

Models of Grammar Evolution I
Biosequence Applications

20.6.07

Sónia Martins
Bruno Martins
José Cruz

IGC, February 20th, 2008

What are Grammars?

“A formal grammar is:
a precise description of a *formal language*.”

in Wikipedia

“A formal language is:

An organized set of symbols (...) that (...) can be precisely defined in terms of just the shapes and locations of those symbols. Such a language can be defined, then, without any reference to any meanings of any of its expressions; it can exist before any interpretation is assigned to it --that is, before it has any meaning.”

in Wikipedia

- A computer language (e.g. python) is the most handy example of a formal language.
- In our case the language can be: secondary structure of RNA, protein fold, sequence alignment...

How we define grammars?

The grammar $G = \{T, N, P\}$ consists of:

- **T** – A finite set of terminal elements
- **N** – A finite set of non-terminal elements
- **P** – A finite set of production rules

Example:

$T = \{a, b, c\}$, the “words” of our language

$N = \{S, A, B, C\}$

P:

$S \rightarrow A \mid B \mid C \mid \varepsilon$

$A \rightarrow aS \mid a$

$B \rightarrow baS$

$C \rightarrow caS$

This grammar can produce the following texts:

- **a** (from $S1, A2,$)
- **aaaa** (from $S1, A1, S1, A1, S1, A1, S1, A1$)
- **ba** (from $S2, B, S4$)
- **bacaba** (from $S2, B, S3, C, S2, B, S4$)

...and a infinity of other useless garbage.

Types of grammars

Chomsky classification:

Type 3: Regular $(A \rightarrow \alpha A \mid \alpha)$

Type 2: Context Free $(A \rightarrow \gamma)$

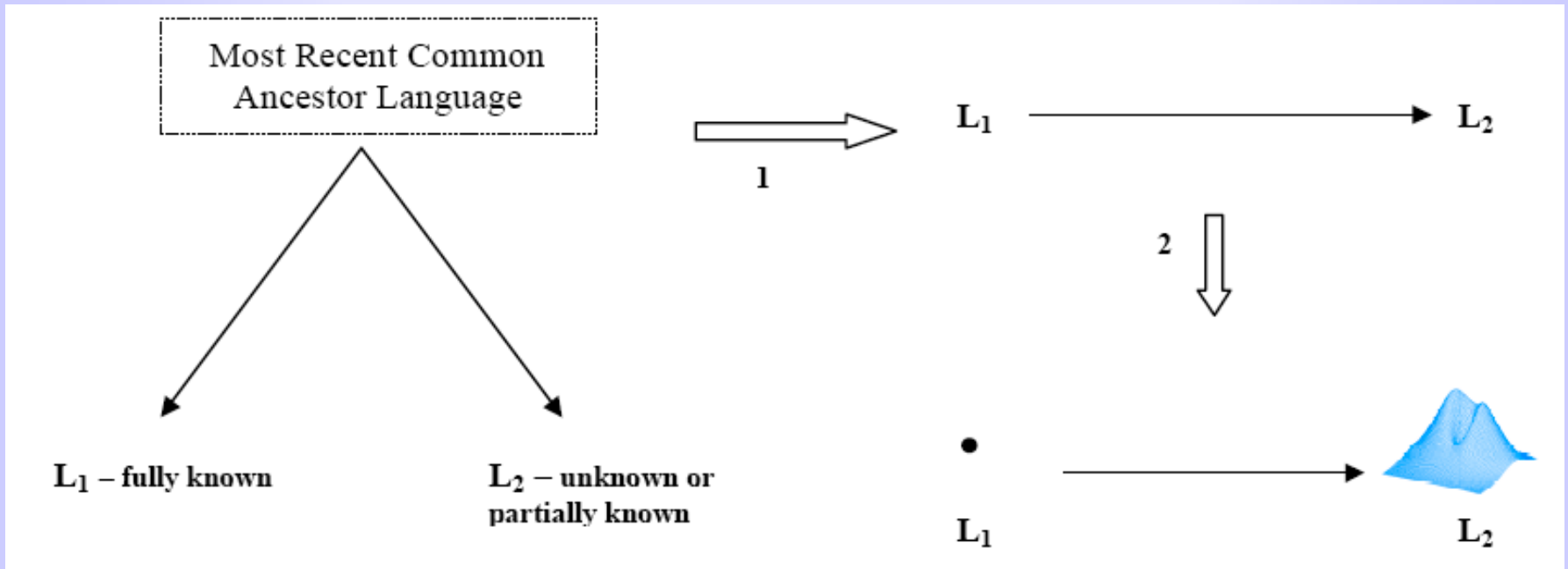
Type 1: Context Sensitive $(\alpha A \beta \rightarrow \alpha \gamma \beta)$

