II.1.1 Introduction to Part II

II.1.2 The Chomsky Hierarchy (12.1)
II.1.1 Introduction to Part II

II.1.2 The Chomsky Hierarchy (12.1)
II.1. Introduction

- Disclaimer
  - These notes are heavily based on
    - J. V. Tucker and K. Stephenson: *Data, Syntax and Semantics, Course Notes, Dept. of Computer Science, Swansea University, 2006.*
    Substantial parts are identical to that text.
  - Numbers in brackets (e.g. (10), (10.1)) in section headings, definitions, etc. refer to the sections in this book.

- In general none of the material in these slides is considered as original material.
Administrative Issues

Lecturer of Part II and parts of Part III:
Dr. A. Setzer
Dept. of Computer Science
University of Wales Swansea
Singleton Park
SA2 8PP
UK
Room: Room 211, Faraday Building
Tel.: (01792) 513368
Fax: (01792) 295651
Email: a.g.setzer@swansea.ac.uk
Home page: http://www.cs.swan.ac.uk/~csetzer/index.html
In this part usually all 4 slots for lecturing will be used.

This gives me more time to make the lecture more interactive.
The homepage for the parts taught by Anton Setzer is located at 
http://www.cs.swan.ac.uk/~csetzer/lectures/automataFormalLanguage/12/index.html

There is an open version, and a password protected version.
- The password is ____________.

Errors in the notes will be corrected on the slides continuously and noted on the list of errata.

The homepage contains as well additional material for each section of the module.

Assessment related material such as coursework, solutions, and the revision lecture will be made available on Blackboard.
II.1.1 Introduction to Part II

Literature


- Main Course Text.
  The slides used in this module will be heavily based on this text.


- Main additional book for this part.
- Classical text on theory of formal languages
Other Texts:


  - The famous “Dragon Book” (because of the book cover), the bible of compiler construction.

Overview over Part II: The Recognition Problem

A hierarchy of grammars.

II.1. The Chomsky Hierarchy.

Regular Languages:

II.2. Basics of Regular Languages and Expressions.
II.3. Finite State Automata.
II.4. Properties of Regular Languages.
   Includes equivalence theorem and pumping lemma.

Context-free languages:

   Includes the Pumping Lemma for context-free grammars.
   Includes equivalence theorem.
II.1.1 Introduction to Part II

II.1.2 The Chomsky Hierarchy (12.1)
A Grammar $G = (T, N, S, P)$ consists of

1. a finite set $T$ called the alphabet or set of terminal symbols,
2. a finite set $N$ of non-terminal symbols or variable symbols such that $T \cap N = \emptyset$,
3. a special non-terminal symbol $S \in N$ called the start symbol,
4. a finite set $P$ of substitution or rewrite rules, called productions, each of which has the form $u \rightarrow v$ where
   4.1 The left hand string $u \in (T \cup N)^+$ (esp. $u$ is non-empty),
   4.2 The right hand string $v \in (T \cup N)^*$.
We present a grammar as a 4-tuple $G = (T, N, S, P)$ and also use a displayed version:

<table>
<thead>
<tr>
<th>grammar</th>
<th>$G$</th>
</tr>
</thead>
<tbody>
<tr>
<td>terminals</td>
<td>$T$</td>
</tr>
<tr>
<td>nonterminals</td>
<td>$N$</td>
</tr>
<tr>
<td>start symbol</td>
<td>$S$</td>
</tr>
<tr>
<td>productions</td>
<td>$P$</td>
</tr>
</tbody>
</table>
**Example: Grammar for English Sentence**

<table>
<thead>
<tr>
<th>grammar</th>
<th>SimpleEnglishGrammarExample</th>
</tr>
</thead>
<tbody>
<tr>
<td>terminals</td>
<td>a, b, c, ..., z, A, B, ..., Z, □□</td>
</tr>
<tr>
<td>nonterminals</td>
<td>Sentence, Nounphrase, Verbphrase, Verb</td>
</tr>
<tr>
<td>start symbol</td>
<td>Sentence</td>
</tr>
<tr>
<td>productions</td>
<td>Sentence $\rightarrow$ Nounphrase □□ Verbphrase</td>
</tr>
<tr>
<td></td>
<td>Verbphrase $\rightarrow$ Verb □□ Nounphrase</td>
</tr>
<tr>
<td></td>
<td>Verb $\rightarrow$ go</td>
</tr>
<tr>
<td></td>
<td>Nounphrase $\rightarrow$ l</td>
</tr>
<tr>
<td></td>
<td>Nounphrase $\rightarrow$ home</td>
</tr>
</tbody>
</table>
Derivations (Informal)

A grammar $G$ defines a formal language $L(G) \subseteq T^*$. The elements of $L(G)$ are the set of strings we can obtain as follows:

- Start with start symbol $S$.
- Select a production such that the left hand string is a substring of what you derived so far.
- Replace this substring by the right hand string of the production.
- Once you have obtained a string consisting of non-terminals, possibly stop.

