Formalization of Categorial Grammar[1]

Masaya Taniguchi

Japan Advanced Institute of Science and Technology

November, 2021

[1] This work was supported by Grant-Aid for JSPS Fellows Number 21J15207.

Outline

Definition (Category)

Let A and B be arbitrary any categories. The category is inductively defined as;

- Atomic categories are $\mathsf{NP}, \mathsf{S}, \ldots$.
- Functional categories are $A/B, B\backslash A$.

We use capital Greek letters $\Gamma, \Delta, \Sigma, \Theta, \ldots$ for sequences of categories.

Example (Functional Category)

Let a lexicon ℓ be a map from a word to a category.

$$\ell(\mathsf{I}) = \mathsf{NP} \qquad \ell(\mathsf{love}) = (\mathsf{NP}\backslash\mathsf{S})/\mathsf{NP} \qquad \ell(\mathsf{you}) = \mathsf{NP}$$

Then, the sequence of words “I love you” is a sentence.

```
datatype 'a category =
  Atomic 'a ("^")
  | LeftFunctional "'a category" "'a category" (infix "←" 65)
  | RightFunctional "'a category" "'a category" (infix "→" 65)
```

Definition (Categorial Grammar (Steedman, 2000))

We use the sequent-style system. This system is *intuitionistic* and it has *no structural rules*.

$$\frac{}{A, A\backslash B \Rightarrow B}< \qquad \frac{}{B/A, A \Rightarrow B}> \qquad \frac{\Gamma \Rightarrow A \qquad \Delta, A, \Sigma \Rightarrow B}{\Delta, \Gamma, \Sigma \Rightarrow B}\text{Cut}$$

For any given lexicon ℓ, a sequence of words $w_1, \ldots, w_n$ is a sentence if and only if $\ell(w_1), \ldots, \ell(w_n) \Rightarrow \mathsf{S}$.

Slogan

The sequence is acceptable if it is derivable.

- Coq: Martín-Gómez *et.al.*, ccg2lambda: A compositional System
- Agda: Wen Kokke, Type-logical Grammar in Agda

```
inductive CategorialGrammar
  :: "'a category list ⇒ 'a category ⇒ bool" (infix "⊢CG" 60) where
    L : "[B←A,A] ⊢CG B"
  | R : "[A,A→B] ⊢CG B"
  | Cut : "⟦Γ ⊢CG A; Δ@[A]@Σ⊢CG B⟧ ⟹ Δ@Γ@Σ⊢CG B"
lemma "[NP,(NP→S)←NP, NP]⊢CG S"
proof-
  have "[(NP→S)←NP, NP]⊢CG NP→S"
    by (simp add: L)
  moreover have "[NP, NP→S] ⊢CG S"
    by (simp add: R)
  ultimately show ?thesis
    by (smt (verit, ccfv_threshold) Cut append_Cons append_Nil append_assoc)
qed
```

Example (Categorial Grammar)

For the following ℓ, the sequence of words "I love you" is a sentence because;

$$\ell(\mathsf{I}) = \mathsf{NP} \qquad \ell(\mathsf{love}) = (\mathsf{NP}\backslash\mathsf{S})/\mathsf{NP} \qquad \ell(\mathsf{you}) = \mathsf{NP}$$

$$\dfrac{\dfrac{}{(\mathsf{NP}\backslash\mathsf{S})/\mathsf{NP}, \mathsf{NP} \Rightarrow \mathsf{NP}\backslash\mathsf{S}}> \qquad \dfrac{}{\mathsf{NP}, \mathsf{NP}\backslash\mathsf{S} \Rightarrow \mathsf{S}}<}{\mathsf{NP}, \mathsf{NP}\backslash(\mathsf{S}/\mathsf{NP}), \mathsf{NP} \Rightarrow \mathsf{S}}\,\mathrm{Cut}$$

Definition (Language of Categorial Grammar)

For any given ℓ, the language L is defined as $L := \{w_1 \ldots w_n \mid n \in \mathbb{N}, l(w_1), \ldots, \ell(w_n) \Rightarrow \mathsf{S}\}$

Chomsky Hierarchy

- Turing Machine
- Context-Sensitive Grammar
  - **Mildly Context-Sensitive Grammar** (*e.g.*, Combinatory Categorial Grammar)
- **Context-Free Grammar** (*e.g.*, Categorial Grammar)
- Regular Grammar

