$ \begin{array}{c} \text{Ly such asic context free grammars} & \text{Ly such asic context free grammars} & \text{Ly such as a context free grammar} & \text{Ly such as a context free grammar} & \text{Ly such as a such as a context free grammar} & \text{Ly such as a such as a context free grammar} & \text{Ly such as a such as a context free grammar} & \text{Ly such as a such as a context free grammar} & \text{Ly such as a such as a context free grammar} & \text{Ly such as a such as a context free grammar} & \text{Ly such as a such as a context free grammar} & \text{Ly such as a context free grammar} & \text{Ly such as a context free grammar} & \text{Ly such as a such as a context free grammar} & \text{Ly such as a such as a context free grammar} & \text{Ly such as a such as a context free grammar} & \text{Ly such as a grammar} & \text{Ly such as a such as a context free grammar} & \text{Ly such as a such as a such as a context free grammar} & \text{Ly such as a such as a context free grammar} & \text{Ly such as a such as a context free grammar} & \text{Ly such as a such a such as a such as a context free grammar} & Ly such as a s$
$ \begin{array}{c} \text{Ly stochastic context free grammars} & \frac{ x_1 + \cdots + y_{k_1} }{ x_1 + \cdots + y_{k_k} } \\ \text{Atore, } a, \beta \text{ et. start for arbitrary sequences of terminals and nonvenirules.} \\ \textbf{Nose} \\ \text{aractry} \\ aractr$
$ \begin{array}{c} \text{Ly suchassic context free grammars} \\ \text{arguing result} \\ \text{arguing ressingly ressing result} \\ \text{arguing result} \\ \text{arguing result} \\ $
$ \begin{array}{c} \text{Let} \\ \text{Jow sochastic context free grammars} \\ Jow sochastic context is described by the LTr. grammar (e.g. Jow sochastic degrammars and finite state automatic is described by the LTr. grammar (e.g. Jow sochastic degrammars and finite state automatic is described by the LTr. grammar (e.g. Jow sochastic degrammars and finite state automatic is described by the LTr. grammar (e.g. CCG CTG, CCG CCG CCG CCG CCG CCG CCG CCG CCG C$
$ \begin{array}{c} \  \  \  \  \  \  \  \  \  \  \  \  \ $
$ \begin{array}{c} \label{eq:context} \left[ \begin{array}{c} u_{1} (x,y_{2}) & ($
$ \begin{array}{c} \label{eq:context free grammars} \end{array} \\ \end{tabular} \end{tabular}$
The Backues Naur form: $(S) ::= a(S)   b(S)   (empty)$ The Backues Naur for a start symbol $S$ , and rules $R$ (above). The Backues Naur form: $(S) ::= a(S)   b(S)   (empty)$
$ \begin{array}{c} \text{Lemannear grammars} \\ \\ \\ \text{Lemannear grammars} \\ \\ \\ \{Lemannear grammars} \\ \\ \{Lemannear grammars} \\ \\ \{Lemannear grammars} \\ \\ \\ \{Lemannear grammars} \\ \\ \{Lemannear gram$
$ for the grammars and context free grammars are widely used in computer sole     arachy a start symbol S, and rules R (above).     \{a, b\}, a \text{ start symbol S}, and rules R (above).  for the solution of the formula start symbol S, and rules R (above).  for the solution of the start symbol S, and rules R (above).  for the solution of the solution of the start symbol S, and rules R (above).  for the solution of the solution of the start symbol S, and rules R (above).  for the solution of the $
$ \begin{array}{c} \  \  \  \  \  \  \  \  \  \  \  \  \ $
I by stochastic context free grammars d by stochastic context free grammars and nonterminals. Hores in trees in the state automatic distribution of the state automatic distribution computational trees in the state automatic distribution compatibility and nules the face of the state automatic distribution compatibility and the state automatic distribution compatibility and trees in the state automatic distribution compatibility and trees in the state automatic distribution compatibility and the state automatic distribution and the
d by stochastic context free grammars $[a_1, \dots, c_n]$ Above, $\alpha, \beta$ etc. stard for arbitrary sequences of terminals and nonterminals.         Notes       • Regular grammars and context free grammars are widely used in computer sciention trees         ion trees       • Programming languages are described by the so-called <i>LR</i> or <i>LL</i> subclasses i         n       • Only part of programming language stret described by the so-called <i>LR</i> or <i>LL</i> subclasses i         n       • Unrestricted grammars are also called term rewrite systems         n       • Unrestricted grammars are also called term rewrite systems         • Intelbegy tree=12:20       Papert         vt       V         Seminar on computational toology reserved.         • <i>b_S</i> • <i>b_S</i> • <i>c</i> • FMR-1 triplet region contains repeats of the triplets CGG and occasionally AGG GCG CTG, GCG
iby stochastic context free grammars $e_1 = 1 - e_2$ $e_1$ arachy       Above, $\alpha, \beta$ etc. stand for arbitrary sequences of terminals and nonterminals.         ness       • Regular grammars and context free grammars are widely used in computer scient         ion trees       • Programming languages are described by the so-called <i>LR</i> or <i>LL</i> subclasses (         n       • Only part of programming languages are also called term rewrite systems         n       • Unrestricted grammars are also called term rewrite systems         nulbergy trees-1220       Page 1         via babory trees-1220       Seminar on computational babory rest-1220         GCG CTG, GCG CGG CTG, GCG CG
