The Grammar of Grammars

K. Kogkalidis

Logic & Language 2020

Recap: Formal Grammars

Formal Grammars

A formal grammar $\mathcal{G}$ is a tuple $\mathcal{G} = \langle V, \Sigma, R, S \rangle$, where

- V the *vocabulary*, a set of symbols
- Σ the set of *terminal* symbols, $\Sigma \subset V$
- R the set of *production rules*, $R \subset V^* \times V^*$
- S the *initial symbol*, $S \in V - \Sigma$

Rules

A rule $r \in R$ is usually written as $\alpha \to \beta$, where α, β *strings* of V, i.e. $\alpha, \beta \in V^*$.
Allowing only specific forms of rules R leads to a hierarchy of formal grammars, each with their own expressivity and complexity.

Language

The set of *words* (strings) $\mathcal{L}_{\mathcal{G}} \in \Sigma^*$ that can be generated by $\mathcal{G}$.

Chomsky Hierarchy

type	grammar	automaton	rule form
3	regular	finite state machine	$A \to a$; $A \to aB$
2	context-free	pushdown automaton	$A \to \gamma$
1	context-sensitive	linear bounded automaton	$\alpha A \beta \to \alpha \gamma \beta$
0	recursively enumerable	Turing machine	$\alpha \to \beta$

A, B: non-terminals, a: terminal, α, β, γ: strings of V

Type-3 $\subset$ Type-2 $\subset$ Type-1 $\subset$ Type-0

Natural Language

- R aligned with speech, phonology, morphology
- CF captures most syntactic patterns (but not all!)
- CS too expressive and complex to be of real use

$\rightsquigarrow$ need a better charting between CF and CS

Pumping Lemma for CFL

Let $\mathcal{G} = \langle V, \Sigma, R, S \rangle$ a CFG generating an infinite language $\mathcal{L}_{\mathcal{G}}$.

$$
\begin{aligned}
&\exists\ k \in \mathbb{N} : \\
&\quad \forall\ w \in \mathcal{L}_{\mathcal{G}} \wedge |w| \geq k : \\
&\quad\quad \exists\ x, y, z, v_1, v_2 \in \Sigma^* : \\
&\quad\quad\quad \bigwedge \Big\{ w = x v_1 y v_2 z,\ |v_1 v_2| \geq 1,\ |v_1 y v_2| \leq k, \\
&\quad\quad\quad\quad \forall\ i \in \mathbb{N} : \{ x v_1^i y v_2^i z \in \mathcal{L}_{\mathcal{G}} \} \Big\}
\end{aligned}
$$

Pumping Lemma for CFL

Let $\mathcal{G} = \langle V, \Sigma, R, S \rangle$ a CFG generating an infinite language $\mathcal{L}_\mathcal{G}$.

$$
\begin{aligned}
&\exists\ k \in \mathbb{N} : \\
&\quad \forall\ w \in \mathcal{L}_\mathcal{G} \wedge |w| \geq k : \\
&\quad\quad \exists\ x, y, z, v_1, v_2 \in \Sigma^* : \\
&\quad\quad\quad \bigwedge \Big\{ w = x v_1 y v_2 z,\ |v_1 v_2| \geq 1,\ |v_1 y v_2| \leq k, \\
&\quad\quad\quad\quad \forall\ i \in \mathbb{N} : \{ x v_1^i y v_2^i z \in \mathcal{L}_\mathcal{G} \} \Big\}
\end{aligned}
$$

Example

The copy language $\mathcal{L} = \{ww \mid w \in \{a, b\}^*\}$ is not context-free, but similar constructions occur in natural language (crossing dependencies):

. . . *dat Wim Jan Marie de kinderen zag helpen leren zwemmen* . . .
"...that Wim saw Jan help Marie teach the kids how to swim . . ."

