Underlying principles and recurring ideas of formal grammars

Alexander Okhotin
St. Petersburg State University, Russia

the 10th of April, A. D. 2018
Part I

Introduction
Formal grammars: a model of syntax

- Substrings with known properties, one after another.
Formal grammars: a model of syntax

- Substrings with known properties, one after another.

“A noun phrase, followed by a verb phrase, is a sentence” ($S \rightarrow NP \ VP$).
Formal grammars: a model of syntax

- Substrings with known properties, one after another.

“A noun phrase, followed by a verb phrase, is a sentence” ($S \rightarrow NP \ VP$).

- Immediate constituent analysis, later Chomsky.
Formal grammars: a model of syntax

- Substrings with known properties, one after another.

“\(A \text{ noun phrase, followed by a verb phrase, is a sentence}\)” \((S \rightarrow NP \ VP)\).

- *Immediate constituent analysis*, later Chomsky.
- Formal grammars: now a classical subject.
Formal grammars: a model of syntax

- Substrings with known properties, one after another.

  “A noun phrase, followed by a verb phrase, is a sentence” ($S \rightarrow NP \ VP$).

- Immediate constituent analysis, later Chomsky.
- Formal grammars: now a classical subject.
- Ongoing research: new models, new ideas.
Formal grammars: a model of syntax

- Substrings with known properties, one after another.

“A noun phrase, followed by a verb phrase, is a sentence” \((S \rightarrow NP \ VP)\).

- Immediate constituent analysis, later Chomsky.
- Formal grammars: now a classical subject.
- Ongoing research: new models, new ideas.
- Classroom presentation rooted in the 1960s.
Surveying the area

- Present useful models *together*. 
Surveying the area

- Present useful models *together*.
- Look for common definitions and notation.
Surveying the area

- Present useful models *together*.
- Look for common definitions and notation.
- Similar results for different grammar families.
Surveying the area

- Present useful models together.
- Look for common definitions and notation.
- Similar results for different grammar families.
- Present these recurring ideas together.
Surveying the area

- Present useful models *together*.
- Look for common definitions and notation.
- Similar results for different grammar families.
- Present these recurring ideas *together*.
- For each idea, limits of its applicability.
Old systematics

- Chomsky (1956): grammars as rewriting $A \rightarrow BC$.
Old systematics

- Chomsky (1956): grammars as rewriting $A \rightarrow BC$.
- Chomsky (1957): generalized rewriting $AB \rightarrow CD$. 

The Chomsky hierarchy

0. Recursively enumerable sets.
1. NSPACE($n^c$) (presumed "context-sensitive grammars")
2. Grammars (hence called "context-free")
3. Regular languages: DSPACE(const).

Ancient complexity theory, important in its time.
Old systematics

- Chomsky (1956): grammars as rewriting $A \rightarrow BC$.
- Chomsky (1957): generalized rewriting $AB \rightarrow CD$.

The Chomsky hierarchy

- Recursively enumerable sets.
- NSPACE($n$) (presumed "context-sensitive grammars")
- Grammars (hence called "context-free")
- Regular languages: DSPACE(const).

Ancient complexity theory, important in its time.

$AB \rightarrow CD$: nothing else but computation.
Old systematics

- Chomsky (1956): grammars as rewriting $A \rightarrow BC$.
- Chomsky (1957): generalized rewriting $AB \rightarrow CD$.

The Chomsky hierarchy

0. Recursively enumerable sets.
Old systematics

- Chomsky (1956): grammars as rewriting $A \rightarrow BC$.
- Chomsky (1957): generalized rewriting $AB \rightarrow CD$.

The Chomsky hierarchy

- $\mathbb{0}$ Recursively enumerable sets.
- $\mathbb{1}$ NSPACE($n$)
Old systematics

- Chomsky (1956): grammars as rewriting $A \rightarrow BC$.
- Chomsky (1957): generalized rewriting $AB \rightarrow CD$.

The Chomsky hierarchy

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Recursively enumerable sets.</td>
</tr>
<tr>
<td>1</td>
<td>NSPACE($n$) (presumed “context-sensitive grammars”)</td>
</tr>
</tbody>
</table>
Old systematics

- Chomsky (1956): grammars as rewriting $A \rightarrow BC$.
- Chomsky (1957): generalized rewriting $AB \rightarrow CD$.

The Chomsky hierarchy

- **0**: Recursively enumerable sets.
- **1**: NSPACE$(n)$ (presumed “context-sensitive grammars”)
- **2**: Grammars (hence called “context-free”)

Ancient complexity theory, important in its time.
Old systematics

- Chomsky (1956): grammars as rewriting $A \rightarrow BC$.
- Chomsky (1957): generalized rewriting $AB \rightarrow CD$.

The Chomsky hierarchy

0. Recursively enumerable sets.
1. $\text{NSPACE}(n)$ (presumed “context-sensitive grammars”)
2. Grammars (hence called “context-free”)
3. Regular languages: $\text{DSPACE}(\text{const})$. 
Old systematics

- Chomsky (1956): grammars as rewriting $A \rightarrow BC$.
- Chomsky (1957): generalized rewriting $AB \rightarrow CD$.

The Chomsky hierarchy

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Recursively enumerable sets.</td>
</tr>
<tr>
<td>1</td>
<td>$\text{NSPACE}(n)$ (presumed “context-sensitive grammars”)</td>
</tr>
<tr>
<td>2</td>
<td>Grammars (hence called “context-free”)</td>
</tr>
<tr>
<td>3</td>
<td>Regular languages: $\text{DSPACE}(\text{const})$.</td>
</tr>
</tbody>
</table>

✓ Ancient complexity theory, important in its time.
Old systematics

- Chomsky (1956): grammars as rewriting $A \rightarrow BC$.
- Chomsky (1957): generalized rewriting $AB \rightarrow CD$.

The Chomsky hierarchy

0. Recursively enumerable sets.
1. \(\text{NSPACE}(n)\) (presumed “context-sensitive grammars”)
2. Grammars (hence called “context-free”)
3. Regular languages: \(\text{DSPACE}(\text{const})\).

✓ Ancient complexity theory, important in its time.

- \(AB \rightarrow CD\): nothing else but computation.
A terminological issue

- Chomsky’s “context-free grammars”: a chance name for a fundamental concept.
A terminological issue

- Chomsky’s “context-free grammars”: a chance name for a fundamental concept.
- Chomsky’s “context-sensitive”: abandoned direction.
A terminological issue

- Chomsky’s “context-free grammars”: a chance name for a fundamental concept.
- Chomsky’s “context-sensitive”: abandoned direction.
- None of today’s major models use contexts.
A terminological issue

- Chomsky’s “context-free grammars”: a chance name for a fundamental concept.
- Chomsky’s “context-sensitive”: abandoned direction.
- None of today’s major models use contexts.
  - Joshi’s “mildly context-sensitive”: actually, no contexts.

Why the strange name “context-free”, then?
Main effect: new students led astray.
Proposed name: ordinary grammars
Other grammar families: in relation to the ordinary.
A terminological issue

- Chomsky’s “context-free grammars”: a chance name for a fundamental concept.
- Chomsky’s “context-sensitive”: abandoned direction.
- None of today’s major models use contexts.
  - Joshi’s “mildly context-sensitive”: actually, no contexts.
- Why the strange name “context-free”, then?
A terminological issue

- Chomsky’s “context-free grammars”: a chance name for a fundamental concept.
- Chomsky’s “context-sensitive”: abandoned direction.
- None of today’s major models use contexts.
  - Joshi’s “mildly context-sensitive”: actually, no contexts.
- Why the strange name “context-free”, then?
- Main effect: new students led astray.
A terminological issue

- Chomsky’s “context-free grammars”: a chance name for a fundamental concept.
- Chomsky’s “context-sensitive”: abandoned direction.
- None of today’s major models use contexts.
  - Joshi’s “mildly context-sensitive”: actually, no contexts.
- Why the strange name “context-free”, then?
- Main effect: new students led astray.
- Proposed name:
A terminological issue

- Chomsky’s “context-free grammars”: a chance name for a fundamental concept.
- Chomsky’s “context-sensitive”: abandoned direction.
- None of today’s major models use contexts.
  - Joshi’s “mildly context-sensitive”: actually, no contexts.
- Why the strange name “context-free”, then?
- Main effect: new students led astray.
- Proposed name:

  ordinary grammars
A terminological issue

- Chomsky’s “context-free grammars”: a chance name for a fundamental concept.
- Chomsky’s “context-sensitive”: abandoned direction.
- None of today’s major models use contexts.
  - Joshi’s “mildly context-sensitive”: actually, no contexts.
- Why the strange name “context-free”, then?
- Main effect: new students led astray.
- Proposed name:
  - ordinary grammars

- Other grammar families: in relation to the ordinary.
Part II

Grammar families
Intuitive syntactic descriptions

- Substrings with known properties, one after another.

Example (the Dyck language)

- If $w$ is well-nested, then so is $awb$.
- If $u$ and $v$ are well-nested, then so is $uv$.

As a grammar:

$$S \rightarrow \varepsilon | aSb | SS$$
Intuitive syntactic descriptions

- Substrings with known properties, one after another.

Example (the Dyck language)

- $\varepsilon$ is well-nested.
Intuitive syntactic descriptions

- Substrings with known properties, one after another.

Example (the Dyck language)

- $\varepsilon$ is well-nested.
- if $w$ is well-nested, then so is $awb$. 
Intuitive syntactic descriptions

- Substrings with known properties, one after another.

**Example (the Dyck language)**

- ε is well-nested.
- if \( w \) is well-nested, then so is \( awb \).
- if \( u \) and \( v \) are well-nested, then so is \( uv \).
Intuitive syntactic descriptions

- Substrings with known properties, one after another.

Example (the Dyck language)

- $\varepsilon$ is well-nested.
- If $w$ is well-nested, then so is $awb$.
- If $u$ and $v$ are well-nested, then so is $uv$.