Definition (Combinatory Categorial Grammar (Steedman, 2000))

We use the sequent-style system. This system is *intuitionistic* and it has *no structural rules*.

$$\frac{}{A, A\backslash B \Rightarrow B} < \qquad \frac{}{B/A, A \Rightarrow B} > \qquad \frac{\Gamma \Rightarrow A \qquad \Delta, A, \Sigma \Rightarrow B}{\Delta, \Gamma, \Sigma \Rightarrow B} \text{Cut}$$

$$\frac{}{A/B, B/C \Rightarrow A/C} >\text{B} \qquad \frac{}{A\backslash B, B\backslash C \Rightarrow A\backslash C} <\text{B}$$

$$\frac{}{A \Rightarrow X/(A\backslash X)} <\text{T} \qquad \frac{}{A \Rightarrow (X/A)\backslash X} >\text{T}$$

For any given lexicon ℓ, a sequence of words $w_1, \ldots, w_n$ is a sentence if and only if $\ell(w_1), \ldots, \ell(w_n) \Rightarrow \text{S}$.

```
inductive CombinatoryCategorialGrammar
  :: "'a category list ⇒ 'a category ⇒ bool" (infix "⊢CCG" 60) where
    L : "[B←A,A] ⊢CCG B"
  | LT : "[A] ⊢CCG (B←A)→B"
  | LB : "[A←B,B←C] ⊢CCG A←C"
  | R : "[A,A→B] ⊢CCG B"
  | RT : "[A] ⊢CCG B←(A→B)"
  | RB : "[A→B,B→C] ⊢CCG A→C"
  | Cut : "⟦Γ ⊢CCG A; Δ@[A]@Σ⊢CCG B⟧ ⟹ Δ@Γ@Σ⊢CCG B"
theorem
  fixes f :: "'a ⇒ 'b category" and S :: "'b category"
  assumes "LCG = {W . map f W ⊢CG S}" and "LCCG = {W . map f W ⊢CCG S}"
  shows "LCG ⊆ LCCG" apply (simp add: assms) using CG_is_CCG by blast
```

Definition (Lambek Calculus (van Benthem, 1988))

We use the sequent-style system. This system is *intuitionistic* and it has *no structural rules*.

$$\frac{}{A \Rightarrow A}\,\text{Ax} \qquad \frac{\Gamma, A, \Delta \Rightarrow B \qquad \Sigma \Rightarrow A}{\Gamma, \Sigma, \Delta \Rightarrow B}\,\text{Cut}$$

$$\frac{\Gamma, A, \Delta \Rightarrow B \qquad \Sigma \Rightarrow C}{\Gamma, \Sigma, C\backslash A, \Delta \Rightarrow B}\,\text{L<} \qquad \frac{\Gamma, A, \Delta \Rightarrow B \qquad \Sigma \Rightarrow C}{\Gamma, A/C, \Sigma, \Delta \Rightarrow B}\,\text{L>}$$

$$\frac{\Gamma, A \Rightarrow B}{\Gamma \Rightarrow B/A}\,\text{R>} \qquad \frac{A, \Gamma \Rightarrow B}{\Gamma \Rightarrow A\backslash B}\,\text{R<}$$

For any given lexicon ℓ, a sequence of words $w_1, \ldots, w_n$ is a sentence if and only if $\ell(w_1), \ldots, \ell(w_n) \Rightarrow \text{S}$.

```
inductive LambekGrammar
  :: "'a category list ⇒ 'a category ⇒ bool" (infix "⊢L" 60) where
  identity : "[A] ⊢L A"
  | Cut : "⟦Γ@[A]@Δ ⊢L B; Σ ⊢L A⟧ ⟹ Γ@Σ@Δ ⊢L B"
  | LR : "⟦Γ@[A]@Δ ⊢L B; Σ ⊢L C⟧ ⟹ Γ@Σ@[C→A]@Δ ⊢L B"
  | LL : "⟦Γ@[A]@Δ ⊢L B; Σ ⊢L C⟧ ⟹ Γ@[A←C]@Σ@Δ ⊢L B"
  | RR : "Γ@[A] ⊢L B ⟹ Γ ⊢L B ← A"
  | RL : "[A]@Γ ⊢L B ⟹ Γ ⊢L A → B"
theorem
  fixes f :: "'a ⇒ 'b category" and S :: "'b category"
  assumes "LCCG = {W . map f W ⊢CCG S}" and "LL = {W . map f W ⊢L S}"
  shows "LCCG ⊆ LL" apply (simp add: assms) using CCG_is_Lambek by blast
```

