Main Ideas

- Grammars can give us insight about the underlying structure of a surprisingly large range of natural phenomena.
- Grammars can be used to help categorize problems and give you an idea of how difficult it is to solve with a computer.
- Grammars can be used to define the limitations of an algorithm.

Brief Overview of Grammars

Noam Chomsky - 1956 *Three models for the description of language* and 1959 *On certain formal properties of grammars*

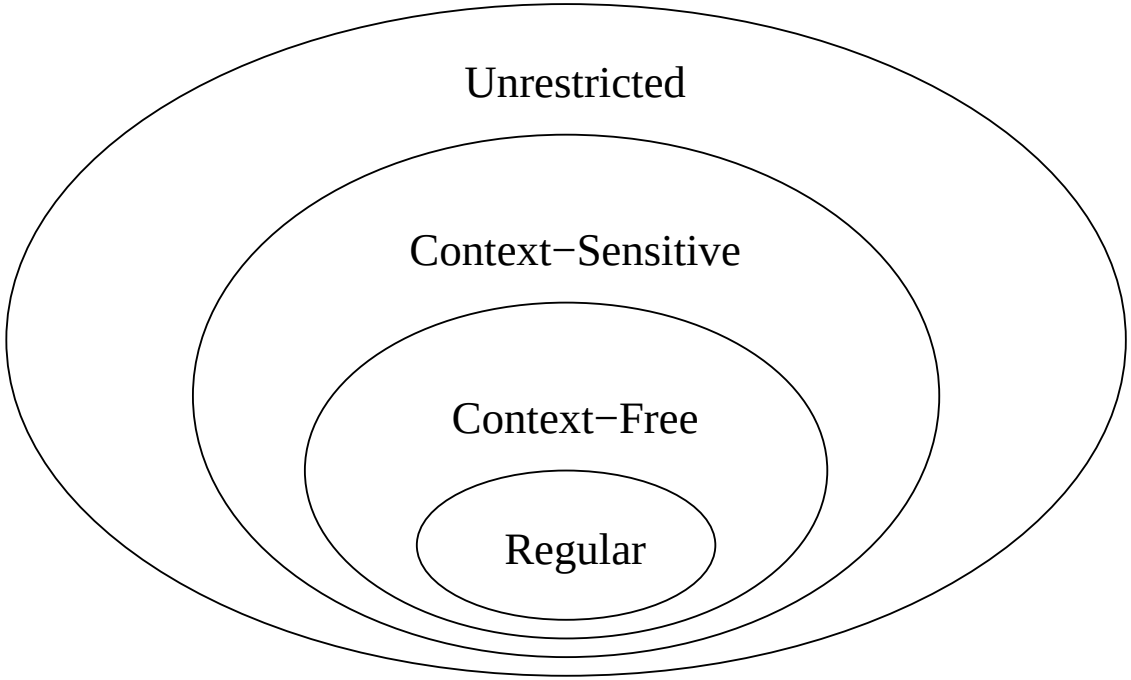
- Attempted to find the underlying structure in languages.
- Abstracted the concept of a grammar to rules that can generate phrases.
  For example: `The frog`

$$
\begin{aligned}
\text{PHRASE} &\longrightarrow \text{ARTICLE NOUN} \\
\text{ARTICLE} &\longrightarrow \{\text{a}, \text{an}, \text{the}, \ldots, \text{etc.}\} \\
\text{NOUN} &\longrightarrow \{\text{frog}, \text{chair}, \ldots, \text{etc.}\}
\end{aligned}
$$