The landscape beyond CFL

The class of mildly context-sensitive languages:

- contains context-free languages
- capture a finite number of cross-serial dependencies, i.e languages of the form: $\mathcal{L} = \{w^k | \ w \in \Sigma^*\}$ for some k
- maintains polynomial parsing time (CFGs have $\mathcal{O}(n^3)$)
- is characterized by constant growth: word length increase is linear-bound

Among the animals in the zoo: TAG, CCG, HG, LIG, (well-nested) k-MCFG, (simple) RCG, MG, ...

Abstract Categorial Grammars

Abstract Categorial Grammars model the landscape of formal grammars as a morphism between two $\mathrm{ILL}_{\multimap}$ logics:

$$\begin{array}{ccc} \mathrm{ILL}^{A}_{\multimap} & \xrightarrow[\textit{Homomorphism}]{\quad h \quad} & \mathrm{IL}^{A'}_{\multimap} \\ \text{Source} & & \text{Target} \end{array}$$

- source
  logic describing the abstract function-argument structure of the language (tectogrammar)
- target
  logic describing the concrete surface materialization of the language: strings, trees, etc (phenogrammar)

Abstract Categorial Grammars

Vocabulary

A vocabulary Σ is a "higher-order linear signature" $\Sigma = \langle \mathcal{A}, C, \tau \rangle$, where:

- $\mathcal{A}$ a set of atomic types ($\mathcal{T}_{\mathcal{A}}$ the type universe)
- C a set of constants (Λ_Σ the set of well-formed λ-terms)
- τ a mapping $C \to \mathcal{T}_{\mathcal{A}}$

Lexicon

A lexicon $\mathfrak{L}$ is a mapping $\Sigma_1 \to \Sigma_2$ consisting of $\langle \eta, \theta \rangle$, where

- η a mapping $\mathcal{A}_1 \to \mathcal{T}_{\mathcal{A}_2}$, deriving the homomorphic extension $\hat{\eta} : \mathcal{T}_{\mathcal{A}_1} \to \mathcal{T}_{\mathcal{A}_2}$
- θ a mapping $C_1 \to \Lambda_{\Sigma_2}$, deriving the homomorphic extension $\hat{\theta} : \Lambda_{\Sigma_1} \to \Lambda_{\Sigma_2}$

such that $\vdash \theta(c) : \hat{\eta}(\tau(c))$, i.e. θ respects typing

Abstract Categorial Grammars

ACG

An abstract categorial grammar is a tuple $\langle \Sigma_1, \Sigma_2, \mathfrak{L}, s \rangle$, where:

- Σ_1 the abstract vocabulary
- Σ_2 the object language
- $\mathfrak{L}$ the map $\Sigma_1 \rightarrow \Sigma_2$
- s the initial or distinguished type, $s \in \mathcal{T}_{\mathcal{A}_1}$

From the vocabularies we obtain languages $\mathcal{L}_1, \mathcal{L}_2$:

- $\mathcal{L}_1$ the abstract language

$$\mathcal{L}_1 = \{t \in \Lambda_{\Sigma_1} |\ t \text{ an inhabitant of } s\}$$

- $\mathcal{L}_2$ the object language

$$\mathcal{L}_2 = \{t \in \Lambda_{\Sigma_2} \mid \exists\ u \in \mathcal{L}_1 : t \text{ the } \hat{\theta}\text{-image of } u\}$$

Example: ACG for the Dyck Language

Dyck Language

The language of well-bracketed parentheses, captured by the CFG:

$$S \rightarrow SS_{(\mathrm{R}_1)} \mid [S]_{(\mathrm{R}_2)} \mid \epsilon_{(\mathrm{R}_3)}$$

Source Signature $\Sigma_1 = \langle \mathcal{A}_1, C_1, \tau_1 \rangle$

$$\mathcal{A}_1 = \{S\} \quad C_1 = \{\mathrm{R}_1,\ \mathrm{R}_2,\ \mathrm{R}_3\} \quad \tau_1 = \{\mathrm{R}_1 \mapsto S \multimap S \multimap S,\ \mathrm{R}_2 \mapsto S \multimap S,\ \mathrm{R}_3 \mapsto S\}$$