As a grammar: $S \rightarrow \varepsilon \mid aSb \mid SS$. 
Intuitive syntactic descriptions

- Substrings with known properties, one after another.

Example (the Dyck language)

- $\varepsilon$ is well-nested.
- If $w$ is well-nested, then so is $awb$.
- If $u$ and $v$ are well-nested, then so is $uv$.

- As a grammar: $S \rightarrow \varepsilon \mid aSb \mid SS$.

Definition

Ordinary grammar: $G = (\Sigma, N, R, S)$, with rules

$$A \rightarrow X_1 \ldots X_\ell \quad (\ell \geq 0, \ X_1, \ldots, X_\ell \in \Sigma \cup N)$$
Intuitive syntactic descriptions

- Substrings with known properties, one after another.

Example (the Dyck language)

- $\varepsilon$ is well-nested.
- If $w$ is well-nested, then so is $awb$.
- If $u$ and $v$ are well-nested, then so is $uv$.

As a grammar: $S \rightarrow \varepsilon \mid aSb \mid SS$.

Definition

Ordinary grammar: $G = (\Sigma, N, R, S)$, with rules

$$A \rightarrow X_1 \ldots X_\ell \quad (\ell \geq 0, X_1, \ldots, X_\ell \in \Sigma \cup N)$$

“$\rightarrow$” denotes implication in the other direction.
Form of syntactic descriptions

“if \( u \) and \( v \) are well-nested, then so is \( uv \)”
Form of syntactic descriptions

“If $u$ and $v$ are well-nested, then so is $uv$”

- “well-nested”: syntactic category.
Form of syntactic descriptions

“if $u$ and $v$ are well-nested, then so is $uv$”

- “well-nested”: syntactic category.
- $u$ and $v$: constituents.
Form of syntactic descriptions

“if $u$ and $v$ are well-nested, then so is $uv$”

- “well-nested”: syntactic category.
- $u$ and $v$: constituents.
- $uv$: operation on constituents.
Form of syntactic descriptions

“if $u$ and $v$ are well-nested, then so is $uv$”

- “well-nested”: syntactic category.
- $u$ and $v$: constituents.
- $uv$: operation on constituents.
- Quantifier over $u$ and $v$: logical operation.
Form of syntactic descriptions

“if $u$ and $v$ are well-nested, then so is $uv$”

- “well-nested”: syntactic category.
- $u$ and $v$: constituents.
- $uv$: operation on constituents.
- Quantifier over $u$ and $v$: logical operation.

“if $w$ is well-nested, then so is $awb$” (2nd rule)
Form of syntactic descriptions

“if $u$ and $v$ are well-nested, then so is $uv$”

- “well-nested”: syntactic category.
- $u$ and $v$: constituents.
- $uv$: operation on constituents.
- Quantifier over $u$ and $v$: logical operation.

“if $w$ is well-nested, then so is $awb$” (2nd rule)

- Disjunction between rules: another logical operation.
Form of syntactic desriptions

“if $u$ and $v$ are well-nested, then so is $uv$”

- “well-nested”: syntactic category.
- $u$ and $v$: constituents.
- $uv$: operation on constituents.
- Quantifier over $u$ and $v$: logical operation.

“if $w$ is well-nested, then so is $awb$” (2nd rule)

- Disjunction between rules: another logical operation.

Formal grammar: a logic for describing syntax.
Grammar families

A family of grammars is characterized by three elements.
Grammar families

A family of grammars is characterized by three elements.

1. **Constituents.**
   - In ordinary grammars: substrings.
Grammar families

A family of grammars is characterized by three elements.

1. **Constituents.**
   - In ordinary grammars: substrings.

2. **Operations on constituents.**
   - In ordinary grammars: concatenation.
Grammar families

A family of grammars is characterized by three elements.

1. **Constituents.**
   - In ordinary grammars: substrings.

2. **Operations on constituents.**
   - In ordinary grammars: concatenation.

3. **Logical operations.**
   - In ordinary grammars: “∃” as part of concatenation; disjunction.
A family of grammars is characterized by three elements.

1. **Constituents.**
   - In ordinary grammars: substrings.

2. **Operations on constituents.**
   - In ordinary grammars: concatenation.

3. **Logical operations.**
   - In ordinary grammars: “∃” as part of concatenation; disjunction.

Each element can be modified.
A family of grammars is characterized by three elements.

1. **Constituents.**
   - In ordinary grammars: substrings.

2. **Operations on constituents.**
   - In ordinary grammars: concatenation.

3. **Logical operations.**
   - In ordinary grammars: “∃” as part of concatenation; disjunction.

- Each element can be modified.
- Restrictions and extensions of ordinary grammars.
Restricting ordinary grammars

Linear grammars

Operations on constituents: $aw$, $wa$, with $a \in \Sigma$.
Restricting ordinary grammars

**Linear grammars**

Operations on constituents: $aw$, $wa$, with $a \in \Sigma$.

**Unambiguous grammars**

Logical operations: disjoint disjunction; “∃!”.
Restricting ordinary grammars

**Linear grammars**
Operations on constituents: $aw$, $wa$, with $a \in \Sigma$.

**Unambiguous grammars**
Logical operations: disjoint disjunction; “∃!”.

- Input-driven automata (Mehlhorn, 1980).
Restricting ordinary grammars

**Linear grammars**
Operations on constituents: $aw, wa$, with $a \in \Sigma$.

**Unambiguous grammars**
Logical operations: disjoint disjunction; “∃!”.

- Input-driven automata (Mehlhorn, 1980).
  - “visibly pushdown automata” (Alur, Madhusudan, 2004)
Restricting ordinary grammars

**Linear grammars**

Operations on constituents: $aw$, $wa$, with $a \in \Sigma$.

**Unambiguous grammars**

Logical operations: disjoint disjunction; “∃!”.

- Input-driven automata (Mehlhorn, 1980).
  - “visibly pushdown automata” (Alur, Madhusudan, 2004)
  - Corresponding “bracketed” grammars.
Restricting ordinary grammars

**Linear grammars**

Operations on constituents: \( aw, wa, \) with \( a \in \Sigma \).

**Unambiguous grammars**

Logical operations: disjoint disjunction; “\( \exists! \)”.

- Input-driven automata (Mehlhorn, 1980).
  - “visibly pushdown automata” (Alur, Madhusudan, 2004)
  - Corresponding “bracketed” grammars.

**Several bracketed grammar models**

Constituents: well-nested strings;
operation on constituents: \( <u> \nu \).
Extension with conjunction

- Add another logical operation: conjunction.
Extension with conjunction

- Add another logical operation: conjunction.

**Definition (Okhotin, 2001)**

Conjunctive grammar: \( G = (\Sigma, N, R, S) \), with rules

\[
A \rightarrow \alpha_1 \& \ldots \& \alpha_n \quad (n \geq 1, \quad \alpha_1, \ldots, \alpha_n \in (\Sigma \cup N)^*)
\]
Add another logical operation: conjunction.

**Definition (Okhotin, 2001)**

Conjunctive grammar: $G = (\Sigma, N, R, S)$, with rules

$$A \rightarrow \alpha_1 \& \ldots \& \alpha_n \quad (n \geq 1, \alpha_1, \ldots, \alpha_n \in (\Sigma \cup N)^*)$$

“If $w$ is represented according to each conjunct $\alpha_i$, then $w$ has the property $A$”.
Extension with conjunction

- Add another logical operation: conjunction.

**Definition (Okhotin, 2001)**

Conjunctive grammar: \( G = (\Sigma, N, R, S) \), with rules

\[
A \rightarrow \alpha_1 \& \ldots \& \alpha_n \quad (n \geq 1, \ \alpha_1, \ldots, \alpha_n \in (\Sigma \cup N)^*)
\]

- “If \( w \) is represented according to each conjunct \( \alpha_i \), then \( w \) has the property \( A \)."

**Grammar for \( \{ a^n b^n c^n \mid n \geq 0 \} \)**
Extension with conjunction

- Add another logical operation: conjunction.

**Definition (Okhotin, 2001)**

Conjunctive grammar: \( G = (\Sigma, N, R, S) \), with rules

\[
A \rightarrow \alpha_1 \& \ldots \& \alpha_n \quad (n \geq 1, \alpha_1, \ldots, \alpha_n \in (\Sigma \cup N)^*)
\]

- “If \( w \) is represented according to each conjunct \( \alpha_i \), then \( w \) has the property \( A \)”.

**Grammar for \( \{a^nb^n c^n \mid n \geq 0\} \)**

\[
\begin{align*}
S & \rightarrow AB \& \ldots \\
A & \rightarrow aA \mid \varepsilon \\
B & \rightarrow bBc \mid \varepsilon
\end{align*}
\]
Add another logical operation: conjunction.

**Definition (Okhotin, 2001)**

Conjunctive grammar: $G = (\Sigma, N, R, S)$, with rules

$$A \rightarrow \alpha_1 \& \ldots \& \alpha_n \quad (n \geq 1, \alpha_1, \ldots, \alpha_n \in (\Sigma \cup N)^*)$$

“If $w$ is represented according to each conjunct $\alpha_i$, then $w$ has the property $A$”.

**Grammar for $\{a^n b^n c^n \mid n \geq 0\}$**

$$S \rightarrow AB \& DC$$

$$A \rightarrow aA \mid \varepsilon$$

$$B \rightarrow bBc \mid \varepsilon$$

$$D \rightarrow aDb \mid \varepsilon$$

$$C \rightarrow cC \mid \varepsilon$$
Extension with conjunction and negation

**Definition (Okhotin, 2004)**

Boolean grammar: \( G = (\Sigma, N, R, S) \), with rules

\[
A \rightarrow \alpha_1 \& \ldots \& \alpha_m \& \neg \beta_1 \& \ldots \& \neg \beta_n
\]
Extension with conjunction and negation

Definition (Okhotin, 2004)

Boolean grammar: \( G = (\Sigma, N, R, S) \), with rules

\[ A \rightarrow \alpha_1 \& \ldots \& \alpha_m \& \neg \beta_1 \& \ldots \& \neg \beta_n \]

- “If \( w \) is represented according to each conjunct \( \alpha_i \), and is not represented according to any \( \beta_j \), then \( w \) has the property \( A \)”.
Extension with conjunction and negation

**Definition (Okhotin, 2004)**

Boolean grammar: \( G = (\Sigma, N, R, S) \), with rules

\[ A \rightarrow \alpha_1 \& \ldots \& \alpha_m \& \neg \beta_1 \& \ldots \& \neg \beta_n \]

- “If \( w \) is represented according to each conjunct \( \alpha_i \), and is not represented according to any \( \beta_j \), then \( w \) has the property \( A \).”

