Formal grammars and parsing: lecture 1

Alexander Okhotin

Department of Mathematics, University of Turku; Academy of Finland

17 November 2009 A. D.
Syntax as such

- Information presented as strings of symbols.
Syntax as such

- Information presented as strings of symbols.
- Syntax of artificial languages.
Syntax as such

- Information presented as strings of symbols.
- Syntax of artificial languages.
- Syntax of natural languages.
Part I

Towards a model of syntax
Context-free grammars

Example (Balanced brackets)

\[ S \rightarrow SS \mid aSb \mid \varepsilon \]
Context-free grammars

Example (Balanced brackets)

$$S \rightarrow SS \mid aSb \mid \varepsilon$$

The most obvious formal model of syntax:
Context-free grammars

Example (Balanced brackets)

\[ S \rightarrow SS \mid aSb \mid \varepsilon \]

The most obvious formal model of syntax:

- Used in Panini’s grammar (ca. 5th century B. C.).
Context-free grammars

Example (Balanced brackets)

\[ S \rightarrow SS \mid aSb \mid \varepsilon \]

The most obvious formal model of syntax:

- Used in Panini’s grammar (ca. 5th century B. C.).
- Rediscovered by Chomsky (1957)
Example (Balanced brackets)

\[ S \rightarrow SS \mid aSb \mid \varepsilon \]

The most obvious formal model of syntax:
- Used in Panini’s grammar (ca. 5th century B.C.).
- Rediscovered by Chomsky (1957)
- Rediscovered by the Algol 60 committee (ca. 1960).
Context-free grammars

Example (Balanced brackets)

\[ S \rightarrow SS \mid aSb \mid \varepsilon \]

The most obvious formal model of syntax:

- Used in Panini’s grammar (ca. 5th century B.C.).
- Rediscovered by Chomsky (1957)
- Rediscovered by the Algol 60 committee (ca. 1960).
- Its mathematical study: formal language theory.
Limitations of context-free grammars

- Cannot specify \( \{ a^n b^n c^n \mid n \geq 0 \} \).

Not enough for the programming languages (Floyd, 1962):

```plaintext
main()

int x ....
x \rightarrow \phi \phi \phi
i \geq 1
x ....
x \rightarrow \phi \phi \phi
j \geq 1
x ....
x \rightarrow \phi \phi \phi
k \geq 1
```

- Cannot specify

\( \{ wcw \mid w \in \{ a, b \}^* \} \).

Identifier checking.

- Cannot specify

\( \{ a^m b^n c^m d^n \mid m, n \geq 0 \} \), etc.
Limitations of context-free grammars

- Cannot specify \( \{a^n b^n c^n \mid n \geq 0\} \).
- Not enough for the programming languages (Floyd, 1962):
Limitations of context-free grammars

- Cannot specify \( \{ a^n b^n c^n | n \geq 0 \} \).
- Not enough for the programming languages (Floyd, 1962):

\[
\text{main()} \{ \text{ int } \underbrace{x \ldots x}; \underbrace{x \ldots x} = \underbrace{x \ldots x}; \} \\
\begin{array}{c}
i \geq 1 \\
j \geq 1 \\
k \geq 1
\end{array}
\]
Limitations of context-free grammars

- Cannot specify \( \{ a^n b^n c^n \mid n \geq 0 \} \).
- Not enough for the programming languages (Floyd, 1962):

  ```
  main() { int x...x; x...x = x...x; }
  ```

  \( i \geq 1 \quad j \geq 1 \quad k \geq 1 \)

- Cannot specify \( \{ wcw \mid w \in \{a, b\}^* \} \).
Limitations of context-free grammars

- Cannot specify \( \{a^n b^n c^n \mid n \geq 0\} \).
- Not enough for the programming languages (Floyd, 1962):

\[
\text{main()} \{ \text{int } \underbrace{\text{x...x};}_{i \geq 1} \underbrace{\text{x...x} = \text{x...x};}_{j \geq 1} \underbrace{\text{x...x}}_{k \geq 1} \}
\]

- Cannot specify \( \{wcw \mid w \in \{a, b\}^*\} \).
  
  ▶ Identifier checking.
Limitations of context-free grammars

- Cannot specify \( \{ a^n b^n c^n \mid n \geq 0 \} \).
- Not enough for the programming languages (Floyd, 1962):
  
  ```
  main() \{ int \underbrace{x \ldots x}; \underbrace{x \ldots x} = \underbrace{x \ldots x}; \}
  \quad i \geq 1 \quad j \geq 1 \quad k \geq 1
  ```

- Cannot specify \( \{ wcw \mid w \in \{a, b\}^* \} \).
  - Identifier checking.
- Cannot specify \( \{ a^m b^n c^m d^n \mid m, n \geq 0 \} \), etc.
Two definitions of context-free grammars

Consider the grammar

\[ S \to aSb \mid SS \mid \varepsilon \]

By derivation (Chomsky, 1957)

By language equations (Ginsburg and Rice, 1962)

\[ S = \{a\}S\{b\} \cup SS \cup \{\varepsilon\} \]

\(w\) has property \(S \uparrow w = au \uparrow b\), where \(u\) has the property \(S\), or \(w = uv\), where \(u\) and \(v\) have the property \(S\), or \(w = \varepsilon\).

What makes context-free grammars natural?

Is there anything missing in these definitions?
Two definitions of context-free grammars

Consider the grammar

\[ S \rightarrow aSb \mid SS \mid \varepsilon \]

By derivation (Chomsky, 1957)

\[ \alpha S \beta \text{ derives } \alpha aSb\beta; \]

By language equations (Ginsburg and Rice, 1962)

\[ S = \{ a \} S \{ b \} \cup SS \cup \{ \varepsilon \} \]

\( w \) has property

\[ S \leftrightarrow w = aub, \text{ where } u \text{ has the property } S, \text{ or } w = uv, \text{ where } u \text{ and } v \text{ have the property } S, \text{ or } w = \varepsilon. \]
Two definitions of context-free grammars

Consider the grammar

\[ S \rightarrow aSb \mid SS \mid \varepsilon \]

By derivation (Chomsky, 1957)

\[ \alpha S\beta \text{ derives } \alpha aSb\beta; \]
\[ \alpha S\beta \text{ derives } \alpha SS\beta; \]

By language equations (Ginsburg and Rice, 1962)

\[ S = \{a\} S \{b\} \cup SS \cup \{\varepsilon\} \]

What makes context-free grammars natural?