Definition (Continuation Passing Style Transformation (Plotkin, 1975))

In lambda calculus, we transform lambda term as follows.

$$
\begin{aligned}
[x] &\equiv \lambda k.kx \\
[\lambda x.M] &\equiv \lambda k.k(\lambda x.[M]) \\
[MN] &\equiv \lambda k.[M](\lambda m.[N](\lambda n.mnk))
\end{aligned}
$$

Definition (Type of CPS)

For any given type A, We also need to transform the type of lambda-term as follows.

$$[X] \equiv (\langle X \rangle \to A) \to A \qquad \langle X \rangle \equiv X \qquad \langle X \to Y \rangle \equiv \langle X \rangle \to [Y]$$

Definition (Type of restricted CPS)

For any given type A, We transform the type of lambda-term as follows.

$$[X] \equiv (\langle X \rangle \to A) \to A \qquad \langle X \rangle \equiv X \qquad \langle X \to Y \rangle \equiv X \to [Y]$$

Definition (Type of CPS)

For any given type A, We also need to transform the type of lambda-term as follows.

$$[X] \equiv (\langle X \rangle \to A) \to A \qquad \langle X \rangle \equiv X \qquad \langle X \to Y \rangle \equiv \langle X \rangle \to [Y]$$

Definition (Type of restricted CPS)

For any given type A, We transform the type of lambda-term as follows.

$$[X] \equiv (\langle X \rangle \to A) \to A \qquad \langle X \rangle \equiv X \qquad \langle X \to Y \rangle \equiv X \to [Y]$$

Restricted Continuation Passing Style Transformation

Definition

Definition (Type of restricted CPS)

For any given category A, We transform the type of lambda-term as follows.

$$[X] \rightsquigarrow A/(\langle X\rangle\backslash A) \qquad [X] \rightsquigarrow (A/\langle X\rangle)\backslash A$$

$$\langle X\rangle \rightsquigarrow X \qquad \langle Y/X\rangle \rightsquigarrow [Y]/X \qquad \langle Y/X\rangle \rightsquigarrow [Y]/X$$

Theorem (Restricted CPS is derivable in Lambek Calculus)

We show $X \Rightarrow [X]$ *and* $X \Rightarrow \langle X\rangle$ *in Isabelle/HOL.*

```
inductive rCPS :: "'a category ⇒ 'a category ⇒ 'a category ⇒ bool"
    and rCPS' :: "'a category ⇒ 'a category ⇒ 'a category ⇒ bool"
  where
  "rCPS' A X Y ⟹ rCPS A X ((A←Y)→A)"
  | "rCPS' A X Y ⟹ rCPS A X (A←(Y→A))"
  | "rCPS A X2 Y2 ⟹ rCPS' A ((^X1)→X2) ((^X1)→Y2)"
  | "rCPS A X2 Y2 ⟹ rCPS' A (X2←(^X1)) (Y2←(^X1))"
  | "rCPS' A (^X) (^X)"
theorem rCPS_transformation:
  fixes A X :: "'a category"
  shows "⋀Y. rCPS' A X Y ⟹ [X]⊢L Y"
   and "⋀Y. rCPS A X Y ⟹ [X]⊢L Y"
proof ... qed
```

Q Combinator and D Combinator

Definition (Q combinator and D Combinator)

We use the sequent-style system. This system is *intuitionistic* and it has *no structural rules*.