Grammar for \( \{a^mb^nc^n \mid m \neq n\} \)
Extension with conjunction and negation

Definition (Okhotin, 2004)

Boolean grammar: \( G = (\Sigma, N, R, S) \), with rules
\[
A \rightarrow \alpha_1 \& \ldots \& \alpha_m \& \neg \beta_1 \& \ldots \& \neg \beta_n
\]

"If \( w \) is represented according to each conjunct \( \alpha_i \), and is not represented according to any \( \beta_j \), then \( w \) has the property \( A \)."

Grammar for \( \{a^m b^n c^n \mid m \neq n\} \)

\[
\begin{align*}
S & \rightarrow AB \& \neg DC \\
A & \rightarrow aA \mid \varepsilon \\
B & \rightarrow bBc \mid \varepsilon \\
D & \rightarrow aDb \mid \varepsilon \\
C & \rightarrow cC \mid \varepsilon
\end{align*}
\]
Extension using substrings with gaps

- Constituents $u_1$-GAP-$u_2$-GAP-...-GAP-$u_k$. 
Extension using substrings with gaps

- Constituents \( u_1\text{-GAP-}u_2\text{-GAP-} \ldots \text{-GAP-}u_k \).

**Definition (Joshi et al., 1987; Seki et al., 1991)**

Multi-component gr.: \( G = (\Sigma, N, \text{dim}, R, S) \), with each \( A \in N \) having *dimension* \( k = \text{dim } A \), and with rules

\[
A(\alpha_1, \ldots, \alpha_k) \rightarrow B_1(x_{1,1}, \ldots, x_{1,m_1}), \ldots, B_\ell(x_{\ell,1}, \ldots, x_{\ell,m_\ell})
\]
Extension using substrings with gaps

- Constituents $u_1$-GAP-$u_2$-GAP- $\ldots$-GAP-$u_k$.

**Definition (Joshi et al., 1987; Seki et al., 1991)**

Multi-component gr.: $G = (\Sigma, N, \text{dim}, R, S)$, with each $A \in N$ having dimension $k = \text{dim} A$, and with rules

$$A(\alpha_1, \ldots, \alpha_k) \rightarrow B_1(x_{1,1}, \ldots, x_{1,m_1}), \ldots, B_\ell(x_{\ell,1}, \ldots, x_{\ell,m_\ell})$$

- $m_1$-tuple, $\ldots$, $m_\ell$-tuple combined into a $k$-tuple.
Extension using substrings with gaps

- Constituents $u_1$-GAP-$u_2$-GAP-$\ldots$-GAP-$u_k$.

**Definition (Joshi et al., 1987; Seki et al., 1991)**

Multi-component gr.: $G = (\Sigma, N, \text{dim}, R, S)$, with each $A \in N$ having *dimension* $k = \text{dim} A$, and with rules

$$A(\alpha_1, \ldots, \alpha_k) \rightarrow B_1(x_{1,1}, \ldots, x_{1,m_1}), \ldots, B_\ell(x_{\ell,1}, \ldots, x_{\ell,m_\ell})$$

- $m_1$-tuple, $\ldots$, $m_\ell$-tuple combined into a $k$-tuple.

**Grammar for** $\{a^n b^n c^n d^n \mid n \geq 0\}$
Extension using substrings with gaps

- Constituents $u_1$-GAP-$u_2$-GAP- $\ldots$-GAP-$u_k$.

**Definition (Joshi et al., 1987; Seki et al., 1991)**

Multi-component gr.: $G = (\Sigma, N, \text{dim}, R, S)$, with each $A \in N$ having dimension $k = \text{dim} \ A$, and with rules

$$A(\alpha_1, \ldots, \alpha_k) \rightarrow B_1(x_{1,1}, \ldots, x_{1,m_1}), \ldots, B_\ell(x_{\ell,1}, \ldots, x_{\ell,m_\ell})$$

- $m_1$-tuple, $\ldots$, $m_\ell$-tuple combined into a $k$-tuple.

**Grammar for** $\{a^n b^n c^n d^n \mid n \geq 0\}$

- $S(xy) \rightarrow A(x, y)$
- $A(AXB, CYD) \rightarrow A(x, y)$
- $A(\varepsilon, \varepsilon) \rightarrow \text{TRUE}$
Well-nested multi-component grammars

Constituents $u_1$-GAP-$u_2$-GAP-...-GAP-$u_k$. 
Well-nested multi-component grammars

- Constituents $u_1$-GAP-$u_2$-GAP-$\ldots$-GAP-$u_k$.

$A(\alpha_1, \ldots, \alpha_k) \rightarrow B_1(x_{1,1}, \ldots, x_{1,m_1}), \ldots, B_\ell(x_{\ell,1}, \ldots, x_{\ell,m_\ell})$
Well-nested multi-component grammars

- Constituents $u_1$-GAP-$u_2$-GAP-$\ldots$-GAP-$u_k$.

$A(\alpha_1, \ldots, \alpha_k) \rightarrow B_1(x_{1,1}, \ldots, x_{1,m_1}), \ldots, B_\ell(x_{\ell,1}, \ldots, x_{\ell,m_\ell})$

- $\alpha_1 \ldots \alpha_k$: all $x_{i,j}$ shuffled.
Well-nested multi-component grammars

- Constituents $u_1$-GAP-$u_2$-GAP-$\ldots$-GAP-$u_k$.

$A(\alpha_1, \ldots, \alpha_k) \rightarrow B_1(x_{1,1}, \ldots, x_{1,m_1}), \ldots, B_{\ell}(x_{\ell,1}, \ldots, x_{\ell,m_{\ell}})$

- $\alpha_1 \ldots \alpha_k$: all $x_{i,j}$ shuffled.
- **Well-nested**: in $\alpha_1 \ldots \alpha_k$, no crossings
  
  $\ldots x_{i,j} \ldots x_{i',m} \ldots x_{i,k} \ldots x_{i',n} \ldots$
Well-nested multi-component grammars

- Constituents $u_1$-GAP-$u_2$-GAP- $\ldots$ -GAP-$u_k$.

$$A(\alpha_1, \ldots, \alpha_k) \rightarrow B_1(x_{1,1}, \ldots, x_{1,m_1}), \ldots, B_\ell(x_{\ell,1}, \ldots, x_{\ell,m_\ell})$$

- $\alpha_1 \ldots \alpha_k$: all $x_{i,j}$ shuffled.

- **Well-nested**: in $\alpha_1 \ldots \alpha_k$, no crossings

  $\ldots x_{i,j} \ldots x_{i',m} \ldots x_{i,k} \ldots x_{i',n} \ldots$

- **Well-nested 2-component**: tree-adjoining, also head.
Well-nested multi-component grammars

- Constituents $u_1$-GAP-$u_2$-GAP-$\ldots$-GAP-$u_k$.

$$A(\alpha_1, \ldots, \alpha_k) \rightarrow B_1(x_{1,1}, \ldots, x_{1,m_1}), \ldots, B_\ell(x_{\ell,1}, \ldots, x_{\ell,m_\ell})$$

- $\alpha_1 \ldots \alpha_k$: all $x_{i,j}$ shuffled.
- **Well-nested**: in $\alpha_1 \ldots \alpha_k$, no crossings
  $$\ldots x_{i,j} \ldots x_{i',m} \ldots x_{i,k} \ldots x_{i',n} \ldots$$
- **Well-nested 2-component**: tree-adjoining, also head.
  - Constituents: substrings $u$, pairs $(u, v)$. 

- Main operation: wrap $(u, v)$ around $(x, y)$, get $(ux, yv)$.

Grammar models combining these ideas, e.g.,

- unambiguous tree-adjoining,
- linear conjunctive, etc.
Well-nested multi-component grammars

- Constituents $u_1$-GAP-$u_2$-GAP- ... -GAP-$u_k$.
  
  $A(\alpha_1, \ldots, \alpha_k) \rightarrow B_1(x_{1,1}, \ldots, x_{1,m_1}), \ldots, B_\ell(x_{\ell,1}, \ldots, x_{\ell,m_\ell})$

- $\alpha_1 \ldots \alpha_k$: all $x_{i,j}$ shuffled.

- Well-nested: in $\alpha_1 \ldots \alpha_k$, no crossings
  
  $\ldots x_{i,j} \ldots x_{i',m} \ldots x_{i,k} \ldots x_{i',n} \ldots$

- Well-nested 2-component: tree-adjoining, also head.
  
  ▶ Constituents: substrings $u$, pairs $(u, v)$.
  ▶ Main operation: wrap $(u, v)$ around $(x, y)$, get $(ux, yv)$.  

▶ Constituents: substrings $u$, pairs $(u, v)$.
Well-nested multi-component grammars

- Constituents $u_1$-GAP-$u_2$-GAP-$\ldots$-GAP-$u_k$.

\[
A(\alpha_1, \ldots, \alpha_k) \rightarrow B_1(x_{1,1}, \ldots, x_{1,m_1}), \ldots, B_{\ell}(x_{\ell,1}, \ldots, x_{\ell,m_\ell})
\]

- $\alpha_1 \ldots \alpha_k$: all $x_{i,j}$ shuffled.
- Well-nested: in $\alpha_1 \ldots \alpha_k$, no crossings

\[
\ldots x_{i,j} \ldots x_{i',m} \ldots x_{i,k} \ldots x_{i',n} \ldots
\]
- Well-nested 2-component: tree-adjoining, also head.
  - Constituents: substrings $u$, pairs $(u, v)$.
  - Main operation: wrap $(u, v)$ around $(x, y)$, get $(ux, yv)$.
- Grammar models combining these ideas, e.g.,
Well-nested multi-component grammars

- Constituents $u_1\text{-GAP-}u_2\text{-GAP-}\ldots\text{-GAP-}u_k$.
  