Type 0: Recursively Enumerable $(\alpha \rightarrow \beta)$

α, β : sequence of terminals

γ : non empty sequence of terminals and non-terminals

Linguistics in sequence analysis



Evolutionary process is time reversible

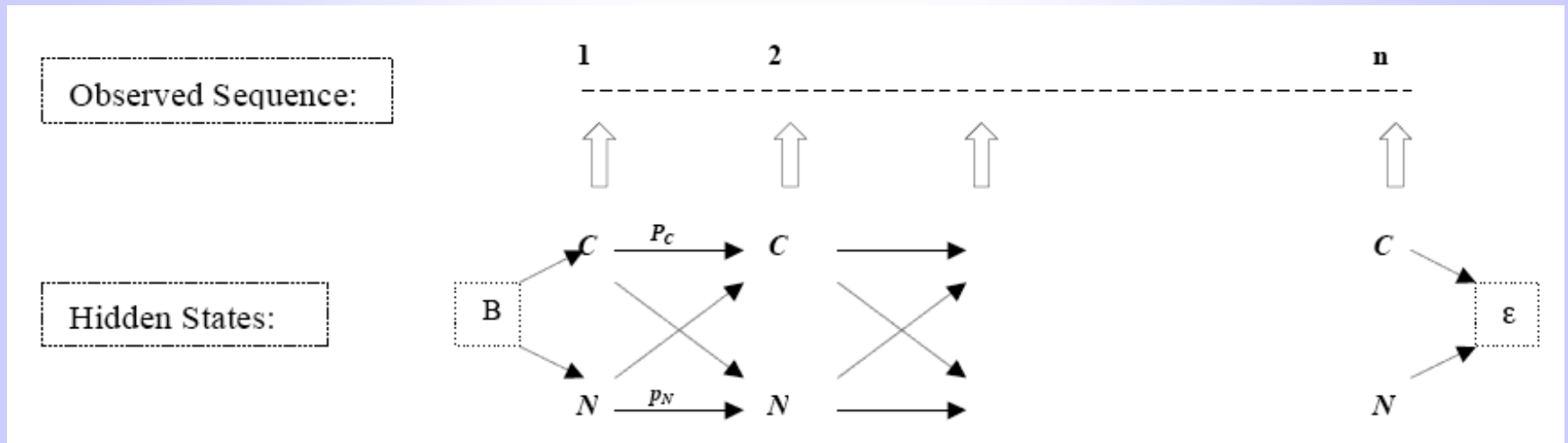
L₁ - all parameters and rules are known

L₂ - set of probabilities derived from the evolution of L₁

Languages and application areas

Stochastic Regular (SRG):

Hidden Markov Models



Protein gene finding
Comparative genomics

Languages and application areas

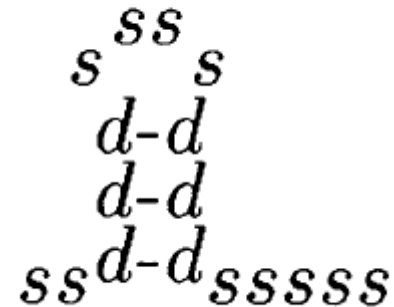
Stochastic Context Free (SCFG):

$S \rightarrow LS (.87) \mid L (.13)$
 $F \rightarrow dFd (.79) \mid LS (.21)$
 $L \rightarrow s (.89) \mid dFd (.11)$



Example:

s s d d d s s s s d d d s s s s s



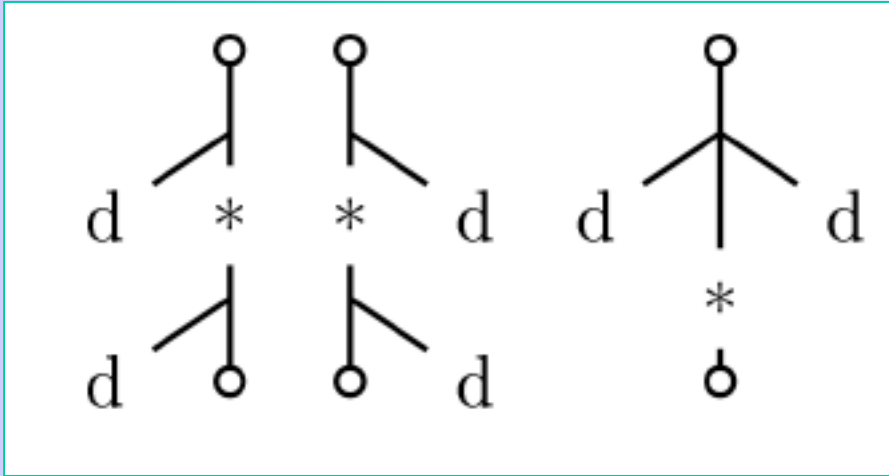
Knudsen et al., Bioinformatics, 15, 6, 1999, 446-454

