。木接合文法は木文法のひとつです。ACGは、記号列や木という特定のデータ型ではなく、線形ラムダ項によって表現可能な任意のデータ型を操作する文法で、概念的には木接合文法の一般化と理解することができます。木接合文法において公理の役割を果たす初等木と推論規則の役割を果たす接合という木に対する演算が、ACGにおいてはいずれも任意の線形ラムダ項で置き換わります。ACGは一義的には線形ラムダ項の集合を定めますが、木や記号列の線形ラムダ項による符号化をとおして木の集合や記号列の集合を定めるものと見なすことができます。文脈自由文法や木接合文法を含む既存の多くの文法フォーマリズムがACGによって自然に表現できることが知られています。





Q. 他の文法フォーマリズムにないACGの特長は何ですか？

A. 重要なのは、次の二点です。

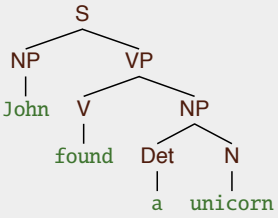
(1) 既存の多くの文法フォーマリズムにおける導出木に相当する抽象的項の定義が明示的に含まれていること。（ACGの一番目の要素である抽象的語彙は、抽象的項の集合の定義になっています。）

(2) 木や記号列のみならず、線形ラムダ項で表現可能な任意のデータ型を扱うことができること。

(2)の特長により、木や記号列に対する高階の演算を扱うことができるという特長や、論理的意味論で使われている高階論理式を扱うことができるという特長が帰結します。また、(1)の特長から、抽象的語彙を共有する２つのACGを組み合わせることが容易になり、抽象的項を統語構造であると同時に意味の合成のレシピーと見なすことによって、統語論と意味論を兼ね備えた文法を統語論のみを持つ文法と同等の形式性・一般性を持って表現することができるという特長が帰結します。

Parsing and Generation as Database Queries

Context-Free Grammar with Montague Semantics

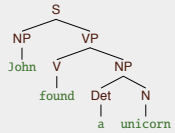


$\mathsf{S}(X_1X_2) \to \mathsf{NP}(X_1)\ \mathsf{VP}(X_2)$
$\mathsf{VP}(\lambda x.X_2(\lambda y.X_1yx)) \to \mathsf{V}(X_1)\ \mathsf{NP}(X_2)$
$\mathsf{NP}(X_1X_2) \to \mathsf{Det}(X_1)\ \mathsf{N}(X_2)$
$\mathsf{NP}(\lambda u.u\ \mathbf{John}) \to \mathtt{John}$
$\mathsf{V}(\mathbf{find}) \to \mathtt{found}$
$\mathsf{Det}(\lambda uv.\exists(\lambda y.\wedge(uy)(vy))) \to \mathtt{a}$
$\mathsf{N}(\mathbf{unicorn}) \to \mathtt{unicorn}$

$(\lambda u.u\ \mathbf{John})(\lambda x.(\lambda uv.\exists(\lambda y.\wedge(uy)(vy)))\ \mathbf{unicorn}\ (\lambda y.\mathbf{find}\ y\ x))$
$\twoheadrightarrow_\beta \exists(\lambda y.\wedge(\mathbf{unicorn}\ y)(\mathbf{find}\ y\ \mathbf{John}))$
semantic representation

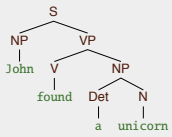
Parsing

string
John found a unicorn
$\Longrightarrow$
$O(n^3)$
derivation tree

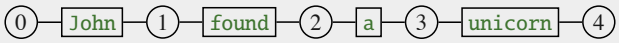


Generation (surface realization)

semantic representation
$\exists(\lambda y.\wedge(\mathbf{unicorn}\ y)(\mathbf{find}\ y\ \mathbf{John}))$
$\Longrightarrow$
?
derivation tree