Is there anything missing in these definitions?
Two definitions of context-free grammars

Consider the grammar

\[ S \rightarrow aSb \mid SS \mid \varepsilon \]

By derivation (Chomsky, 1957)

\[ \alpha S \beta \text{ derives } \alpha aSb \beta; \]
\[ \alpha S \beta \text{ derives } \alpha SS \beta; \]
\[ \alpha S \beta \text{ derives } \alpha \beta. \]
Two definitions of context-free grammars

Consider the grammar\[ S \rightarrow aSb \mid SS \mid \varepsilon \]

By derivation (Chomsky, 1957)
\[ \alpha S \beta \text{ derives } \alpha aSb\beta; \]
\[ \alpha S \beta \text{ derives } \alpha SS\beta; \]
\[ \alpha S \beta \text{ derives } \alpha \beta. \]

By language equations (Ginsburg and Rice, 1962)

What makes context-free grammars natural?
Is there anything missing in these definitions?
Two definitions of context-free grammars

Consider the grammar

\[ S \rightarrow aSb \mid SS \mid \varepsilon \]

By derivation (Chomsky, 1957)

\[ \alpha S \beta \text{ derives } \alpha aSb \beta; \]
\[ \alpha S \beta \text{ derives } \alpha SS \beta; \]
\[ \alpha S \beta \text{ derives } \alpha \beta. \]

By language equations (Ginsburg and Rice, 1962)

\[ S = \{a\}S\{b\} \cup SS \cup \{\varepsilon\} \]
Two definitions of context-free grammars

Consider the grammar

\[
S \rightarrow aSb \mid SS \mid \varepsilon
\]

By derivation (Chomsky, 1957)

\[
\begin{align*}
\alpha S \beta & \text{ derives } \alpha aSb \beta; \\
\alpha S \beta & \text{ derives } \alpha SS \beta; \\
\alpha S \beta & \text{ derives } \alpha \beta.
\end{align*}
\]

By language equations (Ginsburg and Rice, 1962)

\[
S = \{a\}S\{b\} \cup SS \cup \{\varepsilon\}
\]

\(w\) has property \(S\)

\[
\uparrow
\]
Two definitions of context-free grammars

Consider the grammar

\[ S \rightarrow aSb \mid SS \mid \varepsilon \]

By derivation (Chomsky, 1957)

\[ \alpha S \beta \text{ derives } \alpha aSb\beta; \]
\[ \alpha S \beta \text{ derives } \alpha SS\beta; \]
\[ \alpha S \beta \text{ derives } \alpha \beta. \]

By language equations (Ginsburg and Rice, 1962)

\[ S = \{a\}S\{b\} \cup SS \cup \{\varepsilon\} \]

\( w \) has property \( S \)

\[ w = aub, \text{ where } u \text{ has the property } S, \quad \text{or} \]
Two definitions of context-free grammars

Consider the grammar $S \rightarrow aSb | SS | \varepsilon$

By derivation (Chomsky, 1957)

$\alpha S \beta$ derives $\alpha aSb \beta$;
$\alpha S \beta$ derives $\alpha SS \beta$;
$\alpha S \beta$ derives $\alpha \beta$.

By language equations (Ginsburg and Rice, 1962)

$S = \{a\}S\{b\} \cup SS \cup \{\varepsilon\}$

$w$ has property $S$

$\uparrow$

$w = aub$, where $u$ has the property $S$, or $w = uv$, where $u$ and $v$ have the property $S$, or
Two definitions of context-free grammars

Consider the grammar

\[ S \rightarrow aSb \mid SS \mid \varepsilon \]

By derivation (Chomsky, 1957)

\[ \alpha S \beta \text{ derives } \alpha aSb\beta; \]
\[ \alpha S \beta \text{ derives } \alpha SS\beta; \]
\[ \alpha S \beta \text{ derives } \alpha \beta. \]

By language equations (Ginsburg and Rice, 1962)

\[ S = \{a\}S\{b\} \cup SS \cup \{\varepsilon\} \]

\( w \) has property \( S \)

\[ \uparrow \]

\( w = aub \), where \( u \) has the property \( S \),  \text{ or } \]
\( w = uv \), where \( u \) and \( v \) have the property \( S \),  \text{ or } \]
\( w = \varepsilon. \)
Two definitions of context-free grammars

Consider the grammar

\[ S \rightarrow aSb \mid SS \mid \varepsilon \]

By derivation (Chomsky, 1957)

\[ \alpha S \beta \text{ derives } \alpha aSb \beta; \]
\[ \alpha S \beta \text{ derives } \alpha SS \beta; \]
\[ \alpha S \beta \text{ derives } \alpha \beta. \]

By language equations (Ginsburg and Rice, 1962)

\[ S = \{a\}S\{b\} \cup SS \cup \{\varepsilon\} \]

\[ w \text{ has property } S \]

\[ \uparrow \]

\[ w = aub, \text{ where } u \text{ has the property } S, \text{ or } \]
\[ w = uv, \text{ where } u \text{ and } v \text{ have the property } S, \text{ or } \]
\[ w = \varepsilon. \]

What makes context-free grammars natural?
Two definitions of context-free grammars

Consider the grammar

\[ S \rightarrow aSb \mid SS \mid \varepsilon \]

By derivation (Chomsky, 1957)

\[ \alpha S\beta \text{ derives } \alpha aSb\beta; \]
\[ \alpha S\beta \text{ derives } \alpha SS\beta; \]
\[ \alpha S\beta \text{ derives } \alpha\beta. \]

By language equations (Ginsburg and Rice, 1962)

\[ S = \{a\}S\{b\} \cup SS \cup \{\varepsilon\} \]

\( w \) has property \( S \)

\[ \updownarrow \]

\( w = aub \), where \( u \) has the property \( S \), \text{ or } \]
\( w = uv \), where \( u \) and \( v \) have the property \( S \), \text{ or } \]
\( w = \varepsilon. \)

- What makes context-free grammars natural?
- Is there anything missing in these definitions?
Augmenting context-free derivation

Derivation: operational semantics of context-free grammars.

<table>
<thead>
<tr>
<th>Production</th>
<th>Derivation of</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S \rightarrow aSb$</td>
<td>$\alpha S \beta \ derives \ \alpha aSb \beta$</td>
</tr>
<tr>
<td>$S \rightarrow SS$</td>
<td>$\alpha S \beta \ derives \ \alpha SS \beta$</td>
</tr>
<tr>
<td>$S \rightarrow \varepsilon$</td>
<td>$\alpha S \beta \ derives \ \alpha \beta$</td>
</tr>
</tbody>
</table>
Augmenting context-free derivation

Derivation: operational semantics of context-free grammars.

\[
\begin{array}{|l|l|}
\hline
S \rightarrow aSb & \alpha S \beta \text{ derives } \alpha aSb \beta \\
S \rightarrow SS & \alpha S \beta \text{ derives } \alpha SS \beta \\
S \rightarrow \varepsilon & \alpha S \beta \text{ derives } \alpha \beta \\
\hline
\end{array}
\]

Several natural extensions:
Augmenting context-free derivation

Derivation: operational semantics of context-free grammars.