  $A(\alpha_1, \ldots, \alpha_k) \rightarrow B_1(x_1,1, \ldots, x_1,m_1), \ldots, B_\ell(x_\ell,1, \ldots, x_\ell,m_\ell)$

- $\alpha_1 \ldots \alpha_k$: all $x_{i,j}$ shuffled.

- Well-nested: in $\alpha_1 \ldots \alpha_k$, no crossings
  
  \[
  \ldots x_{i,j} \ldots x_{i',m} \ldots x_{i,k} \ldots x_{i',n} \ldots
  \]

- Well-nested 2-component: tree-adjoining, also head.
  
  - Constituents: substrings $u$, pairs $(u, v)$.
  - Main operation: wrap $(u, v)$ around $(x, y)$, get $(ux, yv)$.

- Grammar models combining these ideas, e.g.,
  
  - unambiguous tree-adjoining,
Well-nested multi-component grammars

- Constituents $u_1$-GAP-$u_2$-GAP-$\ldots$-GAP-$u_k$.

$$A(\alpha_1, \ldots, \alpha_k) \rightarrow B_1(x_{1,1}, \ldots, x_{1,m_1}), \ldots, B_\ell(x_{\ell,1}, \ldots, x_{\ell,m_\ell})$$

- $\alpha_1 \ldots \alpha_k$: all $x_{i,j}$ shuffled.
- Well-nested: in $\alpha_1 \ldots \alpha_k$, no crossings
  
  $\ldots x_{i,j} \ldots x_{i',m} \ldots x_{i,k} \ldots x_{i',n} \ldots$

- Well-nested 2-component: tree-adjoining, also head.
  
  - Constituents: substrings $u$, pairs $(u, v)$.
  - Main operation: wrap $(u, v)$ around $(x, y)$, get $(ux, yv)$.

- Grammar models combining these ideas, e.g.,
  
  - unambiguous tree-adjoining,
  - linear conjunctive, etc.
The hierarchy

- Reg
- LLLin
- LRLin
- UnambLin
- Lin
- Unamb
- Ordinary
- UnambTAG
- TAG
- Multi
- Conj
- Bool
- UnambConj
- UnambBool
- IDPDA
- LinConj
\[ \{ wcw \mid w \in \{a, b\}^* \} \]: LinConj, UnambTAG.
\[ \{ wcw \mid w \in \{a, b\}^* \}: \text{LinConj, UnambTAG.} \]
\[ \{ ww \mid w \in \{a, b\}^* \}: \text{Bool, UnambTAG.} \]
The hierarchy

\[ \{ wcw \mid w \in \{a, b\}^* \} : \text{LinConj, UnambTAG}. \]
\[ \{ ww \mid w \in \{a, b\}^* \} : \text{Bool, UnambTAG}. \]
\[ \{ a^{2^n} \mid n \geq 0 \} : \text{UnambConj}. \]
Part III

Mathematical definitions
Definition by logical derivation

\[ S \rightarrow \text{NP} \text{ VP} \]
Definition by logical derivation

- $S \rightarrow \text{NP VP}$

“If $u$ is NP and $v$ is VP, then $uv$ is a sentence”.

Propositions $A(w)$, meaning "$w$ has the property $A$".

Definition by logical derivation (folklore)

$\text{NP (Every man)} \rightarrow \text{VP (is mortal)}$

$S (\text{Every man is mortal}) \rightarrow \text{NP VP}$

A proof tree is a parse tree.

✓ Directly formalizes the intuition.
Definition by logical derivation

- $S \rightarrow NP \ VP$

“If $u$ is NP and $v$ is VP, then $uv$ is a sentence”.

- Propositions $A(w)$, meaning “$w$ has the property $A$”.

A proof tree is a parse tree.

✓ Directly formalizes the intuition.
Definition by logical derivation

- \( S \rightarrow \text{NP VP} \)

“If \( u \) is NP and \( v \) is VP, then \( uv \) is a sentence”.

- Propositions \( A(w) \), meaning “\( w \) has the property \( A \)”.

Definition by logical derivation (folklore)

\[
\frac{\text{NP(Every man)}}{\text{S(Every man is mortal)}} \quad \frac{\text{VP(is mortal)}}{(S \rightarrow \text{NP VP})}
\]

A proof tree is a parse tree. ✓

Directly formalizes the intuition.
Definition by logical derivation

- $S \rightarrow \text{NP} \ \text{VP}$

“If $u$ is NP and $v$ is VP, then $uv$ is a sentence”.

- Propositions $A(w)$, meaning “$w$ has the property $A$”.

Definition by logical derivation (folklore)

\[
\begin{align*}
\text{NP}(\text{Every man}) & \quad \text{VP}(\text{is mortal}) \\
\hline
\text{S}(\text{Every man is mortal}) & \quad (S \rightarrow \text{NP} \ \text{VP})
\end{align*}
\]

- A proof tree is a parse tree.
Definition by logical derivation

- \( S \rightarrow \text{NP VP} \)

“If \( u \) is NP and \( v \) is VP, then \( uv \) is a sentence”.

- Propositions \( A(w) \), meaning “\( w \) has the property \( A \)”.

Definition by logical derivation (folklore)

\[
\frac{\text{NP}(\text{Every man}) \quad \text{VP}(\text{is mortal})}{\text{S}(\text{Every man is mortal})} (S \rightarrow \text{NP VP})
\]

- A proof tree is a parse tree.

✓ Directly formalizes the intuition.
Logical derivation for generalizations

For conjunctive grammars

\[ \frac{A(a^n) \quad B(b^n c^n) \quad D(a^n b^n) \quad C(c^n)}{S(a^n b^n c^n)} \]

\( (S \to AB \& DC) \)

Not for Boolean grammars.

For multi-component grammars

\[ A(ab, cd) \]

\[ A(aabb, ccdd) \]

\[ (A(axb, cyd) \to A(x, y)) \]

Works for all kinds of grammars without negation.

Alexander Okhotin
Logical derivation for generalizations

For conjunctive grammars

\[
\frac{A(a^n) \quad B(b^n c^n) \quad D(a^n b^n) \quad C(c^n)}{S(a^n b^n c^n)} (S \rightarrow AB \& DC)
\]

- Not for Boolean grammars.
Logical derivation for generalizations

For conjunctive grammars

\[
\frac{A(a^n) \quad B(b^n c^n) \quad D(a^n b^n) \quad C(c^n)}{S(a^n b^n c^n)} \quad (S \rightarrow AB \& DC)
\]

- Not for Boolean grammars.

For multi-component grammars

\[
\frac{A(ab, cd)}{A(aabb, ccdd)} \quad (A(axb, cyd) \rightarrow A(x, y))
\]
Logical derivation for generalizations

For conjunctive grammars

\[
\frac{A(a^n) \quad B(b^n c^n) \quad D(a^n b^n) \quad C(c^n)}{S(a^n b^n c^n)} \quad (S \to AB \& DC)
\]

- Not for Boolean grammars.

For multi-component grammars

\[
\frac{A(ab, cd)}{A(aabb, ccdd)} \quad (A(axb, cyd) \to A(x, y))
\]

Works for all kinds of grammars without negation.
Definition by language equations

- $S \rightarrow NP \ VP \ | \ ...$
Definition by language equations

- \( S \rightarrow \text{NP VP} \mid \ldots \)

“\( w \) is a sentence if and only if \( w = uv \), with \( u \) an NP and \( v \) a VP, or\ldots”.
Definition by language equations

- $S \rightarrow \text{NP VP} \mid \ldots$

“$w$ is a sentence if and only if $w = uv$, with $u$ an NP and $v$ a VP, or...”.

- $S, \text{NP}, \text{VP}$: unknown languages.
Definition by language equations

- \( S \rightarrow \text{NP} \text{ VP} | \ldots \)

"\( w \) is a sentence if and only if \\
\( w = uv \), with \( u \) an NP and \( v \) a VP, or\ldots".

- \( S, \text{NP}, \text{VP}: \) unknown languages.
- Equation \( S = (\text{NP} \cdot \text{VP}) \bigcup \ldots \)
Definition by language equations

- $S \rightarrow \text{NP VP} \mid \ldots$

“$w$ is a sentence if and only if $w = uv$, with $u$ an NP and $v$ a VP, or . . .”.

- $S, \text{NP, VP}$: unknown languages.
- Equation $S = (\text{NP} \cdot \text{VP}) \cup \ldots$

Definition (Ginsburg, Rice, 1962)

Grammar $G = (\Sigma, N, R, S)$ as a least solution of:

$$A = \bigcup_{A \rightarrow X_1 \ldots X_\ell \in R} X_1 \cdot \ldots \cdot X_\ell \quad (\text{for each } A \in N)$$
Language equations for extensions

Conjunctive: conjunction as intersection.

\[ S \rightarrow AB \land DC \text{ as } S = (A \cdot B) \cap (D \cdot C). \]

Boolean: negation as complementation.

\[ S \rightarrow AB \land \neg DC \text{ as } S = (A \cdot B) \cap D \cdot C. \]

Can express contradiction:

\[ S \rightarrow \neg S. \]
Language equations for extensions

Conjunctive: conjunction as intersection.

\[ S \rightarrow AB \& DC \text{ as } S = (A \cdot B) \cap (D \cdot C). \]
Language equations for extensions

Conjunctive: conjunction as intersection.

\[ S \rightarrow AB \& DC \text{ as } S = (A \cdot B) \cap (D \cdot C). \]

Boolean: negation as complementation.