1 by stochastic context free grammars       Page 1       Above, $\alpha, \beta$ etc. stand for arbitrary sequences of terminals and nonterminals.         arachy       • Regular grammars and context free grammars are widely used in computer scie         ion trees       • Programming languages are described by the so-called LR or LL subclasses •         ion trees       • Only part of programming language surfax is described by the LR-grammar (e.         in trees       • Chomsky's original goal was to describe aturn ewrite systems         in albology 1991-1220       Page 1         vial bology 1991-1220       The FMR-1 triplet region contains repeats of the triplets CGG and occasionally AGG
$f = \frac{1}{2} + $
1 by stochastic context free grammars       Above, α, β etc. stand for arbitrary sequences of terminals and nonterminals.         Notes       • Regular grammars and context free grammars are widely used in computer scie         ion trees       • Programming languages are described by the so-called <i>LR</i> or <i>LL</i> subclasses.         n       • Unrestricted grammars are also called term rewrite systems         n       • Chomsky's original goal was to describe natural languages. He (and everybody         nal biology 1989-12-20       Page 1         KL       Seminar on computational biology 1989-12-20         Regular grammars and finite state automata
they stochastic context free grammars       Above, α, β etc. stand for arbitrary sequences of terminals and nonterminals.         Notes       • Regular grammars and context free grammars are widely used in computer scie         ararchy       • Regular grammars and context free grammars are widely used in computer scie         ion trees       • Only part of programming languages are described by the so-called <i>LR</i> or <i>LL</i> subclasses         • Unrestricted grammars are also called term rewrite systems         • Chomsky's original goal was to describe natural languages. He (and everybody         • Natbology 1999-12:20       • Page 1         wt       Seminar on computational biology 1999-12:20
$\frac{\alpha}{\alpha} = \frac{\alpha}{\alpha} + \frac{\alpha}{\alpha} + \frac{\alpha}{\beta}$ etc. stand for arbitrary sequences of terminals and nonterminals. <b>Notes</b> <b>a</b> Regular grammars and context free grammars are widely used in computer sci <b>b</b> Programming languages are described by the so-called <i>LR</i> or <i>LL</i> subclasses <b>c</b> Only part of programming language syntax is described by the <i>LR</i> -grammar ( <b>c</b> Unrestricted grammars are also called term rewrite systems <b>c</b> Chomsky's original goal was to describe natural languages. He (and everybod) <b>c</b> Mathematical States and computational biology 1999-12-20
d by stochastic context free grammars       Above, α, β etc. stand for arbitrary sequences of terminals and nonterminals.         Notes       • Regular grammars and context free grammars are widely used in computer sci         ion trees       • Programming languages are described by the so-called <i>LR</i> or <i>LL</i> subclasses         • Only part of programming language syntax is described by the <i>LR</i> -grammar (         • Unrestricted grammars are also called term rewrite systems         • Chomsky's original goal was to describe natural languages. He (and everybod)
$ \begin{array}{c} \textbf{Convexer grammars} \\ \textbf{Above, } \alpha, \beta \text{ etc. stand for arbitrary sequences of terminals and nonterminals.} \\ \textbf{Notes} \\ \textbf{ararchy} \\ ara$
$\begin{array}{c} \textbf{c} \textbf{c} \textbf{c} \textbf{c} \textbf{c} \textbf{c} \textbf{c} \textbf$
a by stochastic context free grammars $Above, \alpha, \beta$ etc. stand for arbitrary sequences of terminals and nonterminals. <b>Notes</b> • Regular grammars and context free grammars are widely used in computer so • Programming languages are described by the so-called $LR$ or $LL$ subclasse
Above, $\alpha, \beta$ etc. stand for arbitrary sequences of terminals and nonterminals. Notes  • Regular grammars and context free grammars are widely used in computer sc
ararchy $\alpha, \beta$ etc. stand for arbitrary sequences of terminals and nonterminals. Notes $\alpha, \beta$ etc. stand for arbitrary sequences of terminals and nonterminals. Notes $\alpha, \beta$ etc. stand for arbitrary sequences of terminals and nonterminals.
Above, $\alpha, \beta$ etc. stand for arbitrary sequences of terminals and nonterminals. Notes
by stochastic context free grammars $lpha$ , $eta$ , eta, $eta$ , eta, $eta$ , $eta$ , eta, $eta$ , $eta$ , $eta$ , eta, $eta$ , $eta$ , eta, $eta$ , $eta$ , eta, $eta$ , eta, $eta$ , eta, $eta$ , eta, eta, $eta$ , eta, eta, $eta$ , eta, $eta$ , eta, eta, $eta$ , eta, eta, eta, eta, eta, $eta$ , eta, et
Intestricted drammars $\rho_1 W \rho_2 \longrightarrow \gamma_1$
nars Context sensitive grammars $a_1Wa_2 \longrightarrow a_1eta a_2$ , non-co
Context tree grammars $V \longrightarrow p$
Varkov models
<b>P</b> equilar reasonance $W \rightarrow aW$ and $W \rightarrow a$
Grammar class Admissible rules
By restricting the possible form of grammar rules, we get a hierarchy of increasing
sestori@dna.kvi.dk Grammar classes: the Chomsky hierarchy (ca. 1956)