<table>
<thead>
<tr>
<th>$S \rightarrow aSb$</th>
<th>$\alpha S \beta$ derives $\alpha aSb \beta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S \rightarrow SS$</td>
<td>$\alpha S \beta$ derives $\alpha SS \beta$</td>
</tr>
<tr>
<td>$S \rightarrow \varepsilon$</td>
<td>$\alpha S \beta$ derives $\alpha \beta$</td>
</tr>
</tbody>
</table>

Several natural extensions:

- Chomsky’s *context-sensitive grammars*. 
Augmenting context-free derivation

Derivation: operational semantics of context-free grammars.

<table>
<thead>
<tr>
<th>Rule</th>
<th>αSβ derives αaSbβ \nS → aSb</th>
<th>αSβ derives αSSβ \nS → SS</th>
<th>αSβ derives αβ \nS → ε</th>
</tr>
</thead>
</table>

Several natural extensions:

- Chomsky’s *context-sensitive grammars*.
- A reformulation of $\text{NSPACE}(n)$.
Augmenting context-free derivation

Derivation: operational semantics of context-free grammars.

\[
\begin{array}{|c|c|}
\hline
S \rightarrow aSb & \alpha S\beta \text{ derives } \alpha aSb\beta \\
S \rightarrow SS & \alpha S\beta \text{ derives } \alpha SS\beta \\
S \rightarrow \varepsilon & \alpha S\beta \text{ derives } \alpha \beta \\
\hline
\end{array}
\]

Several natural extensions:

- Chomsky’s context-sensitive grammars.
- A reformulation of $\text{NSPACE}(n)$.
- Theoretically and historically very important, but...
Augmenting context-free derivation

Derivation: operational semantics of context-free grammars.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Derivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S \rightarrow aSb$</td>
<td>$\alpha S\beta \text{ derives } \alpha aSb\beta$</td>
</tr>
<tr>
<td>$S \rightarrow SS$</td>
<td>$\alpha S\beta \text{ derives } \alpha SS\beta$</td>
</tr>
<tr>
<td>$S \rightarrow \epsilon$</td>
<td>$\alpha S\beta \text{ derives } \alpha \beta$</td>
</tr>
</tbody>
</table>

Several natural extensions:

- Chomsky’s *context-sensitive grammars*.
- A reformulation of $\text{NSPACE}(n)$.
- Theoretically and historically very important, but . . .
- Hardly a model of syntax.
Augmenting context-free derivation

Derivation: operational semantics of context-free grammars.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Derivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S \rightarrow aSb$</td>
<td>$\alpha S \beta$ derives $\alpha aSb \beta$</td>
</tr>
<tr>
<td>$S \rightarrow SS$</td>
<td>$\alpha S \beta$ derives $\alpha SS \beta$</td>
</tr>
<tr>
<td>$S \rightarrow \varepsilon$</td>
<td>$\alpha S \beta$ derives $\alpha \beta$</td>
</tr>
</tbody>
</table>

Several natural extensions:

- Chomsky’s context-sensitive grammars.
- A reformulation of $\text{NSPACE}(n)$.
- Theoretically and historically very important, but...
- Hardly a model of syntax.

Numerous artificial extensions.
Augmenting context-free derivation

Derivation: operational semantics of context-free grammars.

<table>
<thead>
<tr>
<th>$S \rightarrow aSb$</th>
<th>$\alpha S \beta$ derives $\alpha aSb \beta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S \rightarrow SS$</td>
<td>$\alpha S \beta$ derives $\alpha SS \beta$</td>
</tr>
<tr>
<td>$S \rightarrow \epsilon$</td>
<td>$\alpha S \beta$ derives $\alpha \beta$</td>
</tr>
</tbody>
</table>

Several natural extensions:

- Chomsky’s context-sensitive grammars.
- A reformulation of $\text{NSPACE}(n)$.
- Theoretically and historically very important, but...
- Hardly a model of syntax.

Numerous artificial extensions.

- May be challenging mathematically, but...
Augmenting context-free derivation

Derivation: operational semantics of context-free grammars.

<table>
<thead>
<tr>
<th>Derivation</th>
<th>Derives</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S \rightarrow aSb$</td>
<td>$\alpha S \beta$ derives $\alpha aSb \beta$</td>
</tr>
<tr>
<td>$S \rightarrow SS$</td>
<td>$\alpha S \beta$ derives $\alpha SS \beta$</td>
</tr>
<tr>
<td>$S \rightarrow \varepsilon$</td>
<td>$\alpha S \beta$ derives $\alpha \beta$</td>
</tr>
</tbody>
</table>

Several natural extensions:

- Chomsky’s context-sensitive grammars.
- A reformulation of $\text{NSPACE}(n)$.
- Theoretically and historically very important, but...
- Hardly a model of syntax.

Numerous artificial extensions.

- May be challenging mathematically, but...
- Hardly a model of anything.
Logical content of context-free grammars

$w$ has property $S$

$w = aub$, where $u$ has the property $S$

$w = uv$, where $u$ and $v$ have the property $S$

$w = \varepsilon$
Logical content of context-free grammars

$w$ has property $S$ \iff

$w = aub$, where $u$ has the property $S$

$w = uv$, where $u$ and $v$ have the property $S$

$w = \varepsilon$

- Multiple rules for a nonterminal represent disjunction.
Logical content of context-free grammars

- $w$ has property $S$ if
  - $w = aub$, where $u$ has the property $S$
  - $w = uv$, where $u$ and $v$ have the property $S$
  - $w = \varepsilon$

- Multiple rules for a nonterminal represent disjunction.
- Where are the conjunction and the negation?
Multiple rules for a nonterminal represent disjunction.

Where are the conjunction and the negation?

Essential for syntax: “$w$ satisfies both conditions $A$ and $B$”
Finite intersections of context-free languages:
Finite intersections of context-free languages:


Wotschke (1973, 1978): \( \{wcw \mid w \in \{a, b\}^*\} \) non-representable.


Free use of Boolean operations within grammars:

Okhotin (2000): "Conjunctive grammars".