RNA structure prediction
Gene finding

Time complexity: $O(N^3)$

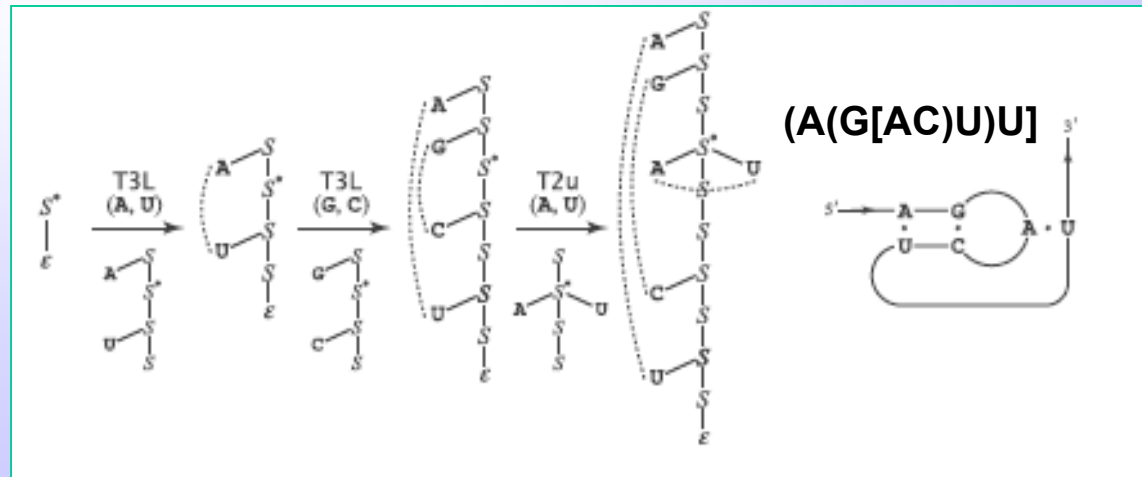
Languages and application areas

Stochastic tree adjoining (STAG):