Target Signature $\Sigma_2 = \langle \mathcal{A}_2, C_2, \tau_2 \rangle$

$$\mathcal{A}_2 = \{*\} \quad C_2 = \{\texttt{[},\ \texttt{]}\} \quad \tau_2 = \{\texttt{[} \mapsto * \multimap *,\ \texttt{]} \mapsto * \multimap *\}$$

where $*$ a primitive type s.t. $\texttt{str} = * \multimap *$

$\cdot : \texttt{str} \multimap \texttt{str} \multimap \texttt{str} = \lambda f.\lambda g.\lambda i.f(g\ \ i)$

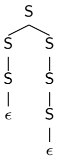
Translation $\mathfrak{L} = \langle \eta, \theta \rangle$

$$\eta = \{S \mapsto \texttt{str}\} \quad \theta = \{\mathrm{R}_1 \mapsto \lambda x \lambda y.x \cdot y,\ \mathrm{R}_2 \mapsto \lambda x.\texttt{[} \cdot x \cdot \texttt{]},\ \mathrm{R}_3 \mapsto \lambda x.x\}$$

Example: ACG for the Dyck Language

Parsing

$$\texttt{[] [[]]} \in ?\mathcal{L}_2 \quad \Leftrightarrow \quad \exists ? u \in \mathcal{L}_1 . \hat{\theta}(u) = \texttt{[] [[]]}$$



$$\dfrac{\dfrac{\mathrm{R}_1 : S \multimap S \multimap S \qquad \dfrac{\mathrm{R}_2 : S \multimap S \quad \mathrm{R}_3 : S}{\mathrm{R}_2\mathrm{R}_3 : S}\multimap E}{\mathrm{R}_1(\mathrm{R}_2\mathrm{R}_3) : S \multimap S}\multimap E \qquad \dfrac{\mathrm{R}_2 : S \multimap S \qquad \dfrac{\mathrm{R}_2 : S \multimap S \quad \mathrm{R}_3 : S}{\mathrm{R}_2\mathrm{R}_3 : S}\multimap E}{\mathrm{R}_2(\mathrm{R}_2\mathrm{R}_3) : S}\multimap E}{\vdash (\mathrm{R}_1(\mathrm{R}_2\mathrm{R}_3))(\mathrm{R}_2(\mathrm{R}_2\mathrm{R}_3)) : S}\multimap E$$

$$\begin{aligned} u &= (\mathrm{R}_1(\mathrm{R}_2\mathrm{R}_3))\ (\mathrm{R}_2(\mathrm{R}_2\mathrm{R}_3)) \\ \hat{\theta}(u) &= (\theta(\mathrm{R}_1)\ (\theta(\mathrm{R}_2)\ \theta(\mathrm{R}_3)))\ (\theta(\mathrm{R}_2)\ (\theta(\mathrm{R}_2)\ \theta(\mathrm{R}_3))) \\ &= \ldots \\ &\overset{\beta}{\rightsquigarrow} \texttt{[] [[]]} \end{aligned}$$

ACG Hierarchy

The order $\mathcal{O}$ of a type T is $\mathcal{O}(T) = \begin{cases} 0 & T \in \mathcal{A} \\ \max\left(\mathcal{O}(A) + 1, \mathcal{O}(B)\right) & T = A \multimap B \end{cases}$

ACG Hierarchy

The order $\mathcal{O}$ of a type T is $\mathcal{O}(T) = \begin{cases} 0 & T \in \mathcal{A} \\ \max\left(\mathcal{O}(A)+1, \mathcal{O}(B)\right) & T = A \multimap B \end{cases}$

ACG measures of complexity:

- Complexity of abstract signature: $\mathcal{C}(\Sigma_1) = \max_{c \in C_1}\{\mathcal{O}\left(\tau\left(c\right)\right)\}$
- Complexity of interpretation: $\mathcal{C}(\mathfrak{L}) = \max_{\alpha \in \mathcal{A}_1}\{\mathcal{O}\left(\eta\left(\alpha\right)\right)\}$

The type of an ACG is the tuple $(\mathcal{C}(\Sigma_1), \mathcal{C}(\mathfrak{L}))$.