Okhotin (2003): "Boolean grammars".
Finite intersections of context-free languages:

- Wotschke (1973, 1978): \( \{ wcw \mid w \in \{a, b\}^* \} \) non-representable.
Finite intersections of context-free languages:

- Wotschke (1973, 1978): \( \{wcw \mid w \in \{a, b\}^*\} \) non-representable.
Context-free grammars and Boolean operations

Finite intersections of context-free languages:

- Wotschke (1973, 1978): \( \{ wcw \mid w \in \{ a, b \}^* \} \) non-representable.
Context-free grammars and Boolean operations

Finite intersections of context-free languages:

- Wotschke (1973, 1978): \( \{ wcw \mid w \in \{a, b\}^* \} \) non-representable.

Free use of Boolean operations within grammars:
Finite intersections of context-free languages:
- Wotschke (1973, 1978): \( \{wcw \mid w \in \{a, b\}^*\} \) non-representable.

Free use of Boolean operations within grammars:
Finite intersections of context-free languages:
- Wotschke (1973, 1978): $\{wcw \mid w \in \{a, b\}^*\}$ non-representable.

Free use of Boolean operations within grammars:

Context-free grammars and Boolean operations
Part II

Conjunctive and Boolean grammars
Intuitive definitions

Conjunctive grammar: quadruple $G = (\Sigma, N, P, S)$, where rules in $P$ are of the form

$$A \rightarrow \alpha_1 \& \ldots \& \alpha_m$$
Intuitive definitions

**Conjunctive grammar:** quadruple \( G = (\Sigma, N, P, S) \),

where rules in \( P \) are of the form

\[ A \rightarrow \alpha_1 \& \ldots \& \alpha_m \]

“\( w \) is generated by each \( \alpha_i \), then \( w \) is generated by \( A \)”.

**Boolean grammar:** quadruple \( G = (\Sigma, N, P, S) \),

with rules of the form

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

“\( w \) is generated by each \( \alpha_i \) and by none of \( \beta_j \), then \( w \) is generated by \( A \)”.
Intuitive definitions

Conjunctive grammar: quadruple $G = (\Sigma, N, P, S)$,
where rules in $P$ are of the form

$$A \rightarrow \alpha_1 & \ldots & \alpha_m$$

“If $w$ is generated by each $\alpha_i$, then $w$ is generated by $A$”.

Boolean grammar: quadruple $G = (\Sigma, N, P, S)$,
with rules of the form

$$A \rightarrow \alpha_1 & \ldots & \alpha_m & \neg \beta_1 & \ldots & \neg \beta_n$$
Intuitive definitions

Conjunctive grammar: quadruple $G = (\Sigma, N, P, S)$,
where rules in $P$ are of the form

$$A \rightarrow \alpha_1 \& \ldots \& \alpha_m$$

“If $w$ is generated by each $\alpha_i$, then $w$ is generated by $A$”.

Boolean grammar: quadruple $G = (\Sigma, N, P, S)$,
with rules of the form

$$A \rightarrow \alpha_1 \& \ldots \& \alpha_m \& \neg \beta_1 \& \ldots \& \neg \beta_n$$

“If $w$ is generated by each $\alpha_i$ and by none of $\beta_j$, then $w$ is generated by $A$”.
Conjunctive grammars: semantics by derivation

Rewriting terms over concatenation and conjunction:

\[ \ldots A \ldots \rightarrow \ldots (\alpha_1 \& \ldots \& \alpha_m) \ldots \]
Conjunctive grammars: semantics by derivation

Rewriting terms over concatenation and conjunction:

- \( \ldots A \ldots \Rightarrow \ldots (\alpha_1 & \ldots & \alpha_m) \ldots \)
- \( \ldots (w & \ldots & w) \ldots \Rightarrow \ldots w \ldots \)
Conjunctive grammars: semantics by derivation

Rewriting terms over concatenation and conjunction:

- \( \ldots A \ldots \Rightarrow \ldots (\alpha_1 \& \ldots \& \alpha_m) \ldots \)
- \( \ldots (w \& \ldots \& w) \ldots \Rightarrow \ldots w \ldots \)

Example

A conjunctive grammar for \( \{ a^n b^n c^n \mid n \geq 0 \} \):

\[
S \rightarrow AB \& DC \\
A \rightarrow aA | \varepsilon \\
B \rightarrow bBc | \varepsilon \\
C \rightarrow cC | \varepsilon \\
D \rightarrow aDb | \varepsilon \\
S = \Rightarrow (AB \& DC) = \Rightarrow (aAB \& DC) = \Rightarrow (aB \& DC) = \Rightarrow \ldots = \Rightarrow (abc \& DC) = \Rightarrow \ldots = \Rightarrow (abc \& abc)
\]
Conjunctive grammars: semantics by derivation

Rewriting terms over concatenation and conjunction:

- \( \ldots A \ldots \Rightarrow \ldots (\alpha_1 \& \ldots \& \alpha_m) \ldots \)
- \( \ldots (w \& \ldots \& w) \ldots \Rightarrow \ldots w \ldots \)

**Example**

A conjunctive grammar for \( \{ a^n b^n c^n \mid n \geq 0 \} \):

\[
\begin{align*}
S & \rightarrow AB\&DC \\
A & \rightarrow aA \mid \varepsilon \\
B & \rightarrow bBc \mid \varepsilon \\
C & \rightarrow cC \mid \varepsilon \\
D & \rightarrow aDb \mid \varepsilon
\end{align*}
\]
Conjunctive grammars: semantics by derivation

Rewriting terms over concatenation and conjunction:

- ... $A$ ... $\Rightarrow$ ... $(\alpha_1 \& \cdots \& \alpha_m)$ ...
- ... $(w \& \cdots \& w)$ ... $\Rightarrow$ ... $w$ ...

Example

A conjunctive grammar for $\{a^n b^n c^n \mid n \geq 0\}$:

$$
S \rightarrow AB \& DC \\
A \rightarrow aA \mid \varepsilon \\
B \rightarrow bBc \mid \varepsilon \\
C \rightarrow cC \mid \varepsilon \\
D \rightarrow aDb \mid \varepsilon
$$
Conjunctive grammars: semantics by derivation

Rewriting terms over concatenation and conjunction:

- \( \ldots A \ldots \implies \ldots (\alpha_1 \& \ldots \& \alpha_m) \ldots \)
- \( \ldots (w \& \ldots \& w) \ldots \implies \ldots w \ldots \)

**Example**

A conjunctive grammar for \( \{ a^n b^n c^n \mid n \geq 0 \} \):

\[
S \rightarrow AB \& DC \\
A \rightarrow aA \mid \epsilon \\
B \rightarrow bBc \mid \epsilon \\
C \rightarrow cC \mid \epsilon \\
D \rightarrow aDb \mid \epsilon
\]

\[
S \implies (AB \& DC) \implies (aAB \& DC)
\]
Conjunctive grammars: semantics by derivation

Rewriting terms over concatenation and conjunction:

- \[ ... A ... \rightarrow ... (\alpha_1 \& \ldots \& \alpha_m) ... \]
- \[ ... (w \& \ldots \& w) ... \rightarrow ... w ... \]

Example

A conjunctive grammar for \( \{ a^n b^n c^n \mid n \geq 0 \} \):

\[
\begin{align*}
S & \rightarrow AB\&DC & S & \rightarrow (AB\&DC) \rightarrow \\
A & \rightarrow aA \mid \varepsilon & (aA) & \rightarrow (aAB\&DC) \rightarrow \\
B & \rightarrow bBc \mid \varepsilon & (aB\&DC) & \rightarrow \ldots \rightarrow \\
C & \rightarrow cC \mid \varepsilon & (abc\&DC) & \\
D & \rightarrow aDb \mid \varepsilon
\end{align*}
\]
Conjunctive grammars: semantics by derivation

Rewriting terms over concatenation and conjunction:

- \[ \cdots A \cdots \implies \cdots (\alpha_1 \& \cdots \& \alpha_m) \cdots \]
- \[ \cdots (w \& \cdots \& w) \cdots \implies \cdots w \cdots \]

Example

A conjunctive grammar for \( \{a^n b^n c^n \mid n \geq 0\} \):

\[
S \rightarrow AB\&DC
\]
\[
A \rightarrow aA \mid \varepsilon
\]
\[
B \rightarrow bBc \mid \varepsilon
\]
\[
C \rightarrow cC \mid \varepsilon
\]
\[
D \rightarrow aDb \mid \varepsilon
\]

\[
S \implies (AB\&DC) \implies
\]
\[
(aAB\&DC) \implies
\]
\[
(aB\&DC) \implies \cdots \implies
\]
\[
(abc\&DC) \implies \cdots \implies
\]
\[
(abc\&abc)
\]
Conjunctive grammars: semantics by derivation

Rewriting terms over concatenation and conjunction:

- ... $A$ ... $\rightarrow$ ... $(\alpha_1 \& \ldots \& \alpha_m)$ ...
- ... $(w \& \ldots \& w)$ ... $\rightarrow$ ... $w$ ...

Example

A conjunctive grammar for $\{a^n b^n c^n \mid n \geq 0\}$:

- $S \rightarrow AB\&\overline{DC}$
- $A \rightarrow aA \mid \varepsilon$
- $B \rightarrow bBc \mid \varepsilon$
- $C \rightarrow cC \mid \varepsilon$
- $D \rightarrow aDb \mid \varepsilon$

$LHS \rightarrow RHS$:

- $S \Rightarrow (AB\&\overline{DC})$ $\Rightarrow$
- $(aAB\&\overline{DC})$ $\Rightarrow$
- $(aB\&\overline{DC})$ $\Rightarrow$ ... $\Rightarrow$
- $(abc\&\overline{DC})$ $\Rightarrow$ ... $\Rightarrow$
- $(abc\&abc)$ $\Rightarrow$ $abc$
Conjunctive grammars: semantics by derivation

Rewriting terms over concatenation and conjunction:

- \[ \ldots A \ldots \Rightarrow \ldots (\alpha_1 \& \ldots \& \alpha_m) \ldots \]
- \[ \ldots (w \& \ldots \& w) \ldots \Rightarrow \ldots w \ldots \]

Example

A conjunctive grammar for \( \{ a^n b^n c^n \mid n \geq 0 \} \):

\[
\begin{align*}
S & \rightarrow AB\&DC \\
A & \rightarrow aA \mid \varepsilon \\
B & \rightarrow bBc \mid \varepsilon \\
C & \rightarrow cC \mid \varepsilon \\
D & \rightarrow aDb \mid \varepsilon \\
S & \Rightarrow (AB\&DC) \Rightarrow (aAB\&DC) \Rightarrow (aB\&DC) \Rightarrow \ldots \Rightarrow (abc\&DC) \Rightarrow \ldots \Rightarrow (abc\&abc) \Rightarrow abc
\end{align*}
\]
Conjunctive grammars: semantics by language equations

<table>
<thead>
<tr>
<th>grammar</th>
<th>system of equations</th>
<th>least solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S \rightarrow AB&amp;DC$</td>
<td>$S = AB \cap DC$</td>
<td>${a^n b^n c^n \mid n \geq 0}$</td>
</tr>
<tr>
<td>$A \rightarrow aA \mid \varepsilon$</td>
<td>$A = {a}A \cup {\varepsilon}$</td>
<td>$a^*$</td>
</tr>
<tr>
<td>$B \rightarrow bBc \mid \varepsilon$</td>
<td>$B = {b}B{c} \cup {\varepsilon}$</td>
<td>${b^k c^k \mid k \geq 0}$</td>
</tr>
<tr>
<td>$C \rightarrow cC \mid \varepsilon$</td>
<td>$C = {c}C \cup {\varepsilon}$</td>
<td>$c^*$</td>
</tr>
<tr>
<td>$D \rightarrow aDb \mid \varepsilon$</td>
<td>$D = {a}D{b} \cup {\varepsilon}$</td>
<td>${a^\ell b^\ell \mid \ell \geq 0}$</td>
</tr>
</tbody>
</table>

- The two semantics are equivalent.
In practical cases, unique solution of language equations. But:

- Logical contradiction is expressible.

What should a grammar $S \rightarrow \neg S$ mean?

What about the following grammar?

$S \rightarrow \neg AA \rightarrow A$?

One approach: declare both ill-formed.

Alternative (Kountouriotis et al., 2006): semantics in three-valued languages.
Semantics of Boolean grammars

- In practical cases, unique solution of language equations. But:
- Logical contradiction is expressible.

What should a grammar $S \rightarrow \neg S$ mean?

What about the following grammar?

$S \rightarrow \neg AA \rightarrow A$

One approach: declare both ill-formed.

Alternative (Kountouriotis et al., 2006): semantics in three-valued languages.
Semantics of Boolean grammars

• In practical cases, unique solution of language equations. But:
  ✓ Logical contradiction is expressible.
  • What should a grammar $S \rightarrow \neg S$ mean?

One approach: declare both ill-formed.
Alternative (Kountouriotis et al., 2006): semantics in three-valued languages.
In practical cases, unique solution of language equations. But:

✓ Logical contradiction is expressible.

What should a grammar $S \rightarrow \neg S$ mean?