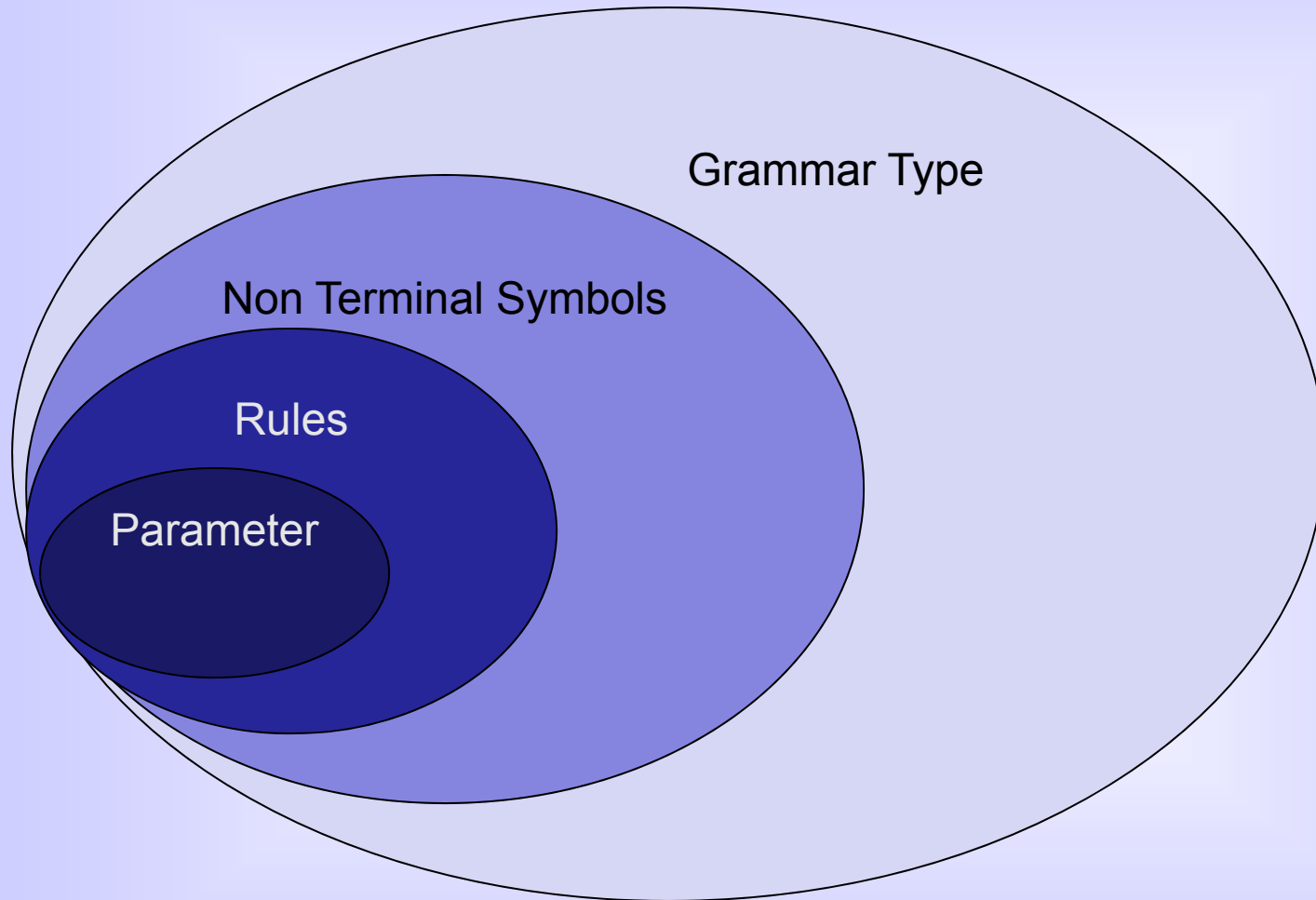


Model of non nested dependencies

Model RNA pseudoknots
Complex protein structures

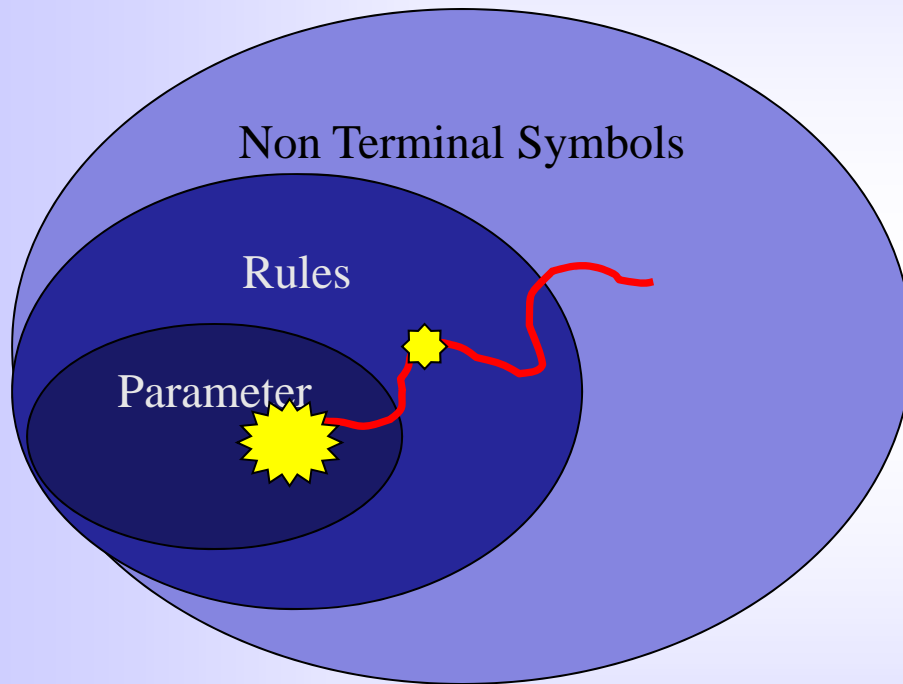


Evolve (where)?



For a given set of terminal symbols

Evolve (how)?