Page 3

KVL Seminar on computational biology 1999-12-20 Page 6	Expressive power in the Chomsky hierarchy Let $\alpha, \beta \in (N \cup T)^*$ denote arbitrary sequences of nonterminals and terminals. • Context sensitive grammars can describe copy languages, such as: $\{\alpha \alpha \mid \alpha \in (N \cup T)^*\}$ Context free grammars can describe palindrome languages (where $\alpha^{-1}$ is $\alpha$ reversed), such as: $\{\alpha \alpha^{-1} \mid \alpha \in (N \cup T)^*\}$ (Context free grammars can describe proper parenthetical nesting, as used in programming languages). Regular grammars can describe languages with uncorrelated repeats, such as: $\{a^n \alpha a^m \mid \alpha \in (N \cup T)^*\}$ But they cannot encode the additional requirement $n = m$ .	Regular grammars and regular expressions         It is customary to use regular expressions as a shorthand for regular grammars.         Every regular expression corresponds to a regular grammar and vice versa.         The FMR-1 grammar can be written as the regular expression: $gcgcgg((a + c)gg)^*cdg$ In uNix grep or emacs notation, that is: $gcgcgg([a c]gg)^*cdg$ More regular expressions: PROSITE patterns         Regular expressions are used to describe 'signature' conserved protein sequences and their variants.         Brackets [RK] indicate choice, braces {EDRKHPCG} choice from the complement, x matches anything.         (Figure 9.3)         More regular expression are used to describe 'signature' conserved protein sequences and their variants.         Brackets [RK] indicate choice, braces {EDRKHPCG} choice from the complement, x matches anything.         (Figure 9.3)
KVL Seminar on computational biology 1999-12-20 Page 8	<ul> <li>Derivation trees and parsing</li> <li>The derivation from a context free grammar may be shown as a tree.</li> <li>(The derivation from a regular grammar is a degenerate – linear – tree, not very interesting).</li> <li>Parsing: find a derivation tree for a given sequence, if any</li> <li>Given a grammar <i>G</i> and a sequence <i>α</i>.</li> <li>is sequence <i>α</i> derivable from <i>G</i>?</li> <li>if so, what derivation trees would produce sequence <i>α</i>?</li> </ul>	A context free grammar for palindromes over { $a, b$ } $S \rightarrow aSa   bSb   aa   bb$ $Derivation of the palindrome aabaabaa S \Rightarrow aSa \Rightarrow aaSaa \Rightarrow aabSbaa \Rightarrow aabaabaa Context free grammars and RNA secondary structureRNA sequences can form 'stem loops' when one part of the sequence matches another part (a \leftrightarrow u, c \leftrightarrow g).Three-base RNA stem loops' when one part of the sequence matches another part (a \leftrightarrow u, c \leftrightarrow g).Three-base RNA stem loops' when S \rightarrow aW_1u   cW_1g   gW_1c   uW_1aW_1 \rightarrow aW_2u   cW_2g   gW_2c   uW_2aW_2 \rightarrow aW_3u   cW_3g   gW_3c   uW_3aW_3 \rightarrow gaaa   gcaa$