What about the following grammar?

$$S \rightarrow \neg AA \rightarrow A$$
Semantics of Boolean grammars

- In practical cases, unique solution of language equations. But:
  - Logical contradiction is expressible.
  - What should a grammar $S \rightarrow \neg S$ mean?
  - What about the following grammar?

\[ S \rightarrow \neg AA \rightarrow A \]

- One approach: declare both ill-formed.
Semantics of Boolean grammars

- In practical cases, unique solution of language equations. But:
  - Logical contradiction is expressible.
  - What should a grammar $S \rightarrow \neg S$ mean?
  - What about the following grammar?

$$S \rightarrow \neg AA \rightarrow A$$

- One approach: declare both ill-formed.
- Alternative (Kountouriotis et al., 2006): semantics in three-valued languages.
Floyd (1962): Algol 60 is not CF.
Grammar for a programming language

Floyd (1962): Algol 60 is not CF.

Extended types of grammars
(Aho, 1969; Schuler, 1974):

- Increased expressive power
- Still no grammar for any programming language.
- Clearly syntactical requirements not covered by CFGs:
  - Declaration before use.
  - No duplicate declarations.
  - Scoping rules for variables.

Done using a Boolean grammar.
Grammar for a programming language

Floyd (1962): Algol 60 is not CF.

Extended types of grammars (Aho, 1969; Schuler, 1974):
- Increased expressive power

Clearly syntactical requirements not covered by CFGs:
- Declaration before use.
- No duplicate declarations.
- Scoping rules for variables.
  Done using a Boolean grammar.
Grammar for a programming language

Floyd (1962): Algol 60 is not CF.

Extended types of grammars (Aho, 1969; Schuler, 1974):
- Increased expressive power
- Still no grammar for any programming language.
Grammar for a programming language

Floyd (1962): Algol 60 is not CF.

Extended types of grammars (Aho, 1969; Schuler, 1974):
- Increased expressive power
- Still no grammar for any programming language.

Clearly syntactical requirements not covered by CFGs:
Grammar for a programming language

Floyd (1962): Algol 60 is not CF.

Extended types of grammars (Aho, 1969; Schuler, 1974):

- Increased expressive power
- Still no grammar for any programming language.

Clearly syntactical requirements not covered by CFGs:

- Declaration before use.
Grammar for a programming language

Floyd (1962): Algol 60 is not CF.

Extended types of grammars (Aho, 1969; Schuler, 1974):
- Increased expressive power
- Still no grammar for any programming language.

Clearly syntactical requirements not covered by CFGs:
- Declaration before use.
- No duplicate declarations.
Grammar for a programming language

Floyd (1962): Algol 60 is not CF.

Extended types of grammars (Aho, 1969; Schuler, 1974):
- Increased expressive power
- Still no grammar for any programming language.

Clearly syntactical requirements not covered by CFGs:
- Declaration before use.
- No duplicate declarations.
- Scoping rules for variables.
Grammar for a programming language

Floyd (1962): Algol 60 is not CF.

Extended types of grammars (Aho, 1969; Schuler, 1974):
- Increased expressive power
- Still no grammar for any programming language.

Clearly syntactical requirements not covered by CFGs:
- Declaration before use.
- No duplicate declarations.
- Scoping rules for variables.

```c
factorial(n)
{
    if(n>1)
        return n*factorial(n-1);
    else
        return 1;
}
c(n, k)
{
    return factorial(n) / (factorial(k)*factorial(n-k))
}
main(arg)
{
    var tmp;
    tmp=c(arg*arg, arg);
    return tmp;
}
```
Grammar for a programming language

Floyd (1962): Algol 60 is not CF.

Extended types of grammars (Aho, 1969; Schuler, 1974):
- Increased expressive power
- Still no grammar for any programming language.

Clearly syntactical requirements not covered by CFGs:
- Declaration before use.
- No duplicate declarations.
- Scoping rules for variables.

Done using a Boolean grammar.

factorial(n)
{
    if(n>1)
        return n*factorial(n-1);
    else
        return 1;
}

c(n, k)
{
    return factorial(n) /
    (factorial(k)*factorial(n-k))
}

main(arg)
{
    var tmp;
    tmp=c(arg*arg, arg);
    return tmp;
}
Practical properties

Greater expressive power:
Practical properties

Greater expressive power:

- Boolean operations can be freely specified.
Practical properties

Greater expressive power:

- Boolean operations can be freely specified.
- Can generate more languages.
Practical properties

Greater expressive power:

- Boolean operations can be freely specified.
- Can generate more languages.
- *Completed form of the context-free grammars.*
Practical properties

Greater expressive power:

- Boolean operations can be freely specified.
- Can generate more languages.
- Completed form of the context-free grammars.

Key good properties of context-free grammars retained:
Practical properties

Greater expressive power:

- Boolean operations can be freely specified.
- Can generate more languages.
- *Completed form of the context-free grammars.*

Key good properties of context-free grammars retained:

- Parsing in time $O(n^3)$ (Okhotin, 2000, 2003).
Practical properties

Greater expressive power:

- Boolean operations can be freely specified.
- Can generate more languages.
- *Completed form of the context-free grammars.*

Key good properties of context-free grammars retained:

- Parsing in time $O(n^3)$ (Okhotin, 2000, 2003).
- Extension of generalized LR parsing (Okhotin, 2005).
Practical properties

Greater expressive power:

- Boolean operations can be freely specified.
- Can generate more languages.
- Completed form of the context-free grammars.

Key good properties of context-free grammars retained:

- Parsing in time $O(n^3)$ (Okhotin, 2000, 2003).
- Extension of generalized LR parsing (Okhotin, 2005).
- Extension of recursive descent parsing (Okhotin, 2007).
Practical properties

Greater expressive power:
- Boolean operations can be freely specified.
- Can generate more languages.
- Completed form of the context-free grammars.

Key good properties of context-free grammars retained:
- Parsing in time $O(n^3)$ (Okhotin, 2000, 2003).
- Extension of generalized LR parsing (Okhotin, 2005).
- Extension of recursive descent parsing (Okhotin, 2007).
- Unambiguous subfamily with $O(n^2)$ parsing (Okhotin, 2007).
Theoretical properties

- Closed under $\cup$, $\cap$, $\sim$, $\cdot$, $\ast$.
Theoretical properties

- Closed under $\cup$, $\cap$, $\sim$, $\cdot$, $\ast$.
- Closed under inverse gsm mappings (Lehtinen, Okhotin, 2008).
Theoretical properties

- Closed under $\cup$, $\cap$, $\sim$, $\cdot$, $\ast$.
- Closed under inverse gsm mappings (Lehtinen, Okhotin, 2008).
- Not closed under $h$. 
Theoretical properties

- Closed under $\cup$, $\cap$, $\sim$, $\cdot$, $\ast$.
- Closed under inverse gsm mappings (Lehtinen, Okhotin, 2008).
- Not closed under $h$.
- Undecidable emptiness.
Theoretical properties