Chomsky Hierarchy

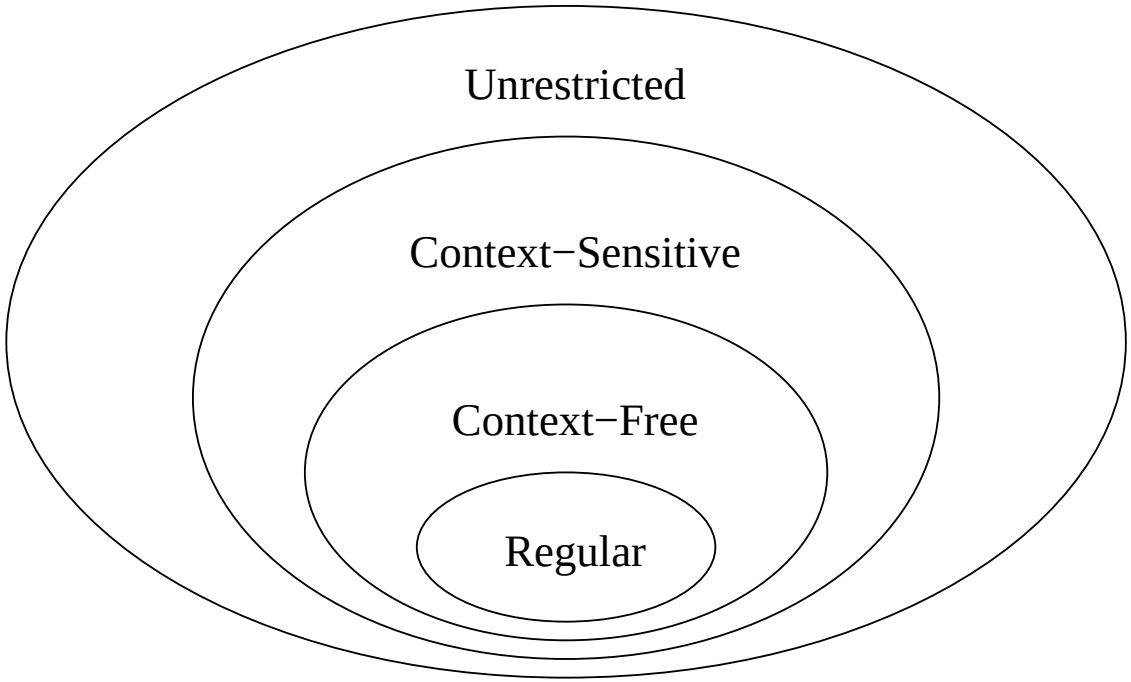
Grammars:

- **Regular:** Not very expressive, but easy for a computer to handle. For example: The frog.



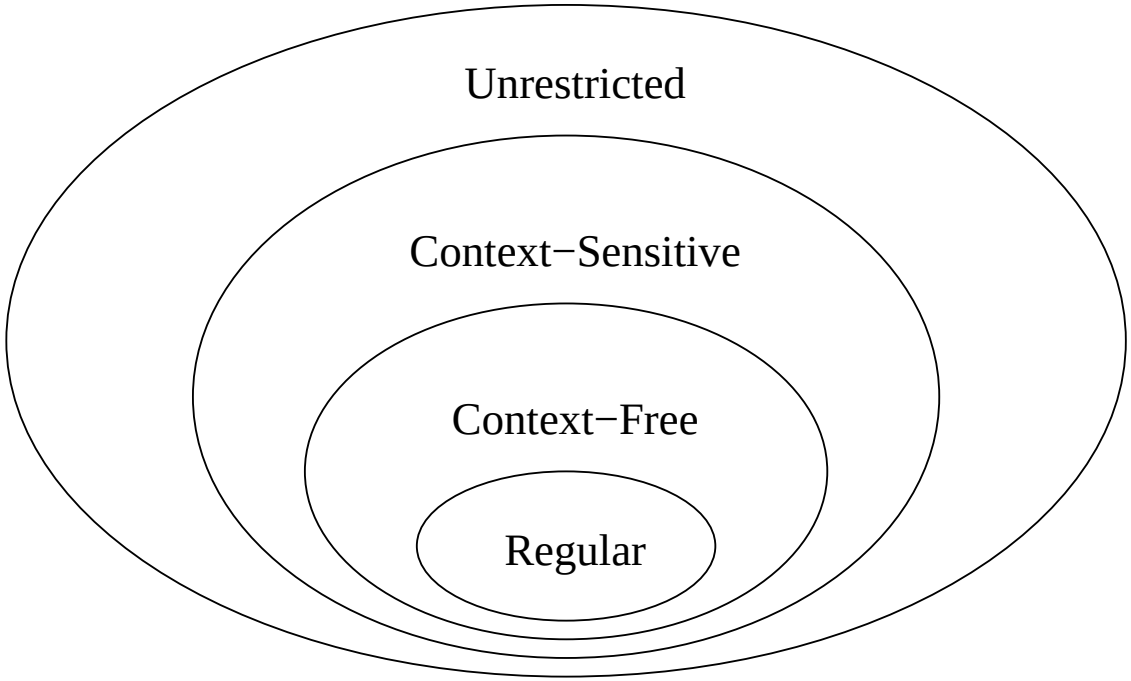
Chomsky Hierarchy

- **Context-Free:** Can express everything found in Regular Grammars plus can generate *nesting* (for example: (nested parenthesis)). These grammars are also relatively easy for a computer to handle.