Parsing with string positions



Datalog program

$\mathsf{S}(i,j)$:- $\mathsf{NP}(i,k), \mathsf{VP}(k,j).$
$\mathsf{VP}(i,j)$:- $\mathsf{V}(i,k), \mathsf{NP}(k,j).$
$\mathsf{NP}(i,j)$:- $\mathsf{Det}(i,k), \mathsf{N}(k,j).$
$\mathsf{NP}(i,j)$:- $\mathtt{John}(i,j).$
$\mathsf{V}(i,j)$:- $\mathtt{found}(i,j).$
$\mathsf{Det}(i,j)$:- $\mathtt{a}(i,j).$
$\mathsf{N}(i,j)$:- $\mathtt{unicorn}(i,j).$

database

$\mathtt{John}(0,1).$
$\mathtt{found}(1,2).$
$\mathtt{a}(2,3).$
$\mathtt{unicorn}(3,4).$

query

?- $\mathsf{S}(0,4).$

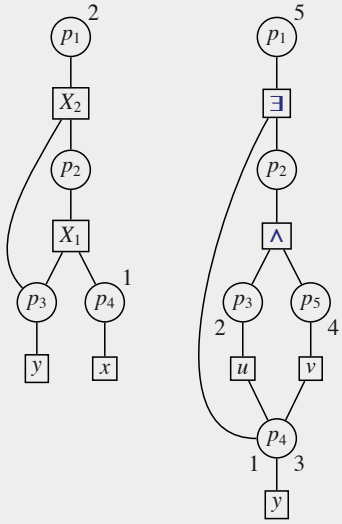
Generation as parsing of λ-terms

$\mathsf{S}(X_1X_2)$:- $\mathsf{NP}(X_1), \mathsf{VP}(X_2).$
$\mathsf{VP}(\lambda x.X_2(\lambda y.X_1yx))$:- $\mathsf{V}(X_1), \mathsf{NP}(X_2).$
$\mathsf{NP}(X_1X_2)$:- $\mathsf{Det}(X_1), \mathsf{N}(X_2).$
$\mathsf{NP}(\lambda u.u\ \mathbf{John}).$
$\mathsf{V}(\mathbf{find}).$
$\mathsf{Det}(\lambda uv.\exists(\lambda y.\wedge(uy)(vy))).$
$\mathsf{N}(\mathbf{unicorn}).$

- This grammar generates a set of λ-terms and is equivalent to a second-order *abstract categorial grammar* (ACG, de Groote 2001).
- If all λ-terms in a second-order ACG are *linear*, parsing complexity is *polynomial* (Salvati 2005).

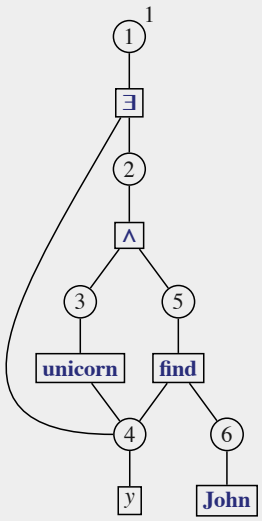
Graph representations of λ-terms

$\lambda x.X_2(\lambda y.X_1yx)$ $\quad$ $\lambda uv.\exists(\lambda y.\wedge(uy)(vy))$



Reduction to Datalog

$\exists(\lambda y.\wedge(\mathbf{unicorn}\ y)(\mathbf{find}\ y\ \mathbf{John}))$



Datalog program

$\mathsf{S}(p_1)$:- $\mathsf{NP}(p_3,p_2,p_1), \mathsf{VP}(p_3,p_2).$
$\mathsf{VP}(p_4,p_1)$:- $\mathsf{V}(p_3,p_4,p_2), \mathsf{NP}(p_3,p_2,p_1).$
$\mathsf{NP}(p_5,p_4,p_1)$:- $\mathsf{Det}(p_3,p_2,p_5,p_4,p_1), \mathsf{N}(p_3,p_2).$
$\mathsf{NP}(p_2,p_1,p_1)$:- $\mathbf{John}(p_2).$
$\mathsf{V}(p_2,p_3,p_1)$:- $\mathbf{find}(p_2,p_3,p_1).$
$\mathsf{Det}(p_4,p_3,p_4,p_5,p_1)$:- $\exists(p_4,p_2,p_1), \wedge(p_3,p_5,p_2).$
$\mathsf{N}(p_2,p_1)$:- $\mathbf{unicorn}(p_2,p_1).$

database

$\exists(4,2,1).$
$\wedge(3,5,2).$
$\mathbf{unicorn}(4,3).$
$\mathbf{find}(4,6,5).$
$\mathbf{John}(6).$

query

?- $\mathsf{S}(1).$

Complexity of generation ≤ *data complexity of Datalog* = *P-complete*.

- The reduction is correct if all λ-terms are free of *non-atomic contraction*.