We will first give some examples and then a formal definition of $L(G)$. 
Example: English Grammar

Sentence $\rightarrow$ Nounphrase $\sqsubseteq$ Verbphrase
Verbphrase $\rightarrow$Verb $\sqsubseteq$ Nounphrase
Verb $\rightarrow$ go
Nounphrase $\rightarrow$I
Nounphrase $\rightarrow$ home

The following derives the sentence “I go home”:

Sentence $\Rightarrow$ NounPhrase $\sqsubseteq$ VerbPhrase
$\Rightarrow$ I $\sqsubseteq$ VerbPhrase
$\Rightarrow$ I $\sqsubseteq$ Verb $\sqsubseteq$ NounPhrase
$\Rightarrow$ I $\sqsubseteq$ go $\sqsubseteq$ NounPhrase
$\Rightarrow$ I $\sqsubseteq$ go $\sqsubseteq$ home
Chomsky Hierarchy

The **Chomsky hierarchy** is the classification of grammars by means of 4 properties of its production rules:

- **Regular grammars.**
  - Simple to parse. Used for dividing the input stream of characters into tokens.

- **Context-free grammars.**
  - Easy to understand and supported by parse generators.
  - In language design one aims at languages having an underlying context-free grammar.

- **Context-sensitive grammars.**
  - It’s usually an accident if the basic grammar (excluding features such as variables need to be declared) of a language is context-sensitive. C and C++ have some context-sensitive aspects (dealt with by selecting correct strings after the parsing).

- **Unrestricted grammars.**
  - The limit of grammars.
Examples of Equivalent Grammars

We give grammars of each type for defining the language

\[ L^{a^{2^n}} := \{ a^i \mid i \text{ is even} \} \]
Regular Grammars

Definition

1. A grammar $G$ is **left-linear**, iff all its productions have the form

   $$ A \rightarrow Ba \text{ or } A \rightarrow a \text{ or } A \rightarrow \epsilon $$

2. A grammar $G$ is **right-linear**, iff all its productions have the form

   $$ A \rightarrow aB \text{ or } A \rightarrow a \text{ or } A \rightarrow \epsilon $$

3. A grammar $G$ is of **Type 3** or **regular**, iff it is left-linear or right-linear

In the above we have $A, B \in N$ and $a \in T$.

Note that in a regular grammar either all productions must be left-linear or all productions must be right-linear, so no mixing of the left-linear and right-linear is allowed.
## Left Linear (Regular) Grammar for \( L^{a^{2n}} \)

<table>
<thead>
<tr>
<th>grammar</th>
<th>( G_{\text{left linear}, a^{2n}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>terminals</td>
<td>( a )</td>
</tr>
<tr>
<td>nonterminals</td>
<td>( S, A )</td>
</tr>
<tr>
<td>start symbol</td>
<td>( S )</td>
</tr>
<tr>
<td>productions</td>
<td>( S \rightarrow \epsilon )</td>
</tr>
<tr>
<td></td>
<td>( S \rightarrow Aa )</td>
</tr>
<tr>
<td></td>
<td>( A \rightarrow Sa )</td>
</tr>
</tbody>
</table>
Right Linear (Regular) Grammar for $L^{a^{2n}}$

<table>
<thead>
<tr>
<th>grammar</th>
<th>$G_{rightlinear,a^{2n}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>terminals</td>
<td>$a$</td>
</tr>
<tr>
<td>nonterminals</td>
<td>$S, A$</td>
</tr>
<tr>
<td>start symbol</td>
<td>$S$</td>
</tr>
<tr>
<td>productions</td>
<td>$S \rightarrow \epsilon$</td>
</tr>
<tr>
<td></td>
<td>$S \rightarrow aA$</td>
</tr>
<tr>
<td></td>
<td>$A \rightarrow aS$</td>
</tr>
</tbody>
</table>
Regular languages are exactly those languages which can be recognised using a constant space algorithm. So the memory used for parsing is fixed (apart from the input).