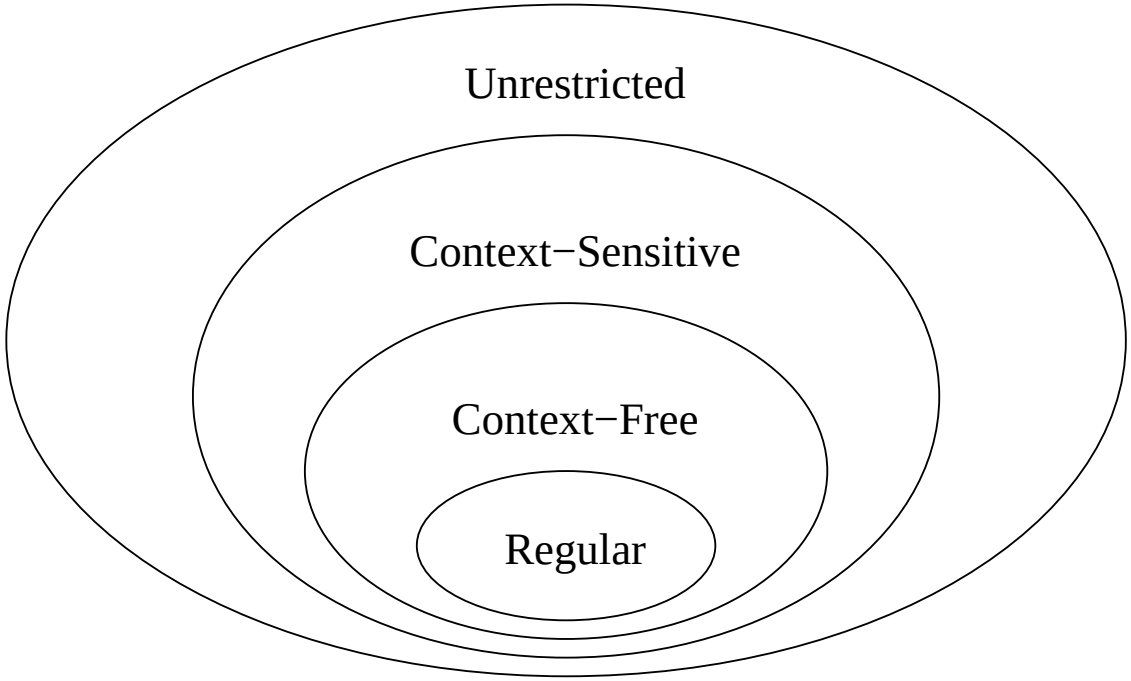
Chomsky Hierarchy

- **Context-Sensitive:** Can express Context-Free and Regular grammars as well as allowing for non-nested dependencies between terminals. Computers can only parse very short sequences generated with this kind of grammar. Large sequences can easily cripple the fastest computers of both today and tomorrow...



Chomsky Hierarchy

- **Unrestricted:** No holds barred. You can not even determine in advance if a computer can parse the sequence or not.



What are Grammars Useful For?

In Computing...

- Specification / interpretation of programming languages (C, Perl, HTML, Database Query Syntax, etc.)
- Pattern recognition (regular expressions in Perl)
- Speech and language processing

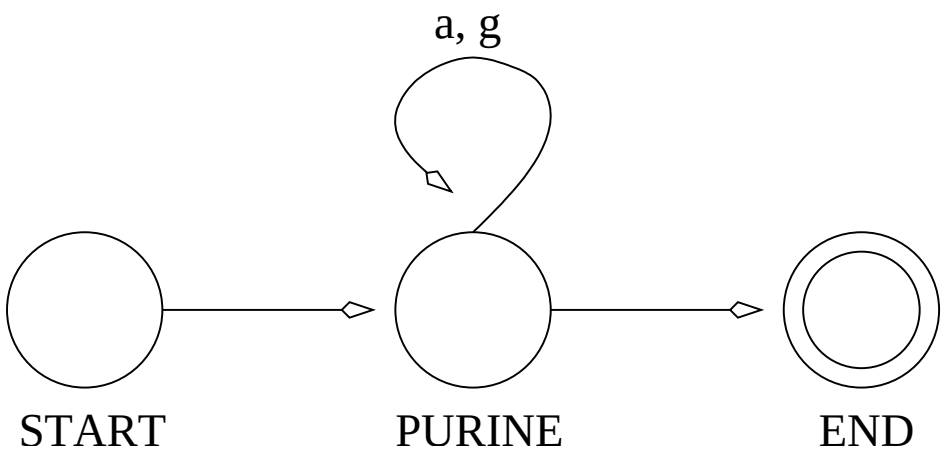
In Biology...

- Is a genome really a “Book of Life”?
  Does a genome have a grammar?