$$\frac{}{A, A\backslash B \Rightarrow B}<\qquad \frac{}{B/A, A \Rightarrow B}>\qquad \frac{\Gamma \Rightarrow A \qquad \Delta, A, \Sigma \Rightarrow B}{\Delta, \Gamma, \Sigma \Rightarrow B}\text{Cut}$$

$$\frac{}{A/B, B/C \Rightarrow A/C}>\text{B}\qquad \frac{}{A/B \Rightarrow (A/C)/(B/C)}>\text{D}\qquad \frac{}{B/C \Rightarrow (A/B)\backslash(A/C)}>\text{Q}$$

$$\frac{}{A\backslash B, B\backslash C \Rightarrow A\backslash C}<\text{B}\qquad \frac{}{B\backslash C \Rightarrow (A\backslash B)\backslash(A\backslash C)}<\text{D}\qquad \frac{}{A\backslash B \Rightarrow (A\backslash C)/(B\backslash C)}<\text{Q}$$

$$\frac{}{A \Rightarrow X/(A\backslash X)}<\text{T}\qquad \frac{}{A \Rightarrow (X/A)\backslash X}>\text{T}$$

For any given lexicon ℓ, a sequence of words $w_1, \ldots, w_n$ is a sentence if and only if $\ell(w_1), \ldots, \ell(w_n) \Rightarrow \text{S}$.

The *incremental parsing* is a strategy for parsing a sentence from a head.

- We obtain a parsing tree even if the utterance/ reading is in progress.
- We predict what category comes next.

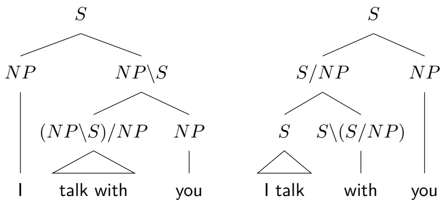

Figure: Variations of Parsing Tree

Q Combinator and D Combinator

Incremental Parsing

The *incremental parsing* is a strategy for parsing a sentence from a head.

- We obtain a parsing tree even if the utterance/ reading is in progress.
- We predict what category comes next.

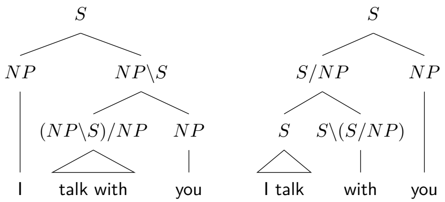

Figure: Variations of Parsing Tree

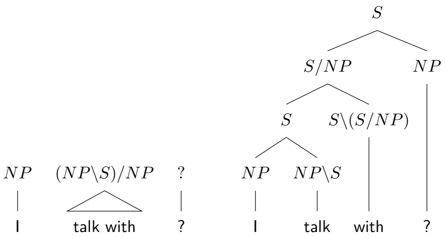

Figure: Incomplete sentence

Conjecture (Taniguchi and Tojo, LENLS2021)

For any given derivation of Categorial Grammar, there exists an incremental parsing of Q Combinatory Categorial Grammar.

Table: Number of Accepted Sentences

Grammar \Number of Words	2	3	4	5	6
Non-incremental CG	2	8	40	224	1344
Incremental CG	2	4	8	16	32
Incremental CCG	2	8	32	128	512
Incremental QCCG	2	8	40	224	1344

The number of derivations of the non-incremental CG is $2^{n-1}\mathcal{C}_{n-1} = \frac{2^{n-1}}{n}\binom{2n-2}{n-1}$, where n is the number of words and $\mathcal{C}_{n-1}$ is the Catalan number of n leaves.

Conjecture (Taniguchi and Tojo, LENLS2021)

For any given derivation of Categorial Grammar, there exists an incremental parsing of Q Combinatory Categorial Grammar.

Future Work

We would like to proof this conjecture in Isabelle.

Conclusion

- We have given the formal proof in Isabelle/HOL as follows.
  - The restricted CPS transformation of Lambek Calculus.
  - The inclusions of languages generated by CG, CCG, and Lambek.
- We expect implementing a new parser by Sledgehammer in Isabelle/HOL.
- Proof of the existence of the incremental paring is our immediate future task.

Bonus: Editors for Isabelle

Isabelle/PIDE is Language Server Protocol.

- jEdit†: Isabelle/jEdit
- Visual Studio Code†: Isabelle/VSCode
- Vim: coc-isabelle (https://github.com/ThreeFx/coc-isabelle)
- Emacs: isabelle-emacs (https://github.com/m-fleury/isabelle-emacs)
- jEdit with Vim: vimulator (https://github.com/thoughtbot/vimulator)

† jEdit and Visual Studio Code are official projects of Isabelle developers.