Can express contradiction: \( S \rightarrow \neg S \).

▶ Resolved by restricting the grammar.
▶ Also resolved by using 3-valued logic.

Multi-component: unknown sets of \( k \)-tuples.

Applies to all grammar families.

Alexander Okhotin
Principles and recurring ideas of grammars
LATA 2018 21 / 41
Language equations for extensions

Conjunctive: conjunction as intersection.

\[ S \rightarrow AB \& DC \text{ as } S = (A \cdot B) \cap (D \cdot C). \]

Boolean: negation as complementation.

\[ S \rightarrow AB \& \neg DC \text{ as } S = (A \cdot B) \cap D \cdot \overline{C}. \]

Can express contradiction: \( S \rightarrow \neg S \).

Resolved by restricting the grammar.
Also resolved by using 3-valued logic.

Multi-component: unknown sets of \( k \)-tuples.
Applies to all grammar families.
Language equations for extensions

Conjunctive: conjunction as intersection.

\[ S \rightarrow AB \& DC \text{ as } S = (A \cdot B) \cap (D \cdot C). \]

Boolean: negation as complementation.

\[ S \rightarrow AB \& \neg DC \text{ as } S = (A \cdot B) \cap \overline{D} \cdot C. \]
\[ \text{Can express contradiction: } S \rightarrow \neg S. \]
Language equations for extensions

Conjunctive: conjunction as intersection.

- $S \rightarrow AB \& DC$ as $S = (A \cdot B) \cap (D \cdot C)$.

Boolean: negation as complementation.

- $S \rightarrow AB \& \overline{DC}$ as $S = (A \cdot B) \cap \overline{D} \cdot C$.
- Can express contradiction: $S \rightarrow \overline{S}$.
  - Resolved by restricting the grammar.
Language equations for extensions

Conjunctive: conjunction as intersection.

\[ S \rightarrow AB \& DC \text{ as } S = (A \cdot B) \cap (D \cdot C). \]

Boolean: negation as complementation.

\[ S \rightarrow AB \& \neg DC \text{ as } S = (A \cdot B) \cap \overline{D \cdot C}. \]

Can express contradiction: \( S \rightarrow \neg S \).

- Resolved by restricting the grammar.
- Also resolved by using 3-valued logic.
Language equations for extensions

Conjunctive: conjunction as intersection.

\[ S \rightarrow AB \& DC \text{ as } S = (A \cdot B) \cap (D \cdot C). \]

Boolean: negation as complementation.

\[ S \rightarrow AB \& \neg DC \text{ as } S = (A \cdot B) \cap \overline{D} \cdot C. \]
- Can express contradiction: \( S \rightarrow \neg S \).
  - Resolved by restricting the grammar.
  - Also resolved by using 3-valued logic.

Multi-component: unknown sets of \( k \)-tuples.
Language equations for extensions

**Conjunctive:** conjunction as intersection.

- \( S \rightarrow AB \& DC \) as \( S = (A \cdot B) \cap (D \cdot C) \).

**Boolean:** negation as complementation.

- \( S \rightarrow AB \& \neg DC \) as \( S = (A \cdot B) \cap D \cdot C \).
- Can express contradiction: \( S \rightarrow \neg S \).
  - Resolved by restricting the grammar.
  - Also resolved by using 3-valued logic.

**Multi-component:** unknown sets of \( k \)-tuples.

Applies to all grammar families.
Definition by rewriting

By string rewriting (Chomsky, 1956)

\[ S \Rightarrow NP \ VP \Rightarrow \ldots \Rightarrow NP \ is \ mortal \Rightarrow \ldots \Rightarrow \]

Every man is mortal
Definition by rewriting

By string rewriting (Chomsky, 1956)

\[ S \rightarrow NP \ VP \rightarrow \ldots \rightarrow NP \text{ is mortal} \rightarrow \ldots \rightarrow \]

Every man is mortal

- Equivalent to logical rewriting, therefore correct.
Definition by rewriting

By string rewriting (Chomsky, 1956)

\[ S \Rightarrow NP \ VP \Rightarrow \ldots \Rightarrow NP \text{ is mortal} \Rightarrow \ldots \Rightarrow \]

Every man is mortal

- Equivalent to logical rewriting, therefore correct.
- May give a totally wrong impression of grammars!
Definition by rewriting

By string rewriting (Chomsky, 1956)

\[ S \rightarrow NP \ VP \rightarrow \ldots \rightarrow NP \text{ is mortal} \rightarrow \ldots \rightarrow \]

Every man is mortal

- Equivalent to logical rewriting, therefore correct.
- May give a totally wrong impression of grammars!

Conjunctive: *term rewriting*

\[ S \rightarrow (AB \& DC) \rightarrow \ldots \rightarrow (w \& w) \rightarrow w. \]
Definition by rewriting

By string rewriting (Chomsky, 1956)

\[ S \Rightarrow NP \ VP \Rightarrow \ldots \Rightarrow NP \text{ is mortal} \Rightarrow \ldots \Rightarrow \]

Every man is mortal

- Equivalent to logical rewriting, therefore correct.
- May give a totally wrong impression of grammars!

Conjunctive: \textit{term rewriting}

\[ S \Rightarrow (AB \& DC) \Rightarrow \ldots \Rightarrow (w \& w) \Rightarrow w. \]

- Not for Boolean and multi-component.
Definition by rewriting

By string rewriting (Chomsky, 1956)

\[ S \Rightarrow NP \ VP \Rightarrow \ldots \Rightarrow NP \text{ is mortal} \Rightarrow \ldots \Rightarrow \]
Every man is mortal

- Equivalent to logical rewriting, therefore correct.
- May give a totally wrong impression of grammars!

Conjunctive: \textit{term rewriting}

\[ S \Rightarrow (AB \& DC) \Rightarrow \ldots \Rightarrow (w \& w) \Rightarrow w. \]

- Not for Boolean and multi-component.

Works for ordinary grammars, gives wrong ideas.
Part IV

Basic properties
Expressibility of operations

- A family of grammars.

Closure under operation $f$

If $L_1, \ldots, L_n$ are in the family, then so is $f(L_1, \ldots, L_n)$.

Practical value: a way of constructing grammars.

Explicit:
- \{·, ∪\} in ordinary grammars,
- \{·, ∪, ∩\} in conjunctive, etc.

Easily expressed: Kleene star, reversal.

Any other operations expressible?
Expressibility of operations

- A family of grammars.

**Closure under operation $f$**

If $L_1, \ldots, L_n$ are in the family, then so is $f(L_1, \ldots, L_n)$. 

Practical value: a way of constructing grammars.

Explicit: \{·, ∪\} in ordinary grammars, \{·, ∪, ∩\} in conjunctive, etc.

Easily expressed: Kleene star, reversal.

Any other operations expressible?
Expressibility of operations

- A family of grammars.

**Closure under operation** $f$

If $L_1, \ldots L_n$ are in the family, then so is $f(L_1, \ldots L_n)$.

- Practical value: a way of constructing grammars.
Expressibility of operations

- A family of grammars.

**Closure under operation \( f \)**

If \( L_1, \ldots, L_n \) are in the family, then so is \( f(L_1, \ldots, L_n) \).

- Practical value: a way of constructing grammars.
- Explicit: \( \{\cdot, \cup\} \) in ordinary grammars, \( \{\cdot, \cup, \cap\} \) in conjunctive, etc.
Expressibility of operations

- A family of grammars.

**Closure under operation $f$**

If $L_1, \ldots L_n$ are in the family, then so is $f(L_1, \ldots L_n)$.

- Practical value: a way of constructing grammars.
- Explicit: $\{\cdot, \cup\}$ in ordinary grammars, $\{\cdot, \cup, \cap\}$ in conjunctive, etc.
- Easily expressed: Kleene star, reversal.
Expressibility of operations

- A family of grammars.

Closure under operation $f$

If $L_1, \ldots, L_n$ are in the family, then so is $f(L_1, \ldots, L_n)$.

- Practical value: a way of constructing grammars.
- Explicit: $\{\cdot, \cup\}$ in ordinary grammars, $\{\cdot, \cup, \cap\}$ in conjunctive, etc.
- Easily expressed: Kleene star, reversal.
- Any other operations expressible?
Theorem (Bar-Hillel et al., 1962)

Ordinary grammars closed under $\cap \text{Reg}$. 
Expressibility of operations, II

Theorem (Bar-Hillel et al., 1962)

Ordinary grammars closed under ∩Reg.

✓ Embed DFA’s computation into a parse tree.
Theorem (Bar-Hillel et al., 1962)

Ordinary grammars closed under $\cap\text{Reg}$. 

✓ Embed DFA’s computation into a parse tree.

Ordinary grammars closed under finite transductions.
Expressibility of operations, II

Theorem (Bar-Hillel et al., 1962)

Ordinary grammars closed under $\cap \text{Reg}$.

✓ Embed DFA’s computation into a parse tree.

Ordinary grammars closed under finite transductions.

- Applies to linear; multi-component.
Expressibility of operations, II

Theorem (Bar-Hillel et al., 1962)

Ordinary grammars closed under $\cap\text{Reg}$.

✓ Embed DFA’s computation into a parse tree.

Ordinary grammars closed under finite transductions.

- Applies to linear; multi-component.
- Non-closure for unambiguous; conjunctive.
Expressibility of operations, II

Theorem (Bar-Hillel et al., 1962)

*Ordinary grammars closed under $\cap \mathsf{Reg}$.]*

✓ Embed DFA’s computation into a parse tree.

*Ordinary grammars closed under finite transductions.*

- Applies to linear; multi-component.
- Non-closure for unambiguous; conjunctive.
  - Closed under inverse transductions.
Expressibility of operations, II

Theorem (Bar-Hillel et al., 1962)

Ordinary grammars closed under \( \cap \text{Reg} \).

✓ Embed DFA’s computation into a parse tree.