Seminar on computational biology 1999-12-20

Let $n$ be the length of the sequence be $\lceil$	eing parsed.		-			Regular	Simple ('deterministic') PROSITE patterns	Stochastic Primary structure, probabilistically (HMM)
Grammar class	Recognizer	Time complexity	Space complexity			Context free	RNA secondary structure	RNA secondary structure, probabilistically
Regular grammars	Finite state automata	Linear	Constant					
Context free grammars	Pushdown automata	$O(n^3)$	$O(n^3)$		Chomsky	/ normal form		
Context sensitive grammars	Linear bounded automata	NP-complete	PSPACE-complete		A gramm:	त्रr is on Chomsky r	normal form if all rules have fo	orm $W \longrightarrow W_1 W_2$ or $W \longrightarrow a.$
Unrestricted grammars	Turing machines	Undecidable	Unbounded		Every cor	ntext free grammar	· can be transformed to Chorr	nsky normal form.
	Seminar on computational biology 199	99-12-20		Page 9	KVL		Seminar on computation	al biology 1999-12-20
Stochastic regular grammars					Finding t Dynamic A probabi	he most probable programming (tabu ilistic version of the stochastic context	parse tree: the CYK algori Jation). Analogous to the Vit Cocke-Younger-Kasami (CY Cocke-Younger-Kasami (CY)	<b>ithm</b> erbi algorithm, which finds the most probable $\varepsilon$ 'K) algorithm for context free grammar parsing v normal form. and a sequence $x = x_1$ r.
For each non-terminal sympole ${\it W}$ , ass. Think of $W$ as a state and $a$ as an em			► 117		Finding t Dynamic A probabi Input: A Let the g	he most probable programming (tabu listic version of the listic context stochastic context 'ammar <i>G</i> have te	; parse tree: the CYK algori Jation). Analogous to the Vitt Cocke-Younger-Kasami (CY Cocke-Younger-Kasami (CY Tree grammar $G$ on Chomsky free grammar $T = \{V_1,$	ithm erbi algorithm, which finds the most probable $\varepsilon$ erbi algorithm for context free grammar parsing 'K' algorithm for context free grammar parsing y normal form, and a sequence $x=x_{1L}.$ . , $V_M\}$ and start symbol $S=V_1.$
The stochastic grammar emits symbols A hidden Markov model (HMM) emits s	sign probabilities to rules of to nitted symbol.	orm $W \longrightarrow a W_1$	or $W \longrightarrow a.$		Finding t Dynamic A probabi Input: A : Let the gr Let $e_v(a)$ Let $t_v(y)$	he most probable programming (tab listic version of the stochastic context ammar <i>G</i> have te ) be <i>p</i> if $V_v \longrightarrow c$	<b>*</b> parse tree: the CYK algori Jation). Analogous to the Vite Cocke-Younger-Kasami (CY Cocke-Younger-Kasami (CY Cy Cocke-Younger-Kasami (CY Cocke-Younger-Kasami (C) Cocke-Younger-Kasami (C) Cocke-Younger-Kasami (C) Cocke-Younger-Kasami (C) Cocke-Younger-Kasami (C) Cocke-Younger-Kasami (C) Cocke-Younger-Kasami (C) Cocke	<b>ithm</b> erbi algorithm, which finds the most probable a 'K) algorithm for context free grammar parsing y normal form, and a sequence $x = x_{1L}$ . , $V_M$ and start symbol $S = V_1$ . herwise. nd 0 otherwise.
The two kinds of machines are interco	sign probabilities to rules of fe nitted symbol. Is on state transitions.	orm $W \longrightarrow a W_1$ ions, and emits not	or $W \longrightarrow a.$ hing on state transitio	φ.	Finding t Dynamic A probabi Input: A : Let the gr Let $t_v(a)$ Output: $\prime$ Algorith	he most probable programming (tab listic version of the stochastic context ammar <i>G</i> have ter ) be <i>p</i> if $V_v \longrightarrow c$ ) be <i>p</i> if $V_v \longrightarrow c$ 2) be <i>p</i> if $V_v$ table $\gamma$ such that A table $\gamma$ such that	<b>;</b> parse tree: the CYK algori Jation). Analogous to the Vite Cocke-Younger-Kasami (CY Cocke-Younger-Kasami (C) cocke-Younger-Kasami (C) cocke-Yo	<b>ithm</b> erbi algorithm, which finds the most probable a 'K) algorithm for context free grammar parsing y normal form, and a sequence $x = x_{1L}$ . , $V_M$ } and start symbol $S = V_1$ . herwise. nd 0 otherwise. of the most probable parse tree that derives $x_i$ . = 1 <i>M</i> .