This algorithm will be an automaton, and will recognise the language in linear time

- i.e. in time $O(|s|)$, where $|s|$ is the length of the string.
Definition

Any grammar $G$ is of **Type 2** or **context-free**, if all its productions have the form

$$A \rightarrow w$$

where $A \in N$ is a nonterminal, which rewrites to a string $w \in (T \cup N)^*$. 
### Context-Free Grammar for $L^{a^{2n}}$

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>grammar</td>
<td>$G_{context-free,a^{2n}}$</td>
</tr>
<tr>
<td>terminals</td>
<td>$a$</td>
</tr>
<tr>
<td>nonterminals</td>
<td>$S$</td>
</tr>
<tr>
<td>start symbol</td>
<td>$S$</td>
</tr>
<tr>
<td>productions</td>
<td>$S \rightarrow \epsilon$</td>
</tr>
<tr>
<td></td>
<td>$S \rightarrow aSa$</td>
</tr>
</tbody>
</table>
Context free languages can be recognised in $O(|s|^4)$ time, where $|s|$ is the length of the input stream. If three tapes are allowed they can be recognised in $O(|s|^3)$ time.

Most practical examples belong to subclasses such as the LL(n)- or LR(n)-languages, which can be recognised in linear time (i.e. $O(|s|)$).
Context-Sensitive Grammars

Definition

Any grammar $G$ is of **Type 1** or **context-sensitive**, if all its productions have the form

$$uAv \longrightarrow uwv$$

where $A \in N$ is a nonterminal, which rewrites to a non-empty string $w \in (T \cup N)^+$, but only where $A$ is in the context of strings $u, v \in (T \cup N)^*$. Furthermore a production

$$A \longrightarrow \epsilon$$

is allowed, but only if $A$ does not occur in the right hand side of any production.
## Context-Sensitive Grammar for $L^{a^{2n}}$

<table>
<thead>
<tr>
<th>grammar</th>
<th>$G_{\text{context-sensitive}, a^{2n}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>terminals</td>
<td>$a$</td>
</tr>
<tr>
<td>nonterminals</td>
<td>$S, T$</td>
</tr>
<tr>
<td>start symbol</td>
<td>$S$</td>
</tr>
<tr>
<td>productions</td>
<td>$S \rightarrow \epsilon$</td>
</tr>
<tr>
<td></td>
<td>$S \rightarrow aa$</td>
</tr>
<tr>
<td></td>
<td>$S \rightarrow aaT$</td>
</tr>
<tr>
<td></td>
<td>$aT \rightarrow aTaa$</td>
</tr>
<tr>
<td></td>
<td>$aT \rightarrow aaa$</td>
</tr>
</tbody>
</table>
The class of context sensitive is PSPACE complete, where PSPACE is the class of languages, which can be recognised with polynomial amount of space.

The complexity class PSPACE is at least as big as P (polynomial time) and a strict subset of EXPTIME (exponential time).
Let in the following four definitions $G = (T, N, S, P)$ be a grammar.

**Definition**

Any grammar $G$ is of **Type 0** or **unrestricted**, so any production

$$u \rightarrow v$$

for $u \in (T \cup N)^+$, $v \in (T \cup N)^*$ are allowed.
Unrestricted Grammar for $L_{a^{2^n}}$

<table>
<thead>
<tr>
<th>grammar</th>
<th>$G_{unrestricted,a^{2^n}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>terminals</td>
<td>$a$</td>
</tr>
<tr>
<td>nonterminals</td>
<td>$S$</td>
</tr>
<tr>
<td>start symbol</td>
<td>$S$</td>
</tr>
<tr>
<td>productions</td>
<td>$S \rightarrow \epsilon$</td>
</tr>
<tr>
<td></td>
<td>$S \rightarrow aa$</td>
</tr>
<tr>
<td></td>
<td>$a \rightarrow aaa$</td>
</tr>
</tbody>
</table>
The class of unrestricted grammars is equal to the set of computably enumerable languages (also called recursively enumerable languages).

This means that we cannot decide whether a string belongs to such a language. However there exists an algorithm, which

- if a string belongs to a language, eventually will accept it,
- if a string does not belong to the language possibly will run forever.

Computable and computable enumerable languages will be discussed in part III of this module.
A Hierarchy of Languages

Definition

A language $L \subseteq T^*$ is **regular**, **context-free**, **context-sensitive**, or **unrestricted**, iff there exists a grammar $G$ of the relevant type such that $L(G) = L$.

Remark

*For any $L$ we have $L$ regular $\Rightarrow$ $L$ context-free $\Rightarrow$ $L$ context-sensitive $\Rightarrow$ $L$ unrestricted.*
We have that

- every regular grammar is context-free.
- every context-sensitive grammar is an unrestricted grammar.

However not every context-free grammar is context sensitive, since context-sensitive languages allow only productions $A \rightarrow \epsilon$ if $A$ does not occur at the right hand side of a production. (Otherwise all unrestricted languages would be context-sensitive).