Ordinary grammars closed under finite transductions.

- Applies to linear; multi-component.
- Non-closure for unambiguous; conjunctive.
  - Closed under inverse transductions.

Idea and the limits of its applicability.
Negative results using iteration

Ordinary grammars: pumping lemma (Bar-Hillel et al., 1962).
Negative results using iteration

Ordinary grammars: pumping lemma (Bar-Hillel et al., 1962).

✓ Large parse trees: insert repetitive structure.
Negative results using iteration

Ordinary grammars: pumping lemma (Bar-Hillel et al., 1962).

✓ Large parse trees: insert repetitive structure.
Extensions: Ogden (1968); Bader–Moura (1982).
Negative results using iteration

Ordinary grammars: pumping lemma (Bar-Hillel et al., 1962).

- Large parse trees: insert repetitive structure.
- Extensions: Ogden (1968); Bader–Moura (1982).
- Multi-component: restricted variants.
Negative results using iteration

Ordinary grammars: pumping lemma (Bar-Hillel et al., 1962).

✓ Large parse trees: insert repetitive structure.
  • Extensions: Ogden (1968); Bader–Moura (1982).
  • Multi-component: restricted variants.

Ordinary grammars: Parikh’s theorem.
Negative results using iteration

Ordinary grammars: pumping lemma (Bar-Hillel et al., 1962).

- Large parse trees: insert repetitive structure.
  - Extensions: Ogden (1968); Bader–Moura (1982).
  - Multi-component: restricted variants.

Ordinary grammars: Parikh’s theorem.

- Applies to multi-component.
Negative results using iteration

Ordinary grammars: pumping lemma (Bar-Hillel et al., 1962).

✓ Large parse trees: insert repetitive structure.
  • Extensions: Ogden (1968); Bader–Moura (1982).
  • Multi-component: restricted variants.

Ordinary grammars: Parikh’s theorem.

  • Applies to multi-component.

Methods based on iteration for grammars with disjunction. Nothing known for conjunctive!
Normal forms

Chomsky normal form: $A \rightarrow BC$, $A \rightarrow a$. 

Greibach normal form: $A \rightarrow a\alpha$.

Rosenkrantz normal form: $A \rightarrow a\alpha b$.

Applies to unambiguous, LL, LR.

Conjunctive, Boolean: not known.
Chomsky normal form: $A \rightarrow BC$, $A \rightarrow a$.

- Applies to unambiguous, LL, LR.
Normal forms

Chomsky normal form: $A \rightarrow BC$, $A \rightarrow a$.

- Applies to unambiguous, LL, LR.
- Applies to conjunctive: $A \rightarrow B_1 C_1 \& \ldots \& B_n C_n$. 
Normal forms

Chomsky normal form: $A \rightarrow BC$, $A \rightarrow a$.

- Applies to unambiguous, LL, LR.
- Applies to conjunctive: $A \rightarrow B_1 C_1 \& \ldots \& B_n C_n$.
- Not to multi-component grammars.
Normal forms

Chomsky normal form: \( A \rightarrow BC, \ A \rightarrow a. \)

- Applies to unambiguous, LL, LR.
- Applies to conjunctive: \( A \rightarrow B_1 C_1 \& \ldots \& B_n C_n. \)
- Not to multi-component grammars.

Greibach normal form: \( A \rightarrow a\alpha. \)
Rosenkrantz normal form: \( A \rightarrow a\alpha b. \)
Normal forms

Chomsky normal form: $A \rightarrow BC, \ A \rightarrow a$.

- Applies to unambiguous, LL, LR.
- Applies to conjunctive: $A \rightarrow B_1 C_1 \& \ldots \& B_n C_n$.
- Not to multi-component grammars.

Greibach normal form: $A \rightarrow a\alpha$.
Rosenkrantz normal form: $A \rightarrow a\alpha b$.

- Applies to unambiguous, linear, LL, LR.
Normal forms

Chomsky normal form: $A \rightarrow BC, A \rightarrow a$.
- Applies to unambiguous, LL, LR.
- Applies to conjunctive: $A \rightarrow B_1 C_1 \& \ldots \& B_n C_n$.
- Not to multi-component grammars.

Greibach normal form: $A \rightarrow a\alpha$.
Rosenkrantzen normal form: $A \rightarrow a\alpha b$.
- Applies to unambiguous, linear, LL, LR.
- Conjunctive, Boolean: not known.
Parsing time

Ordinary grammars: Cocke–Kasami–Younger

$G$ in CNF, given $w = a_1 \ldots a_n$, construct

$$T_{i,j} = \{ A \mid a_{i+1} \ldots a_j \in L_G(A) \}$$
Ordinary grammars: Cocke–Kasami–Younger

Given $G$ in CNF, given $w = a_1 \ldots a_n$, construct

$$T_{i,j} = \{ A | a_{i+1} \ldots a_j \in L_G(A) \}$$

- Time $\Theta(n^3)$, space $\Theta(n^2)$. 

By matrix multiplication (Valiant, 1975): time $\Theta(n^{\omega})$.

Conjunctive, Boolean: same algorithms.

Unambiguous: time $O(n^2)$.

Tree-adjoining: time $O(n^6)$, can do in time $O(n^{2 \omega})$.

$k$-component: time $O(n^{3k})$, can do in time $O(n^{\omega k})$.

Polynomial-time parsing for each family.
Parsing time

Ordinary grammars: Cocke–Kasami–Younger

$G$ in CNF, given $w = a_1 \ldots a_n$, construct

$$T_{i,j} = \{ A \mid a_{i+1} \ldots a_j \in L_G(A) \}$$

- Time $\Theta(n^3)$, space $\Theta(n^2)$.
- By matrix multiplication (Valiant, 1975): time $\Theta(n^\omega)$.
## Parsing time

### Ordinary grammars: Cocke–Kasami–Younger

Given a grammar $\mathcal{G}$ in Conjunctive Normal Form (CNF), given $w = a_1 \ldots a_n$, construct

$$T_{i,j} = \{ A \mid a_{i+1} \ldots a_j \in L_\mathcal{G}(A) \}$$

- Time $\Theta(n^3)$, space $\Theta(n^2)$.
- By matrix multiplication (Valiant, 1975): time $\Theta(n^\omega)$.
- Conjunctive, Boolean: same algorithms.
## Parsing time

### Ordinary grammars: Cocke–Kasami–Younger

Given a grammar $G$ in CNF and a string $w = a_1 \ldots a_n$, construct

$$T_{i,j} = \{ A \mid a_{i+1} \ldots a_j \in L_G(A) \}$$

- Time $\Theta(n^3)$, space $\Theta(n^2)$.
- By matrix multiplication (Valiant, 1975): time $\Theta(n^\omega)$.
- Conjunctive, Boolean: same algorithms.
- Unambiguous: time $O(n^2)$. 
Parsing time

Ordinary grammars: Cocke–Kasami–Younger

$G$ in CNF, given $w = a_1 \ldots a_n$, construct

$$T_{i,j} = \{A \mid a_{i+1} \ldots a_j \in L_G(A)\}$$

- Time $\Theta(n^3)$, space $\Theta(n^2)$.
- By matrix multiplication (Valiant, 1975): time $\Theta(n^\omega)$.
- Conjunctive, Boolean: same algorithms.
- Unambiguous: time $O(n^2)$.
- Tree-adjoining: time $O(n^6)$, can do in time $O(n^{2\omega})$. 
Ordinary grammars: Cocke–Kasami–Younger

\( G \) in CNF, given \( w = a_1 \ldots a_n \), construct

\[ T_{i,j} = \{ A \mid a_{i+1} \ldots a_j \in L_G(A) \} \]

- Time \( \Theta(n^3) \), space \( \Theta(n^2) \).
- By matrix multiplication (Valiant, 1975): time \( \Theta(n^\omega) \).
- Conjunctive, Boolean: same algorithms.
- Unambiguous: time \( O(n^2) \).
- Tree-adjoining: time \( O(n^6) \), can do in time \( O(n^{2\omega}) \).
- \( k \)-component: time \( O(n^{3k}) \), can do in time \( O(n^{\omega k}) \).
Ordinary grammars: Cocke–Kasami–Younger

Given $G$ in CNF, given $w = a_1 \ldots a_n$, construct

$$T_{i,j} = \{A \mid a_{i+1} \ldots a_j \in L_G(A)\}$$

- Time $\Theta(n^3)$, space $\Theta(n^2)$.
- By matrix multiplication (Valiant, 1975): time $\Theta(n^\omega)$.
- Conjunctive, Boolean: same algorithms.
- Unambiguous: time $O(n^2)$.
- Tree-adjoining: time $O(n^6)$, can do in time $O(n^{2\omega})$.
- $k$-component: time $O(n^{3k})$, can do in time $O(n^{\omega k})$.

Polynomial-time parsing for each family.
The hierarchy: parsing algorithms

Reg → LLLin → LL → LR → UnambLin → Unamb → Ordinary → UnambConj → UnambBool
IDPDA → LRLin → Lin → LinConj → Multi

Why is there always a polynomial-time algorithm?..
The hierarchy: parsing algorithms

- $O(n)$
- $O(n^2)$
- $O(n^2\omega)$
- $O(n^\omega)$
- $O(n^4)$
- $O(nk)$, $k > 0$
The hierarchy: parsing algorithms

- Why is there always a polynomial-time algorithm?..
Part V

General model: FO(LFP) logic
Logical contents of grammars

- In 1957: grammars as a special case of TM.
Logical contents of grammars

- In 1957: grammars as a special case of TM.
- Not useful! Cf. “remote control as magic wand”.

Examples:

\[ S(x, y) = (\exists z)(S(x, z) \land S(z, y)) \lor (a(x+1) \land S(x+1, y-1) \land b(y)) \lor x = y \]

\[ \sigma = S(begin, end) \]
Logical contents of grammars

- In 1957: grammars as a special case of TM.
- Not useful! Cf. “remote control as magic wand”.