ACG Hierarchy

The order $\mathcal{O}$ of a type T is $\mathcal{O}(T) = \begin{cases} 0 & T \in \mathcal{A} \\ \max\left(\mathcal{O}(A)+1, \mathcal{O}(B)\right) & T = A \multimap B \end{cases}$

ACG measures of complexity:

- Complexity of abstract signature: $\mathcal{C}(\Sigma_1) = \max_{c \in C_1}\{\mathcal{O}\left(\tau\left(c\right)\right)\}$
- Complexity of interpretation: $\mathcal{C}(\mathfrak{L}) = \max_{\alpha \in \mathcal{A}_1}\{\mathcal{O}\left(\eta\left(\alpha\right)\right)\}$

The type of an ACG is the tuple $(\mathcal{C}(\Sigma_1), \mathcal{C}(\mathfrak{L}))$.

Embedding the Chomsky Hierarchy

ACG Type	$\mathcal{L}_2$ Class
(2, 1)	regular
(2, 2)	context-free
(2, 3)	well-nested mildly context-sensitive
(2, n $\geq$ 4)	mildly context-sensitive

Example: m-CFGs in ACG

Multiple context-free grammars operate on tuples of strings; tuples can be encoded as higher-order λ-terms:

$$\langle a_1, \ldots, a_n \rangle \rightsquigarrow \lambda t.(t\ a_1 \ldots a_n) : \mathtt{str}^{(n)} \equiv (\underbrace{\mathtt{str} \multimap \ldots \multimap \mathtt{str}}_{n+1}) \multimap \mathtt{str}$$

The language $\{a^n b^n c^n d^n \mid n > 0\}$ is generated by the 2-CFG:

$$S(xy) \to A(x,y)_{(\mathrm{R}_1)} \quad A(\mathtt{a}x\mathtt{b}, \mathtt{c}y\mathtt{d}) \to A(x,y)_{(\mathrm{R}_2)} \quad A(\epsilon,\epsilon) \to \epsilon_{(\mathrm{R}_3)}$$

ACG encoding

$$\Sigma_1 = \{A, S\} \qquad \tau_1 = \{\mathrm{R}_1 \mapsto A \multimap S,\ \mathrm{R}_2 \mapsto A \multimap A,\ \mathrm{R}_3 \mapsto A\}$$

$$\Sigma_2 = \{*\}, \qquad \tau_2 = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d} \mapsto \mathtt{str}\}$$

$$\eta = \{S \mapsto \mathtt{str}, A \mapsto \mathtt{str}^{(2)}\}$$

$$\begin{aligned}\theta = \{&\mathrm{R}_1 \mapsto \lambda\rho.\,(\rho\ \lambda xy.\,(x+y)) : \mathtt{str}^{(2)} \multimap \mathtt{str},\\ &\mathrm{R}_2 \mapsto \lambda\rho q.\,(\rho\ \lambda xy.\,(q\ \ (a+x+b)\ \ (c+y+d))) : \mathtt{str}^{(2)} \multimap \mathtt{str}^{(2)},\\ &\mathrm{R}_3 \mapsto \lambda t.(t\ \epsilon\ \epsilon) : \mathtt{str}^{(2)}\}\end{aligned}$$