- Closed under $\cup$, $\cap$, $\sim$, $\cdot$, $\ast$.
- Closed under inverse gsm mappings (Lehtinen, Okhotin, 2008).
- Not closed under $h$.
- Undecidable emptiness.
- Contained in $\text{DSPACE}(n)$ (Okhotin, 2003).
Theoretical properties

- Closed under $\cup$, $\cap$, $\sim$, $\cdot$, $\ast$.
- Closed under inverse gsm mappings (Lehtinen, Okhotin, 2008).
- Not closed under $h$.
- Undecidable emptiness.
- Contained in $\text{DSPACE}(n)$ (Okhotin, 2003).
- Linear subfamily characterized by cellular automata (Okhotin, 2004).
Theoretical properties

- Closed under $\cup$, $\cap$, $\sim$, $\cdot$, $\ast$.
- Closed under inverse gsm mappings (Lehtinen, Okhotin, 2008).
- Not closed under $h$.
- Undecidable emptiness.
- Contained in $\text{DSPACE}(n)$ (Okhotin, 2003).
- Linear subfamily characterized by cellular automata (Okhotin, 2004).
- Eliminating disjunction in conjunctive grammars (Okhotin, Reitwießner, 2009).
Part III

Contents of the course
A \rightarrow \alpha_1 \& \ldots \& \alpha_m
Lecture 2. Conjunctive grammars

\[ A \rightarrow \alpha_1 \& \ldots \& \alpha_m \]

- Definition by rewriting.
Lecture 2. Conjunctive grammars

\[ A \rightarrow \alpha_1 \& \ldots \& \alpha_m \]

- Definition by rewriting.
- Definition by language equations.
Lecture 2. Conjunctive grammars

\[ A \rightarrow \alpha_1 \& \ldots \& \alpha_m \]

- Definition by rewriting.
- Definition by language equations.
- Equivalence of the two definitions.
Lecture 2. Conjunctive grammars

\[ A \rightarrow \alpha_1 \& \ldots \& \alpha_m \]

- Definition by rewriting.
- Definition by language equations.
- Equivalence of the two definitions.
- Conjunctive grammar for \( \{ a^n b^n c^n \mid n \geq 0 \} \).
Lecture 2. Conjunctive grammars

$$A \to \alpha_1 \& \ldots \& \alpha_m$$

- Definition by rewriting.
- Definition by language equations.
- Equivalence of the two definitions.
- Conjunctive grammar for \( \{a^n b^n c^n \mid n \geq 0\} \).
- Conjunctive grammar for \( \{wcw \mid w \in \{a, b\}^*\} \).
Lecture 2. Conjunctive grammars

\[ A \rightarrow \alpha_1 \& \ldots \& \alpha_m \]

- Definition by rewriting.
- Definition by language equations.
- Equivalence of the two definitions.
- Conjunctive grammar for \( \{ a^n b^n c^n \mid n \geq 0 \} \).
- Conjunctive grammar for \( \{ wcw \mid w \in \{ a, b \}^* \} \).
- Conjunctive grammar for \( \{ a^{4n} \mid n \geq 0 \} \).
Lecture 3. Normal form for conjunctive grammars

\[ A \rightarrow B_1 C_1 \& \ldots \& B_m C_m \]
\[ A \rightarrow a \]
Lecture 3. Normal form for conjunctive grammars

\[ A \rightarrow B_1 C_1 \& \ldots \& B_m C_m \]
\[ A \rightarrow a \]

● Transformation to the binary normal form.
Transformation to the binary normal form.

The Cocke–Kasami–Younger parsing algorithm:

time $O(n^3)$, space $O(n^2)$. 
Lecture 3. Normal form for conjunctive grammars

\[ A \rightarrow B_1 C_1 \& \ldots \& B_m C_m \]
\[ A \rightarrow a \]

- Transformation to the binary normal form.
- The Cocke–Kasami–Younger parsing algorithm: time \( O(n^3) \), space \( O(n^2) \).
- Eliminating disjunction in conjunctive grammars.
Lecture 4. Boolean grammars

\[ A \rightarrow \alpha_1 \& \ldots \& \alpha_m \& \neg \beta_1 \& \ldots \& \neg \beta_n \]
Lecture 4. Boolean grammars

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

- Semantics by language equations.
Lecture 4. Boolean grammars

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

- Semantics by language equations.
- Semantics in terms of three-valued languages.
Lecture 4. Boolean grammars

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

- Semantics by language equations.
- Semantics in terms of three-valued languages.
- The binary normal form.
Lecture 4. Boolean grammars

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

- Semantics by language equations.
- Semantics in terms of three-valued languages.
- The binary normal form.
Lecture 5. Parsing by matrix multiplication

- The bottleneck of the Cocke–Kasami–Younger algorithm.
Lecture 5. Parsing by matrix multiplication

- The bottleneck of the Cocke–Kasami–Younger algorithm.
- Reduction of parsing to matrix multiplication.
Lecture 5. Parsing by matrix multiplication

- The bottleneck of the Cocke–Kasami–Younger algorithm.
- Reduction of parsing to matrix multiplication.
- Strassen’s method of multiplying integer matrices in time $\Theta(n^{\log_2 7})$. Applying it to Boolean matrices.
Lecture 5. Parsing by matrix multiplication

- The bottleneck of the Cocke–Kasami–Younger algorithm.
- Reduction of parsing to matrix multiplication.
- Strassen’s method of multiplying integer matrices in time $\Theta(n^{\log_2 7})$.
  Applying it to Boolean matrices.
- Multiplying Boolean matrices in time $O\left(\frac{n^3}{\log n}\right)$. 
Lecture 6. Unambiguous grammars

- Ambiguity in grammars.
Lecture 6. Unambiguous grammars

- Ambiguity in grammars.
- The well-known example:
Lecture 6. Unambiguous grammars

- Ambiguity in grammars.
- The well-known example:
  
  \[ Time \text{ flies like an arrow } \]
Lecture 6. Unambiguous grammars

- Ambiguity in grammars.
- The well-known example:
  
  *Time flies like an arrow*
  
  *Fruit flies like a banana*
Lecture 6. Unambiguous grammars

- Ambiguity in grammars.
- The well-known example:

  Time flies like an arrow
  Fruit flies like a banana
  Swallow flies like a spider
Lecture 6. Unambiguous grammars

- Ambiguity in grammars.
- The well-known example:
  
  \begin{quote}
  Time flies like an arrow
  Fruit flies like a banana
  Swallow flies like a spider
  \end{quote}

- A routine example:

  \begin{verbatim}
  if(x!=0) if(x>0) { ... } else { ... }
  \end{verbatim}
Lecture 6. Unambiguous grammars