However one can construct from a context-free grammar a context-free grammar of the same language, which has only productions $A \rightarrow \epsilon$, if $A$ does not occur on the right hand side of a production. This grammar is therefore context-sensitive as well.
Hierarchy of Languages

regular  \( \subseteq \) context-free  \( \subseteq \) context sensitive  \( \subseteq \) unrestricted
### Example 1 (Grammars of the Levels of the Chomsky Hierarchy)

<table>
<thead>
<tr>
<th>grammar</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>terminals</td>
<td>a, b</td>
</tr>
<tr>
<td>nonterminals</td>
<td>S</td>
</tr>
<tr>
<td>start symbol</td>
<td>S</td>
</tr>
<tr>
<td>productions</td>
<td>$S \rightarrow aSa$, $S \rightarrow bSb$, $S \rightarrow \epsilon$</td>
</tr>
</tbody>
</table>

$L(G) = ?$

G is of which type?
Example 2

| grammar     | G            |
| terminals   | a            |
| nonterminals| S            |
| start symbol| S            |
| productions | $S \rightarrow a, S \rightarrow aS$ |

$L(G) = \ ?$

$G$ is of which type?
### Example 3

<table>
<thead>
<tr>
<th>grammar</th>
<th>$G$</th>
</tr>
</thead>
<tbody>
<tr>
<td>terminals</td>
<td>$a, b$</td>
</tr>
<tr>
<td>nonterminals</td>
<td>$S$</td>
</tr>
<tr>
<td>start symbol</td>
<td>$S$</td>
</tr>
<tr>
<td>productions</td>
<td>$S \rightarrow ab, S \rightarrow aSb$</td>
</tr>
</tbody>
</table>

$L(G) = ?$

$G$ is of which type?
Example 4 (Grammars)

Consider the grammar

<table>
<thead>
<tr>
<th>grammar</th>
<th>$G^{a^n b^n c^n}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>terminals</td>
<td>$a, b, c$</td>
</tr>
<tr>
<td>nonterminals</td>
<td>$S, B, C$</td>
</tr>
<tr>
<td>start symbol</td>
<td>$S$</td>
</tr>
<tr>
<td>productions</td>
<td>$S \rightarrow aSBC, \quad S \rightarrow aBC,$  [CB \rightarrow BC, \quad aB \rightarrow ab, \quad bB \rightarrow bb, \quad bC \rightarrow bc, \quad cC \rightarrow cc.]</td>
</tr>
</tbody>
</table>

Example 4 is not a context sensitive grammar (although often referred to as context sensitive). Why?
A derivation of $aaabbbcccc$ in Example 4 is as follows:

\[
\begin{align*}
S & \Rightarrow aSBC = aSBC \\
& \Rightarrow aaSBCBC = aaSBCBC \\
& \Rightarrow aaaBCBCBCC = aaaBCBCBCC \\
& \Rightarrow aaaBCBBCC = aaaBCBBCC \\
& \Rightarrow aaaBBCC = aaaBBCC \\
& \Rightarrow aaBBCC = aaBBCC \\
& \Rightarrow aaBBCCC = aaBBCCC \\
& \Rightarrow aaBCCC = aaBCCC \\
& \Rightarrow aBCCC = aBCCC \\
& \Rightarrow BBCCC = BBCCC \\
& \Rightarrow aBCCC = aBCCC \\
& \Rightarrow bbCCC = bbCCC \\
& \Rightarrow bCCC = bCCC \\
& \Rightarrow cCCC = cCCC \\
& \Rightarrow BCCC = BCCC \\
& \Rightarrow BBCCC = BBCCC \\
& \Rightarrow BCC = BCC \\
& \Rightarrow CCC = CCC \\
\end{align*}
\]
Example 5 (Grammars)

Here is a variant which is context sensitive:

| grammar | $G^{a^n b^n c^n}$ |
| terminals | $a, b, c$ |
| nonterminals | $S, B, C, H$ |
| start symbol | $S$ |
| productions | $S \rightarrow aSBC, S \rightarrow aBC,$  
$CB \rightarrow HB, HB \rightarrow HC, HC \rightarrow BC,$  
$aB \rightarrow ab, bB \rightarrow bb, bC \rightarrow bc, cC \rightarrow cc.$ |
Example Derivation

A derivation of \( aabbcc \) in Example 5 is as follows:

\[
\begin{align*}
S & \Rightarrow aSBC \\
& \Rightarrow aaBCBC \\
& \Rightarrow aaBHBC \\
& \Rightarrow aaBHCC \\
& \Rightarrow aaBBCC \\
& \Rightarrow aabBCC \\
& \Rightarrow aabbCC \\
& \Rightarrow aabbCC \\
& \Rightarrow aabbcc
\end{align*}
\]
Examples

- Regular: $\{a^n \mid n \geq 1\}$
- Context-free: $\{a^n b^n \mid n \geq 1\}$
- Context-sensitive: $\{a^n b^n c^n \mid n \geq 1\}$
- Unrestricted