Correct general model: FO(LFP) logic (Rounds, 1988).
Logical contents of grammars

- In 1957: grammars as a special case of TM.
- Not useful! Cf. “remote control as magic wand”.

Correct general model: FO(LFP) logic (Rounds, 1988).

- Variables ranging over positions in a string.
Logical contents of grammars

- In 1957: grammars as a special case of TM.
- Not useful! Cf. “remote control as magic wand”.

Correct general model: FO(LFP) logic (Rounds, 1988).

- Variables ranging over positions in a string.
- Recursive definitions of predicates.
Logical contents of grammars

- In 1957: grammars as a special case of TM.
- Not useful! Cf. “remote control as magic wand”.

Correct general model: FO(LFP) logic (Rounds, 1988).

- Variables ranging over positions in a string.
- Recursive definitions of predicates.

Example \((S \rightarrow SS \mid aSb \mid \varepsilon)\)

\[
S(x, y) = \bigvee \quad \bigvee
\]

\(\sigma = S(\text{begin}, \text{end})\)
Logical contents of grammars

- In 1957: grammars as a special case of TM.
- Not useful! Cf. “remote control as magic wand”.

Correct general model: FO(LFP) logic (Rounds, 1988).

- Variables ranging over positions in a string.
- Recursive definitions of predicates.

Example \((S \rightarrow SS \mid aSb \mid \varepsilon)\)

\[
S(x, y) = ((\exists z)(S(x, z) \land S(z, y)))
\]

\[
\lor \quad \lor
\]
Logical contents of grammars

- In 1957: grammars as a special case of TM.
- Not useful! Cf. “remote control as magic wand”.

Correct general model: FO(LFP) logic (Rounds, 1988).

- Variables ranging over positions in a string.
- Recursive definitions of predicates.

Example \((S \to SS \mid aSb \mid \varepsilon)\)

\[
S(x, y) = ((\exists z)(S(x, z) \land S(z, y))) \lor (a(x + 1) \land S(x + 1, y - 1) \land b(y)) \lor \\
\]
Logical contents of grammars

- In 1957: grammars as a special case of TM.
- Not useful! Cf. “remote control as magic wand”.

Correct general model: FO(LFP) logic (Rounds, 1988).

- Variables ranging over positions in a string.
- Recursive definitions of predicates.

Example \((S \rightarrow SS \mid aSb \mid \varepsilon)\)

\[
S(x, y) = ((\exists z)(S(x, z) \land S(z, y))) \\
\lor (a(x + 1) \land S(x + 1, y - 1) \land b(y)) \lor x = y
\]
Logical contents of grammars

- In 1957: grammars as a special case of TM.
- Not useful! Cf. “remote control as magic wand”.

Correct general model: FO(LFP) logic (Rounds, 1988).

- Variables ranging over positions in a string.
- Recursive definitions of predicates.

Example

\( (S \rightarrow SS \mid aSb \mid \varepsilon) \)

\[
S(x, y) = ((\exists z)(S(x, z) \land S(z, y))) \\
\lor (a(x + 1) \land S(x + 1, y - 1) \land b(y)) \lor x = y
\]

\( \sigma = S(\text{begin, end}) \)
Grammar families as fragments

- Ordinary: free $\lor$, restricted $\{\exists, \land\}$. 
Grammar families as fragments

- Ordinary: free $\lor$, restricted $\{\exists, \land\}$.
  - $\{\exists, \land\}$: in one particular way to express concatenation.

- Linear: no quantification.
- Unambiguous: one true value for $\{\lor, \exists\}$.
- LL: $A(i,j)$ uses $i$ to determine $j$, grammar is a program.
- Conjunctive: free use of $\land$.
- Boolean: eliminate negation through duality.
- $k$-component: predicates $A(x_1, y_1, ..., x_k, y_k)$.
Grammar families as fragments

- Ordinary: free \( \lor \), restricted \( \{ \exists, \land \} \).
  - \( \{ \exists, \land \} \): in one particular way to express concatenation.

- Linear: no quantification.
Grammar families as fragments

- **Ordinary**: free $\lor$, restricted $\{\exists, \land\}$.
  - $\{\exists, \land\}$: in one particular way to express concatenation.
- **Linear**: no quantification.
- **Unambiguous**: one true value for $\{\lor, \exists\}$.
Grammar families as fragments

- Ordinary: free $\lor$, restricted $\{\exists, \land\}$.
  - $\{\exists, \land\}$: in one particular way to express concatenation.
- Linear: no quantification.
- Unambiguous: one true value for $\{\lor, \exists\}$.
- LL: $A(i, j)$ uses $i$ to determine $j$. 

$A(i) = A(i, j)$, grammar is a program.
Grammar families as fragments

- Ordinary: free $\lor$, restricted $\{\exists, \land\}$.
  - $\{\exists, \land\}$: in one particular way to express concatenation.
- Linear: no quantification.
- Unambiguous: one true value for $\{\lor, \exists\}$.
- LL: $A(i, j)$ uses $i$ to determine $j$.
  - $j = A(i)$, grammar is a program.
Grammar families as fragments

- Ordinary: free $\lor$, restricted $\{\exists, \land\}$.
  - $\{\exists, \land\}$: in one particular way to express concatenation.
- Linear: no quantification.
- Unambiguous: one true value for $\{\lor, \exists\}$.
- LL: $A(i, j)$ uses $i$ to determine $j$.
  - $j = A(i)$, grammar is a program.
- Conjunctive: free use of $\land$. 

Alexander Okhotin
Principles and recurring ideas of grammars
LATA 2018
Grammar families as fragments

- Ordinary: free $\lor$, restricted $\{\exists, \land\}$.
  - $\{\exists, \land\}$: in one particular way to express concatenation.
- Linear: no quantification.
- Unambiguous: one true value for $\{\lor, \exists\}$.
- LL: $A(i, j)$ uses $i$ to determine $j$.
  - $j = A(i)$, grammar is a program.
- Conjunctive: free use of $\land$.
- Boolean: eliminate negation through duality.
Grammar families as fragments

- Ordinary: free $\lor$, restricted $\{\exists, \land\}$.
  - $\{\exists, \land\}$: in one particular way to express concatenation.
- Linear: no quantification.
- Unambiguous: one true value for $\{\lor, \exists\}$.
- LL: $A(i, j)$ uses $i$ to determine $j$.
  - $j = A(i)$, grammar is a program.
- Conjunctive: free use of $\land$.
- Boolean: eliminate negation through duality.
- $k$-component: predicates $A(x_1, y_1, \ldots x_k, y_k)$
The FO(LFP) logic

Definition (Chandra/Harel, 1980)

FO(LFP) definition: $G = (\Sigma, N, \text{dim}, \langle \varphi_A \rangle_{A \in N}, \sigma)$, where

- $\varphi_A(x_1, \ldots, x_{\text{dim} A})$ is a formula defining $A$,
- $\sigma$ is a formula with no free variables.

Semantics: same as for language equations.

Theorem (Immerman, 1986; Vardi, 1982)

$L$ defined in FO(LFP) $\iff L$ recognized in polynomial time.

Membership in P: almost Cocke–Kasami–Younger!
The FO(LFP) logic

**Definition (Chandra/Harel, 1980)**

**FO(LFP) definition:** \( G = (\Sigma, N, \text{dim}, \langle \varphi_A \rangle_{A \in N}, \sigma) \), where

- \( \varphi_A(x_1, \ldots, x_{\text{dim } A}) \) is a formula defining \( A \),
The FO(LFP) logic

Definition (Chandra/Harel, 1980)

FO(LFP) definition: \( G = (\Sigma, N, \text{dim}, \langle \varphi_A \rangle_{A \in N}, \sigma) \), where
- \( \varphi_A(x_1, \ldots, x_{\text{dim} A}) \) is a formula defining \( A \),
- \( \sigma \) is a formula with no free variables.

Semantics: same as for language equations.

Theorem (Immerman, 1986; Vardi, 1982)

Language defined in FO(LFP) \( \Leftrightarrow \) language recognized in polynomial time.

Membership in P: almost Cocke–Kasami–Younger!
The FO(LFP) logic

Definition (Chandra/Harel, 1980)

FO(LFP) definition: $G = (\Sigma, N, \text{dim}, \left\langle \varphi_A \right\rangle_{A \in N}, \sigma)$, where

- $\varphi_A(x_1, \ldots, x_{\text{dim} A})$ is a formula defining $A$,
- $\sigma$ is a formula with no free variables.

Semantics: same as for language equations.
The FO(LFP) logic

**Definition (Chandra/Harel, 1980)**

FO(LFP) definition: \( G = (\Sigma, N, \text{dim}, \langle \varphi_A \rangle_{A \in N}, \sigma) \), where

- \( \varphi_A(x_1, \ldots, x_{\text{dim}A}) \) is a formula defining \( A \),
- \( \sigma \) is a formula with no free variables.

Semantics: same as for language equations.

**Theorem (Immerman, 1986; Vardi, 1982)**

\( L \) defined in FO(LFP)

\( \Leftrightarrow \)
The FO(LFP) Logic

**Definition (Chandra/Harel, 1980)**

FO(LFP) definition:  
\[ G = (\Sigma, N, \dim, \langle \varphi_A \rangle_{A \in N}, \sigma) \], where
- \( \varphi_A(x_1, \ldots, x_{\dim A}) \) is a formula defining \( A \),
- \( \sigma \) is a formula with no free variables.

Semantics: same as for language equations.

**Theorem (Immerman, 1986; Vardi, 1982)**

\( L \) defined in FO(LFP)  
\[ \uparrow \downarrow \]  
\( L \) recognized in polynomial time.
The FO(LFP) logic

Definition (Chandra/Harel, 1980)

FO(LFP) definition: \( G = (\Sigma, N, \text{dim}, \langle \varphi_A \rangle_{A \in N}, \sigma) \), where

- \( \varphi_A(x_1, \ldots, x_{\text{dim} A}) \) is a formula defining \( A \),
- \( \sigma \) is a formula with no free variables.