Intermezzo

Convincing ourselves about θ

$$(\text{“\textit{logic}”}, \text{“\textit{language}”}) \rightsquigarrow \lambda t.(t\ \text{“\textit{logic}”}\ \ \text{“\textit{language}”})$$

$$\theta(\mathrm{R}_1) = \lambda\rho.\,(\rho\ \lambda xy.\,(x+y))$$

Intermezzo

Convincing ourselves about θ

$$(\text{“}\textit{logic}\text{”}, \text{“}\textit{language}\text{”}) \rightsquigarrow \lambda t.(t\ \text{“}\textit{logic}\text{”}\ \ \text{“}\textit{language}\text{”})$$

$$\theta(\mathrm{R}_1) = \lambda\rho.\,(\rho\ \lambda xy.\,(x + y))$$

$$\theta(\mathrm{R}_1)\ \ (\text{“}\textit{logic}\text{”}, \text{“}\textit{language}\text{”}) = \lambda\rho.\,(\rho\ \lambda xy.\,(x + y))\ \ \lambda t.(t\ \text{“}\textit{logic}\text{”}\ \ \text{“}\textit{language}\text{”})$$

Intermezzo

Convincing ourselves about θ

$$(\text{“}\mathit{logic}\text{”}, \text{“}\mathit{language}\text{”}) \rightsquigarrow \lambda t.(t\ \text{“}\mathit{logic}\text{”}\ \ \text{“}\mathit{language}\text{”})$$

$$\theta(\mathrm{R}_1) = \lambda\rho.\,(\rho\ \lambda xy.\,(x + y))$$

$$\theta(\mathrm{R}_1)\ \ (\text{“}\mathit{logic}\text{”}, \text{“}\mathit{language}\text{”}) = \lambda\rho.\,(\rho\ \lambda xy.\,(x + y))\ \ \lambda t.(t\ \ \text{“}\mathit{logic}\text{”}\ \ \text{“}\mathit{language}\text{”})$$

$$\overset{\beta}{\rightsquigarrow} \lambda t.(t\ \ \text{“}\mathit{logic}\text{”}\ \ \text{“}\mathit{language}\text{”})\ \lambda xy.(x + y)$$

Intermezzo

Convincing ourselves about θ

$$(\text{“}\textit{logic}\text{”}, \text{“}\textit{language}\text{”}) \rightsquigarrow \lambda t.(t \ \text{“}\textit{logic}\text{”} \ \text{“}\textit{language}\text{”})$$
$$\theta(\mathrm{R}_1) = \lambda\rho.\,(\rho\ \lambda xy.\,(x+y))$$

$$\begin{aligned}
\theta(\mathrm{R}_1)\ (\text{“}\textit{logic}\text{”}, \text{“}\textit{language}\text{”}) &= \lambda\rho.\,(\rho\ \lambda xy.\,(x+y))\ \ \lambda t.(t\ \text{“}\textit{logic}\text{”}\ \ \text{“}\textit{language}\text{”})\\
&\overset{\beta}{\rightsquigarrow} \lambda t.(t\ \text{“}\textit{logic}\text{”}\ \ \text{“}\textit{language}\text{”})\ \lambda xy.(x+y)\\
&\overset{\beta}{\rightsquigarrow} \lambda xy.(x+y)\ \ \text{“}\textit{logic}\text{”}\ \ \text{“}\textit{language}\text{”}
\end{aligned}$$

Intermezzo

Convincing ourselves about θ

$$(\text{“}\textit{logic}\text{”}, \text{“}\textit{language}\text{”}) \rightsquigarrow \lambda t.(t\ \text{“}\textit{logic}\text{”}\ \ \text{“}\textit{language}\text{”})$$