- Ambiguity in grammars.
- The well-known example:
  
  \[ \text{Time flies like an arrow} \]
  \[ \text{Fruit flies like a banana} \]
  \[ \text{Swallow flies like a spider} \]

- A routine example:

  \[ if(x\neq 0) \hspace{1em} if(x>0) \hspace{1em} \{ \hspace{1em} \ldots \hspace{1em} \} \hspace{1em} \text{else} \hspace{1em} \{ \hspace{1em} \ldots \hspace{1em} \} \]

- Unambiguous context-free, conjunctive and Boolean grammars.
Lecture 6. Unambiguous grammars

- Ambiguity in grammars.
- The well-known example:
  
  - *Time flies like an arrow*
  - *Fruit flies like a banana*
  - *Swallow flies like a spider*

- A routine example:

  \[
  \text{if}(x != 0) \text{ if}(x > 0) \{ \ldots \} \text{ else } \{ \ldots \}
  \]

- Unambiguous context-free, conjunctive and Boolean grammars.
- Parsing in time \( O(n^2) \).
Lecture 7. Parallel context-free parsing

- Basic model: a Boolean circuit.
Lecture 7. Parallel context-free parsing

- Basic model: a Boolean circuit.
- The Brent–Goldschlager–Rytter algorithm: \( O(n^6) \) elements, time \( O(\log^2 n) \).
Lecture 7. Parallel context-free parsing

- Basic model: a Boolean circuit.
- The Brent–Goldschlager–Rytter algorithm: $O(n^6)$ elements, time $O(\log^2 n)$.
- A sequential implementation using space $O(\log^2 n)$. 
Lecture 7. Parallel context-free parsing

- Basic model: a Boolean circuit.
- The Brent–Goldschlager–Rytter algorithm: $O(n^6)$ elements, time $O(\log^2 n)$.
- A sequential implementation using space $O(\log^2 n)$.
- P-completeness of the circuit value problem.
Lecture 7. Parallel context-free parsing

- Basic model: a Boolean circuit.
- The Brent–Goldschlager–Rytter algorithm: $O(n^6)$ elements, time $O(\log^2 n)$.
- A sequential implementation using space $O(\log^2 n)$.
- P-completeness of the circuit value problem.
- A Boolean grammar for a P-complete language.
Lecture 8. Recursive descent parsing

- Each nonterminal symbol represented by a *procedure*. 
Lecture 8. Recursive descent parsing

- Each nonterminal symbol represented by a *procedure*.
- The procedures’ code transcribes grammar rules.
Lecture 8. Recursive descent parsing

- Each nonterminal symbol represented by a *procedure*.
- The procedures’ code transcribes grammar rules.
- Context-free recursive descent.
Lecture 8. Recursive descent parsing

- Each nonterminal symbol represented by a *procedure*.
- The procedures’ code transcribes grammar rules.
- Context-free recursive descent.
- Generalization for conjunctive and Boolean grammars.
Lecture 8. Recursive descent parsing

- Each nonterminal symbol represented by a *procedure*.
- The procedures’ code transcribes grammar rules.
- Context-free recursive descent.
- Generalization for conjunctive and Boolean grammars.
- Ensuring linear time complexity.
Lecture 8. Recursive descent parsing

- Each nonterminal symbol represented by a *procedure*.
- The procedures’ code transcribes grammar rules.
- Context-free recursive descent.
- Generalization for conjunctive and Boolean grammars.
- Ensuring linear time complexity.
- Limitations of the recursive descent.
Lecture 9. Generalized LR parsing

- Deterministic LR parsing (not in the course).
Lecture 9. Generalized LR parsing

- Deterministic LR parsing (not in the course).
- Context-free generalized LR: time $O(n^3)$, space $O(n^2)$. 
Lecture 9. Generalized LR parsing

- Deterministic LR parsing (not in the course).
- Context-free generalized LR: time $O(n^3)$, space $O(n^2)$.
- Extension to conjunctive grammars: time $O(n^3)$. 
Lecture 9. Generalized LR parsing

- Deterministic LR parsing (not in the course).
- Context-free generalized LR: time $O(n^3)$, space $O(n^2)$.
- Extension to conjunctive grammars: time $O(n^3)$.
- Extension to Boolean grammars: time $O(n^4)$. 
Lecture 9. Generalized LR parsing

- Deterministic LR parsing (not in the course).
- Context-free generalized LR: time $O(n^3)$, space $O(n^2)$.
- Extension to conjunctive grammars: time $O(n^3)$.
- Extension to Boolean grammars: time $O(n^4)$.
- Time $O(n)$ on “nice” grammars.
The plan

- As many lectures as can be arranged.

Contacts

- http://users.utu.fi/aleokh/
- alexander.okhotin@utu.fi
- Office 340 or 342.
The plan

- As many lectures as can be arranged.
- Two homework assignments.

Contacts

- http://users.utu.fi/aleokh/
- alexander.okhotin@utu.fi
- Office 340 or 342.
The plan

- As many lectures as can be arranged.
- Two homework assignments.
- Discussion of assignments: 2 seminars by Artur Jeż.

Contacts

- http://users.utu.fi/aleokh/
- alexander.okhotin@utu.fi
- Office 340 or 342.
The plan

- As many lectures as can be arranged.
- Two homework assignments.
- Discussion of assignments: 2 seminars by Artur Jeż.
  - Saturday, November 21?
  - Saturday, November 28?
  - Final examination.
  - Monday, November 30?

Contacts

- http://users.utu.fi/aleokh/
- alexander.okhotin@utu.fi
- Office 340 or 342.
The plan

- As many lectures as can be arranged.
- Two homework assignments.
- Discussion of assignments: 2 seminars by Artur Jeż.
  - Saturday, November 21?
  - Saturday, November 28?

Contacts

- [http://users.utu.fi/aleokh/](http://users.utu.fi/aleokh/)
- alexander.okhotin@utu.fi
- Office 340 or 342.
The plan

- As many lectures as can be arranged.
- Two homework assignments.
- Discussion of assignments: 2 seminars by Artur Jeż.
  - Saturday, November 21?
  - Saturday, November 28?
- Final examination.

Contacts

- http://users.utu.fi/aleokh/
- alexander.okhotin@utu.fi
- Office 340 or 342.
The plan

- As many lectures as can be arranged.
- Two homework assignments.
- Discussion of assignments: 2 seminars by Artur Jeż.
  - Saturday, November 21?
  - Saturday, November 28?
- Final examination.
  - Monday, November 30?

Contacts

- http://users.utu.fi/aleokh/
- alexander.okhotin@utu.fi
- Office 340 or 342.