Semantics: same as for language equations.

Theorem (Immerman, 1986; Vardi, 1982)

\( L \) defined in FO(LFP)

\( \Updownarrow \)

\( L \) recognized in polynomial time.

- Membership in P: almost Cocke–Kasami–Younger!
Part VI

Some theoretical results
Equivalent representations

Ordinary by pushdown automata (NPDA).

Equivalent representations

Ordinary by pushdown automata (NPDA).

- (special cases) LR: DPDA; linear: one-turn.
Equivalent representations

Ordinary by pushdown automata (NPDA).

- (special cases) LR: DPDA; linear: one-turn.
- Conjunctive: PDA with tree-like stack (Aizikowitz, Kaminski).
Equivalent representations

Ordinary by pushdown automata (NPDA).

- (special cases) LR: DPDA; linear: one-turn.
- Conjunctive: PDA with tree-like stack (Aizikowitz, Kaminski).
- Bracketed: input-driven automata.

\[(\text{John}) \land (\text{works}) \Rightarrow (\text{John works})\]
Equivalent representations

Ordinary by pushdown automata (NPDA).

- (special cases) LR: DPDA; linear: one-turn.
- Conjunctive: PDA with tree-like stack (Aizikowitz, Kaminski).
- Bracketed: input-driven automata.
- Linear conjunctive: 1W RT cellular automata.
Equivalent representations

Ordinary by pushdown automata (NPDA).

- (special cases) LR: DPDA; linear: one-turn.
- Conjunctive: PDA with tree-like stack (Aizikowitz, Kaminski).
- Bracketed: input-driven automata.
- Linear conjunctive: 1W RT cellular automata.

Ordinary by categorial grammars (Bar-Hillel et al.)
Equivalent representations

Ordinary by pushdown automata (NPDA).

- (special cases) LR: DPDA; linear: one-turn.
- Conjunctive: PDA with tree-like stack (Aizikowitz, Kaminski).
- Bracketed: input-driven automata.
- Linear conjunctive: 1W RT cellular automata.

Ordinary by categorial grammars (Bar-Hillel et al.)

\[
\begin{align*}
n(John) & \quad n \backslash s(works) \\
\Rightarrow & \\
& \quad s(John \ works)
\end{align*}
\]
Equivalent representations

Ordinary by pushdown automata (NPDA).

- (special cases) LR: DPDA; linear: one-turn.
- Conjunctive: PDA with tree-like stack (Aizikowitz, Kaminski).
- Bracketed: input-driven automata.
- Linear conjunctive: 1W RT cellular automata.

Ordinary by categorial grammars (Bar-Hillel et al.)

\[
\frac{n(\text{John})}{n\backslash s(\text{works})} \quad \frac{s(\text{John works})}{\text{Extends to conjunctive and to multi-component.}}
\]
Homomorphic characterizations

Theorem (Chomsky, Schützenberger, 1963)

Ordinary as \( h(D_k \cap R) \), for a homomorphism \( h \),
Dyck language \( D_k \), regular \( R \).
Homomorphic characterizations

Theorem (Chomsky, Schützenberger, 1963)

Ordinary as $h(D_k \cap R)$, for a homomorphism $h$, Dyck language $D_k$, regular $R$.

- Extends to multi-component.
Homomorphic characterizations

**Theorem (Chomsky, Schützenberger, 1963)**

_Ordinary as \( h(D_k \cap R) \), for a homomorphism \( h \), Dyck language \( D_k \), regular \( R \)._n

- Extends to multi-component.
- Textbook proofs use erasing \( h \).
Homomorphic characterizations

Theorem (Chomsky, Schützenberger, 1963)

Ordinary as $h(D_k \cap R)$, for a homomorphism $h$, Dyck language $D_k$, regular $R$.

- Extends to multi-component.
- Textbook proofs use erasing $h$.
- May use non-erasering (Okhotin, 2012).
Homomorphic characterizations

Theorem (Chomsky, Schützenberger, 1963)

Ordinary as $h(D_k \cap R)$, for a homomorphism $h$, Dyck language $D_k$, regular $R$.

- Extends to multi-component.
- Textbook proofs use erasing $h$.
- May use non-erasing (Okhotin, 2012).
- $k$ depends only on $|\Sigma|$ (Crespi-Reghizzi, San Pietro, 2016).
Homomorphic characterizations

Theorem (Chomsky, Schützenberger, 1963)

Ordinary as $h(D_k \cap R)$, for a homomorphism $h$, Dyck language $D_k$, regular $R$.

- Extends to multi-component.
- Textbook proofs use erasing $h$.
- May use non-erasing (Okhotin, 2012).
- $k$ depends only on $|\Sigma|$ (Crespi-Reghizzi, San Pietro, 2016).

Theorem

$L \subseteq (\Sigma^2)^*$ as $h(D_k \cap R)$, with length-preserving $h$.
Hardest languages

Theorem (Greibach, 1973)

∃ “hardest” ordinary grammar $G_0 = (\Sigma_0, N_0, R_0, S_0)$, s.t. for every $G$ there is $h$: 
$w \in L(G)$ iff $h(w) \in L(G_0)$. 

Extends to conjunctive, Boolean.
Does not extend to LR (Greibach).
Does not extend to linear (Boasson/Nivat).
To unambiguous? To linear conjunctive?
Hardest languages

**Theorem (Greibach, 1973)**

∃ “hardest” ordinary grammar $G_0 = (\Sigma_0, N_0, R_0, S_0)$, s.t. for every $G$ there is $h$:

$w \in L(G)$ iff $h(w) \in L(G_0)$.

- Extends to conjunctive, Boolean.
Hardest languages

**Theorem (Greibach, 1973)**

∃ “hardest” ordinary grammar $G_0 = (\Sigma_0, N_0, R_0, S_0)$,
s.t. for every $G$ there is $h$:

$w \in L(G)$ iff $h(w) \in L(G_0)$.

- Extends to conjunctive, Boolean.
- Does not extend to LR (Greibach).
Hardest languages

Theorem (Greibach, 1973)

\[ \exists \text{ "hardest" ordinary grammar } G_0 = (\Sigma_0, N_0, R_0, S_0), \text{ s.t. for every } G \text{ there is } h: \]
\[ w \in L(G) \text{ iff } h(w) \in L(G_0). \]

- Extends to conjunctive, Boolean.
- Does not extend to LR (Greibach).
- Does not extend to linear (Boasson/Nivat).
Theorem (Greibach, 1973)

∃ “hardest” ordinary grammar $G_0 = (\Sigma_0, N_0, R_0, S_0)$, s.t. for every $G$ there is $h$:

$$w \in L(G) \text{ iff } h(w) \in L(G_0).$$

- Extends to conjunctive, Boolean.
- Does not extend to LR (Greibach).
- Does not extend to linear (Boasson/Nivat).
- To unambiguous? To linear conjunctive?
The hierarchy: complexity

Complete sets for L, NL, P. Ordinary in NC²: depth O((log n)²), with O(n⁶) gates.

Alexander Okhotin
Principles and recurring ideas of grammars
LATA 2018 38 / 41
The hierarchy: complexity

Complete sets for L, NL, P.
Ordinary in NC²: depth $O((\log n)^2)$, with $O(n^6)$ gates.
Complete sets for \( L, NL, P \).
Complete sets for L, NL, P.

Ordinary in $NC^2$:
depth $O((\log n)^2)$, with $O(n^6)$ gates.
Part VII

Conclusion
Towards further models

Desired properties

Necessary syntactic structures expressed.
Expressed in an intuitive way!
Algorithms for efficient processing.

Further fragments of FO(LFP)?
▶ E.g., "grammars with contexts" (Barash, Okhotin, 2014), rules such as $A \rightarrow BC \& D$.

New logical foundation?
▶ Models from descriptive complexity (Immerman).

Understanding the existing models: any negative methods for conjunctive grammars?
Towards further models

Desired properties

1. Necessary syntactic structures expressed.
Towards further models

Desired properties

1. Necessary syntactic structures expressed.
2. Expressed in an intuitive way!
Towards further models

Desired properties

1. Necessary syntactic structures expressed.
2. Expressed in an intuitive way!
3. Algorithms for efficient processing.
Towards further models

Desired properties

1. Necessary syntactic structures expressed.
2. Expressed in an intuitive way!
3. Algorithms for efficient processing.

Further fragments of FO(LFP)?
Towards further models

Desired properties

1. Necessary syntactic structures expressed.
2. Expressed in an intuitive way!
3. Algorithms for efficient processing.

- Further fragments of FO(LFP)?
  - E.g., “grammars with contexts” (Barash, Okhotin, 2014), rules such as $A \rightarrow BC \land \lhd D$. 
Towards further models

Desired properties

1. Necessary syntactic structures expressed.
2. Expressed in an intuitive way!
3. Algorithms for efficient processing.

- Further fragments of FO(LFP)?
  - E.g., “grammars with contexts” (Barash, Okhotin, 2014), rules such as $A \rightarrow BC \& \triangleleft D$.

- New logical foundation?
Towards further models

Desired properties

1. Necessary syntactic structures expressed.
2. Expressed in an intuitive way!
3. Algorithms for efficient processing.

- Further fragments of FO(LFP)?
  - E.g., “grammars with contexts” (Barash, Okhotin, 2014), rules such as \( A \rightarrow BC \land \triangleleft D \).
- New logical foundation?
  - Models from *descriptive complexity* (Immerman).
Towards further models

Desired properties

1. Necessary syntactic structures expressed.
2. Expressed in an intuitive way!
3. Algorithms for efficient processing.

• Further fragments of FO(LFP)?
  ▶ E.g., “grammars with contexts” (Barash, Okhotin, 2014), rules such as $A \rightarrow BC \& \triangleleft D$.

• New logical foundation?
  ▶ Models from descriptive complexity (Immerman).

• Understanding the existing models: any negative methods for conjunctive grammars?