$$\theta(\mathrm{R}_1) = \lambda\rho.\,(\rho\ \lambda xy.\,(x+y))$$

$$\begin{aligned}
\theta(\mathrm{R}_1)\ \ (\text{“}\textit{logic}\text{”}, \text{“}\textit{language}\text{”}) &= \lambda\rho.\,(\rho\ \lambda xy.\,(x+y))\ \ \lambda t.(t\ \text{“}\textit{logic}\text{”}\ \ \text{“}\textit{language}\text{”}) \\
&\overset{\beta}{\rightsquigarrow} \lambda t.(t\ \text{“}\textit{logic}\text{”}\ \ \text{“}\textit{language}\text{”})\ \lambda xy.(x+y) \\
&\overset{\beta}{\rightsquigarrow} \lambda xy.(x+y)\ \ \text{“}\textit{logic}\text{”}\ \ \text{“}\textit{language}\text{”} \\
&\overset{\beta}{\rightsquigarrow} \lambda y.(\text{“}\textit{logic}\text{”} + y)\ \ \text{“}\textit{language}\text{”}
\end{aligned}$$

Intermezzo

Convincing ourselves about θ

$$(\text{“}\mathit{logic}\text{”}, \text{“}\mathit{language}\text{”}) \rightsquigarrow \lambda t.(t\ \text{“}\mathit{logic}\text{”}\ \ \text{“}\mathit{language}\text{”})$$

$$\theta(\mathrm{R}_1) = \lambda \rho.\,(\rho\ \lambda xy.\,(x+y))$$

$$\begin{aligned}
\theta(\mathrm{R}_1)\ \ (\text{“}\mathit{logic}\text{”}, \text{“}\mathit{language}\text{”}) &= \lambda \rho.\,(\rho\ \lambda xy.\,(x+y))\ \ \lambda t.(t\ \text{“}\mathit{logic}\text{”}\ \ \text{“}\mathit{language}\text{”}) \\
&\overset{\beta}{\rightsquigarrow} \lambda t.(t\ \text{“}\mathit{logic}\text{”}\ \ \text{“}\mathit{language}\text{”})\ \lambda xy.(x+y) \\
&\overset{\beta}{\rightsquigarrow} \lambda xy.(x+y)\ \ \text{“}\mathit{logic}\text{”}\ \ \text{“}\mathit{language}\text{”} \\
&\overset{\beta}{\rightsquigarrow} \lambda y.(\text{“}\mathit{logic}\text{”} + y)\ \ \text{“}\mathit{language}\text{”} \\
&\overset{\beta}{\rightsquigarrow} \text{“}\mathit{logic}\text{”} + \text{“}\mathit{language}\text{”}
\end{aligned}$$

Example: m-CFGs in ACG (cont)

Parsing

$$\texttt{aabbccdd} \in ? \mathcal{L}_2 \quad \Leftrightarrow \quad \exists ? u \in \mathcal{L}_1 . \hat{\theta}(u) = \texttt{aabbccdd}$$

$$\dfrac{\mathrm{R}_1 : A \multimap S \qquad \dfrac{\mathrm{R}_2 : A \multimap A \qquad \dfrac{\mathrm{R}_2 : A \multimap A \quad \mathrm{R}_3 : A}{\mathrm{R}_2\mathrm{R}_3 : A} \multimap E}{\mathrm{R}_2\,(\mathrm{R}_2\mathrm{R}_3) : A} \multimap E}{\vdash \mathrm{R}_1\,(\mathrm{R}_2\,(\mathrm{R}_2\mathrm{R}_3)) : S} \multimap E$$

$$\begin{aligned}
\theta(\mathrm{R}_2)\theta(\mathrm{R}_3) &= \lambda pq.(p\ \lambda xy.(q\ (\texttt{a} + x + \texttt{b})\ (\texttt{c} + y + \texttt{d})))\ \lambda t.(t\ \epsilon\ \epsilon) \\
&\overset{\beta}{\rightsquigarrow} \lambda q.(\lambda t.(t\ \epsilon\ \epsilon)\ \lambda xy.(q\ (\texttt{a} + x + \texttt{b})\ (\texttt{c} + y + \texttt{d}))) \\
&\overset{\beta}{\rightsquigarrow} \lambda q.(\lambda xy.(q\ (\texttt{a} + x + \texttt{b})\ (\texttt{c} + y + \texttt{d}))\ \epsilon\ \epsilon) \\
&\overset{\beta}{\rightsquigarrow} \lambda q.(q\ (\texttt{a} + \epsilon + \texttt{b})\ (\texttt{c} + \epsilon + \texttt{d})) \overset{\beta}{\rightsquigarrow} \lambda q.(q\ \texttt{ab}\ \texttt{cd})
\end{aligned}$$