	sign probabilities to rules of to nitted symbol. Is on state transitions. symbols without state transit	orm $W \longrightarrow a W_1$ ions, and emits not	or $W \longrightarrow a.$ hing on state transitio	<u>~</u>	Finding t Dynamic A probabi Input: A : Let $t_v(x)$ , Let $t_v(y)$ , Output: $\lambda$ For $i = 1$	he most probable programming (tab listic version of the stochastic context ammar <i>G</i> have ter ) be <i>p</i> if $V_v \longrightarrow c$ ) be <i>p</i> if $V_v \longrightarrow c$ 2) be <i>p</i> if $V_v$ that A table $\gamma$ such that A table $\gamma$ such that I $(L - 1)$ and <i>j</i>	<b>;</b> parse tree: the CYK algori Jation). Analogous to the Viti ; Cocke-Younger-Kasami (CY ; minal symbols $T = \{V_1,, V_n,, V_n\}$ ; with probability $p$ , and 0 ott ; $V_yV_z$ with probability $p$ , ar ; $\gamma(i, j, v)$ is the probability $p$ , ar ; $\gamma(i, j, v)$ is the probability $p$ ; $e_v(x_i)$ for $i = 1L$ and $v$	<b>ithm</b> erbi algorithm, which finds the most probable a 'K) algorithm for context free grammar parsing 'y normal form, and a sequence $x = x_{1L}$ . , $V_M$ } and start symbol $S = V_1$ . herwise. herwise. nd 0 otherwise. of the most probable parse tree that derives $x_i$ . = 1 $M$ .
c	ign probabilities to rules of fr nitted symbol. Is on state transitions. symbols without state transit nvertible (by introducing extr	orm $W\longrightarrow aW_1$ ions, and emits not a states and transif	or $W \longrightarrow a$ . hing on state transitio ions). stochastic regular gra	imar.	Finding t Dynamic A probabi Input: A : Let the gr Let $t_v(y)$ . Cutput: $\iota$ Algorithr For $i = 1$	he most probable programming (tab listic version of the stochastic context ammar <i>G</i> have ter ) be <i>p</i> if $V_v \longrightarrow c$ 2) be <i>p</i> if $V_v \longrightarrow c$ 2) be <i>p</i> if $V_v$ and <i>i</i> 1( <i>L</i> - 1) and <i>j</i> 1( <i>L</i> - 1) and <i>j</i>	<b>* parse tree: the CYK algori</b> Jation). Analogous to the Vite Cocke-Younger-Kasami (CY Cocke-Younger-Kasami (CY minal symbols $T = \{V_1,, white weights in the probability p, and 0 othV_y V_z with probability p, arV_y V_z with probability p, ar(\gamma(i, j, v)) is the probability o= e_v(x_i) for i = 1L and v= (i + 1)L$ and $v = 1M= (i + 1)L$ and $v = 1M$	Ithm erbi algorithm, which finds the most probable a 'Y) algorithm for context free grammar parsing $y$ normal form, and a sequence $x = x_{1L}$ . . , $V_M$ and start symbol $S = V_1$ . herwise. nd 0 otherwise. nd 0 otherwise. If the most probable parse tree that derives $x_i$ . = 1M. I, put I, put $x, y) \cdot \gamma (k + 1, j, z) \cdot t_v(y, z))$