Exploring Mechanism:

- Random Walk
- Genetic Algorithm
- AI Search Algorithm

Scoring Mechanism:

- RNA – Energy function
- Protein – Energy function
- Alignment Likelihood

Meta Grammar

We can define a grammar to define grammars:

R1: $S \rightarrow RS (p_i) \quad | R (p_i)$

R2: $R \rightarrow NT \rightarrow L (p_i)$

R3: $L \rightarrow NT L(p_i) \quad | T L(p_i) \quad | NT(p_i) \quad | T(p_i)$

R4: $NT \rightarrow 'S'(p_i) \quad | 'L' (p_i) \quad | 'F' (p_i)$

R5: $T \rightarrow 's'(p_i) \quad | 'd' (p_i)$

Meta Grammar

Lets try to evolve the rule $S \rightarrow LS | L$ to $S \rightarrow LS | F$:

Meta Grammar:

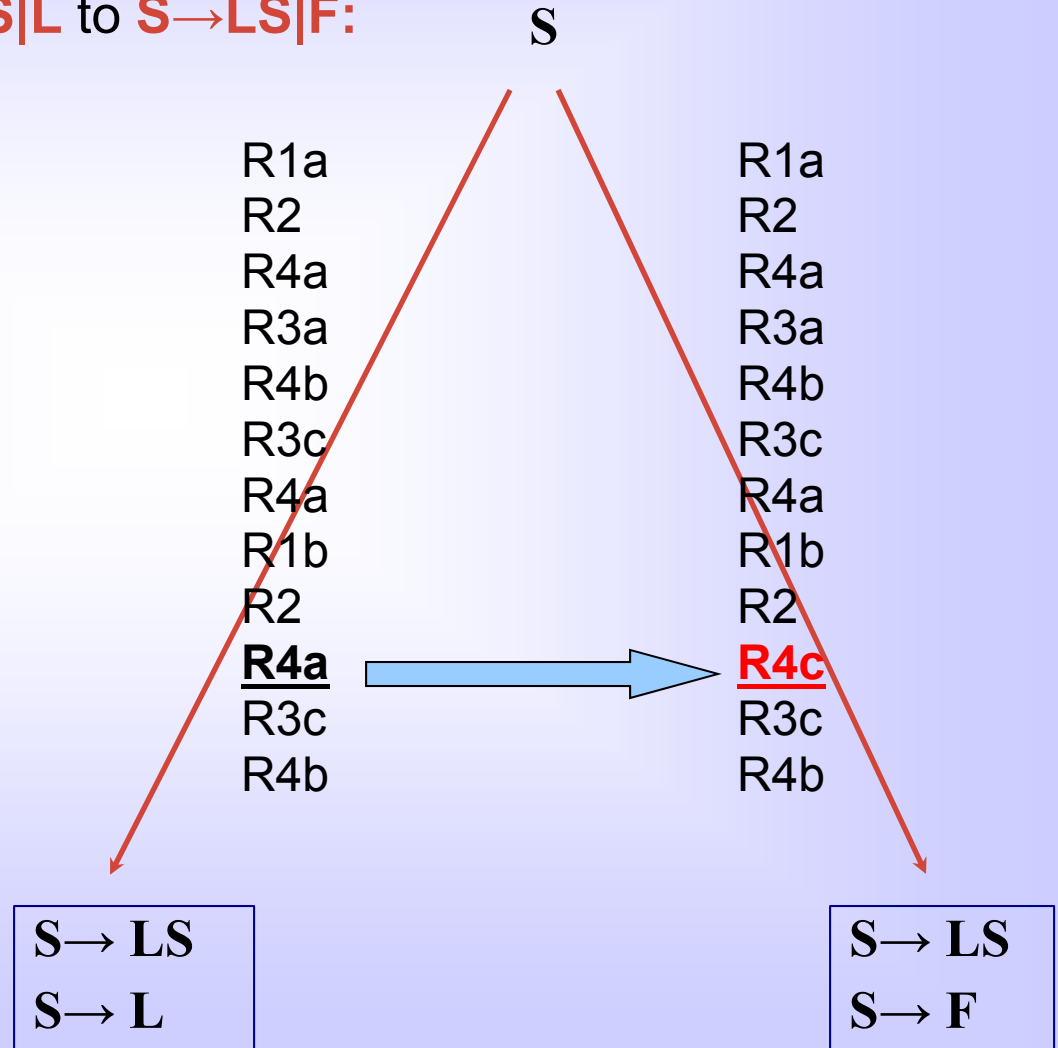
R1: $S \rightarrow RS | R$

R2: $R \rightarrow NT \xrightarrow{\text{green}} L$

R3: $L \rightarrow NT L | T L | NT | T$

R4: $NT \rightarrow \text{'S'} | \text{'L'} | \text{'F'}$

R5: $T \rightarrow \text{'s'} | \text{'d'}$



Remarks

- Search space too big - combinatorial explosion (maybe unfeasible without supervision)
- Easy to fall into a dead end (infinite grammar – Gödel's ghost)
- Do grammars formally describe the phenomena we want to model?