Example: m-CFGs in ACG (cont)

Parsing

$$\mathtt{aabbccdd} \in ?\mathcal{L}_2 \quad \Leftrightarrow \quad \exists ?u \in \mathcal{L}_1.\hat{\theta}(u) = \mathtt{aabbccdd}$$

$$\dfrac{\mathrm{R}_1 : A \multimap S \qquad \dfrac{\mathrm{R}_2 : A \multimap A \qquad \dfrac{\mathrm{R}_2 : A \multimap A \quad \mathrm{R}_3 : A}{\mathrm{R}_2\mathrm{R}_3 : A} \multimap E}{\mathrm{R}_2\,(\mathrm{R}_2\mathrm{R}_3) : A} \multimap E}{\vdash \mathrm{R}_1\,(\mathrm{R}_2\,(\mathrm{R}_2\mathrm{R}_3)) : S} \multimap E$$

$$\begin{aligned}
\theta(\mathrm{R}_2)\theta(\mathrm{R}_3) &= \lambda pq.(p\ \lambda xy.(q\ (\mathtt{a}+x+\mathtt{b})\ (\mathtt{c}+y+\mathtt{d})))\ \lambda t.(t\ \epsilon\ \epsilon) \\
&\overset{\beta}{\rightsquigarrow} \lambda q.(\lambda t.(t\ \epsilon\ \epsilon)\ \lambda xy.(q\ (\mathtt{a}+x+\mathtt{b})\ (\mathtt{c}+y+\mathtt{d}))) \\
&\overset{\beta}{\rightsquigarrow} \lambda q.(\lambda xy.(q\ (\mathtt{a}+x+\mathtt{b})\ (\mathtt{c}+y+\mathtt{d}))\ \epsilon\ \epsilon) \\
&\overset{\beta}{\rightsquigarrow} \lambda q.(q\ (\mathtt{a}+\epsilon+\mathtt{b})\ (\mathtt{c}+\epsilon+\mathtt{d})) \overset{\beta}{\rightsquigarrow} \lambda q.(q\ \mathtt{ab}\ \mathtt{cd}) \\
\theta(\mathrm{R}_2)(\theta(\mathrm{R}_2)\theta(\mathrm{R}_3)) &= \lambda fg.(f\ \lambda xy.(g\ \mathtt{a}+x+\mathtt{b})\ (\mathtt{c}+y+\mathtt{d})))\ \lambda q.(q\ \mathtt{ab}\ \mathtt{cd}) \\
&\overset{\beta}{\rightsquigarrow} \lambda g.(\lambda q.(q\ \mathtt{ab}\ \mathtt{cd})\ \lambda xy.(g\ (\mathtt{a}+x+\mathtt{b})\ (\mathtt{c}+y+\mathtt{d}))) \\
&\overset{\beta}{\rightsquigarrow} \lambda g.(\lambda xy.(g\ (\mathtt{a}+x+\mathtt{b})\ (\mathtt{c}+y+\mathtt{d}))\ \mathtt{ab}\ \mathtt{cd}) \\
&\overset{\beta}{\rightsquigarrow} \lambda g.(g\ (\mathtt{a}+\mathtt{ab}+\mathtt{b})\ (\mathtt{c}+\mathtt{cd}+\mathtt{d})) \overset{\beta}{\rightsquigarrow} \lambda g.(g\ \mathtt{aabb}\ \mathtt{ccdd})
\end{aligned}$$