c	sign probabilities to rules of t nitted symbol. son state transitions. symbols without state transit nvertible (by introducing extr	orm $W \longrightarrow a W_1$ ions, and emits not a states and transif aquence $lpha$ using a	or $W \longrightarrow a.$ hing on state transitio ions). stochastic regular gra	imar.	Finding t Dynamic A probabi Input: A : Let the gr Let $e_v(a)$ Let $t_v(y)$ Algorithn For $i = 1$	he most probable programming (tablistic version of the listic version of the stochastic context ammar <i>G</i> have ten ) be <i>p</i> if $V_v \longrightarrow c$ ) be <i>p</i> if $V_v \longrightarrow c$ 1 be <i>p</i> if $V_v = -i$ 1 be <i>i</i> if $V_v = -i$ 1 be <i>p</i> if $V_v = -i$ 1 be <i>i</i> if $V_v = -i$	<b>; parse tree: the CYK algori</b> Jlation). Analogous to the Viti i Cocke-Younger-Kasami (CY i Cocke-Younger-Kasami (CY i Cocke-Younger-Kasami (CY i Cy i Cy the grammar <i>G</i> on Chomsky minal symbols $T = \{V_1,, v_i\}$ i with probability <i>p</i> , and 0 oth $V_y V_z$ with probability <i>p</i> , ar $V_y V_z$ with probability <i>p</i> , ar $V_y V_z$ with probability <i>p</i> , ar $P_y V_z$ with probability <i>p</i> , ar $P_y V_z$ with probability <i>p</i> . $P_y V_z$ with probability <i>p</i> , ar $P_y V_z$ with <i>p</i> , ar $P_y V_z$ with probability <i>p</i> , ar $P_y V_z$ with probabi	Ithm erbi algorithm, which finds the most probable a 'K) algorithm for context free grammar parsing ' normal form, and a sequence $x = x_{1L}$ . . , $V_{M}$ } and start symbol $S = V_1$ . herwise. nd 0 otherwise. nd 0 otherwise. if the most probable parse tree that derives $x_i$ . = 1M. f, put f, put $x, y) \cdot \gamma (k + 1, j, z) \cdot t_v(y, z))$ $G$ is $\gamma(1, L, 1)$ .
	sign probabilities to rules of t nitted symbol. symbols without state transit nvertible (by introducing extr nvertible (by introducing of se	orm $W\longrightarrow aW_1$ ions, and emits not a states and transit equence $lpha$ using a	or $W \longrightarrow a.$ hing on state transitio .ions). stochastic regular gra	imar.	Finding t Dynamic A probabi Input: A : Let the gr Let $e_v(a)$ Let $t_v(y)$ . Coutput: / Algorithr For $i = 1$ The prob	he most probable programming (tablistic version of the listic version of the stochastic context ammar $G$ have ter ) be $p$ if $V_v \longrightarrow c$ ) be $p$ if $V_v \longrightarrow c$ table $\gamma$ such that $\lambda$ table $\gamma$ such that $n$ : Put $\gamma(i, i, v) =$ $\dots (L - 1)$ and $j$ $\dots (L - 1)$ and $j$ $\dots (L - 1)$ and $j$	<b>;</b> parse tree: the CYK algori Jlation). Analogous to the Viti ; Cocke-Younger-Kasami (CY rminal symbols $T = \{V_1,, with probability p, and 0 othV_y V_z with probability p, and 0 othV_y V_z with probability p, ar:\gamma(i, j, v) is the probability o = e_v(x_i) for i = 1L and v = 1h= (i + 1)L$ and $v = 1h= (i + 1)L$ and $v = 1h= \sum_{y,z} \sum_{k=i}^{j-1} (\gamma(i, k))probable derivation of x fromis built as in the Viterbi algor$	Ithm erbi algorithm, which finds the most probable a 'Y) algorithm for context free grammar parsing $Y$ normal form, and a sequence $x = x_{1L}$ . . $V_M$ and start symbol $S = V_1$ . herwise. nd 0 otherwise. nd 0 otherwise. if the most probable parse tree that derives $x_i$ = 1 $M$ . I, put I, put $i, y) \cdot \gamma (k + 1, j, z) \cdot t_v(y, z))$ $G$ is $\gamma(1, L, 1)$ . ithm.

Ş

## Grammar classes and 'parsers' (recognizers)

For each grammar class there is a characteristic 'machine type' that solves the parsing problem for that class.

Stochastic context free grammars

For each non-terminal symbol W, assign a distribution to the set of rules  $W \longrightarrow \alpha.$ 

More expressive grammar classes require more powerful recognizers (and more time and space).

Grammar class	Recognizer	Time complexity	Space complexity
Regular grammars	Finite state automata	Linear	Constant
Context free grammars	Pushdown automata	$O(n^3)$	$O(n^3)$
Context sensitive grammars	Linear bounded automata	NP-complete	PSPACE-complete
Unrestricted grammars	Turing machines	Undecidable	Unbounded

Ř