Grammars in Biology

A simple regular grammar that produces strings of purines:

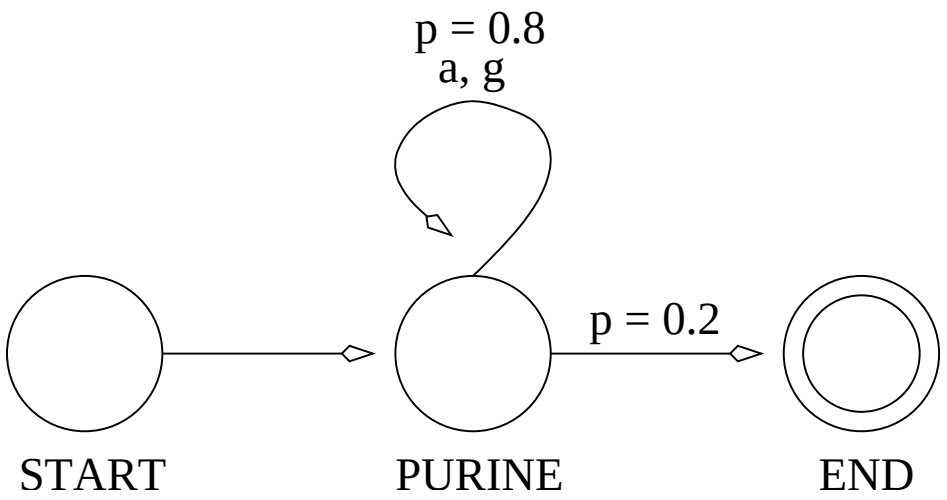
$$\begin{aligned} \text{START} &\longrightarrow \text{PURINE} \\ \text{PURINE} &\longrightarrow \{a, g\}\ \text{PURINE} \\ \text{PURINE} &\longrightarrow \text{END} \end{aligned}$$



Grammars in Biology

A stochastic regular grammar for strings of purines:

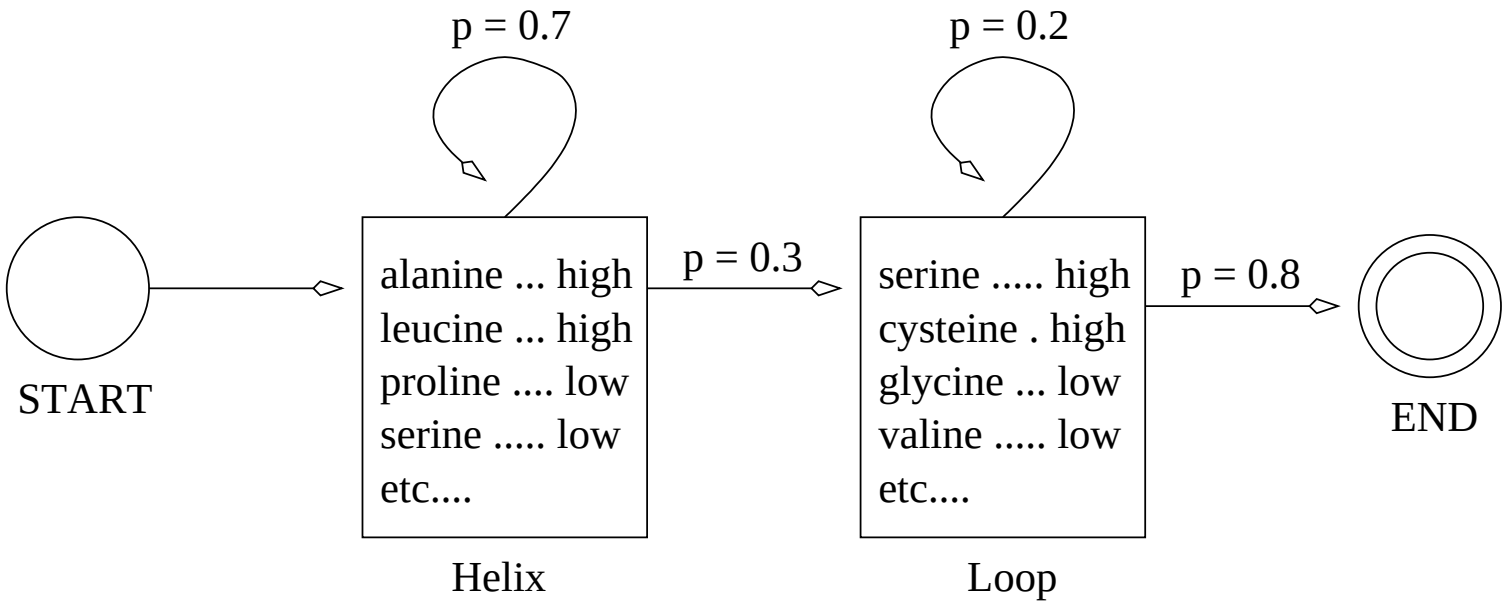
$$\begin{aligned} \text{START} &\xrightarrow{1} \text{PURINE} \\ \text{PURINE} &\xrightarrow{0.8} \{a, g\}\ \text{PURINE} \\ \text{PURINE} &\xrightarrow{0.2} \text{END} \end{aligned}$$



HMMs as Grammars

In fact, any Hidden Markov Model can be represented by a stochastic regular grammar.

- Simple HMM example from two weeks ago:



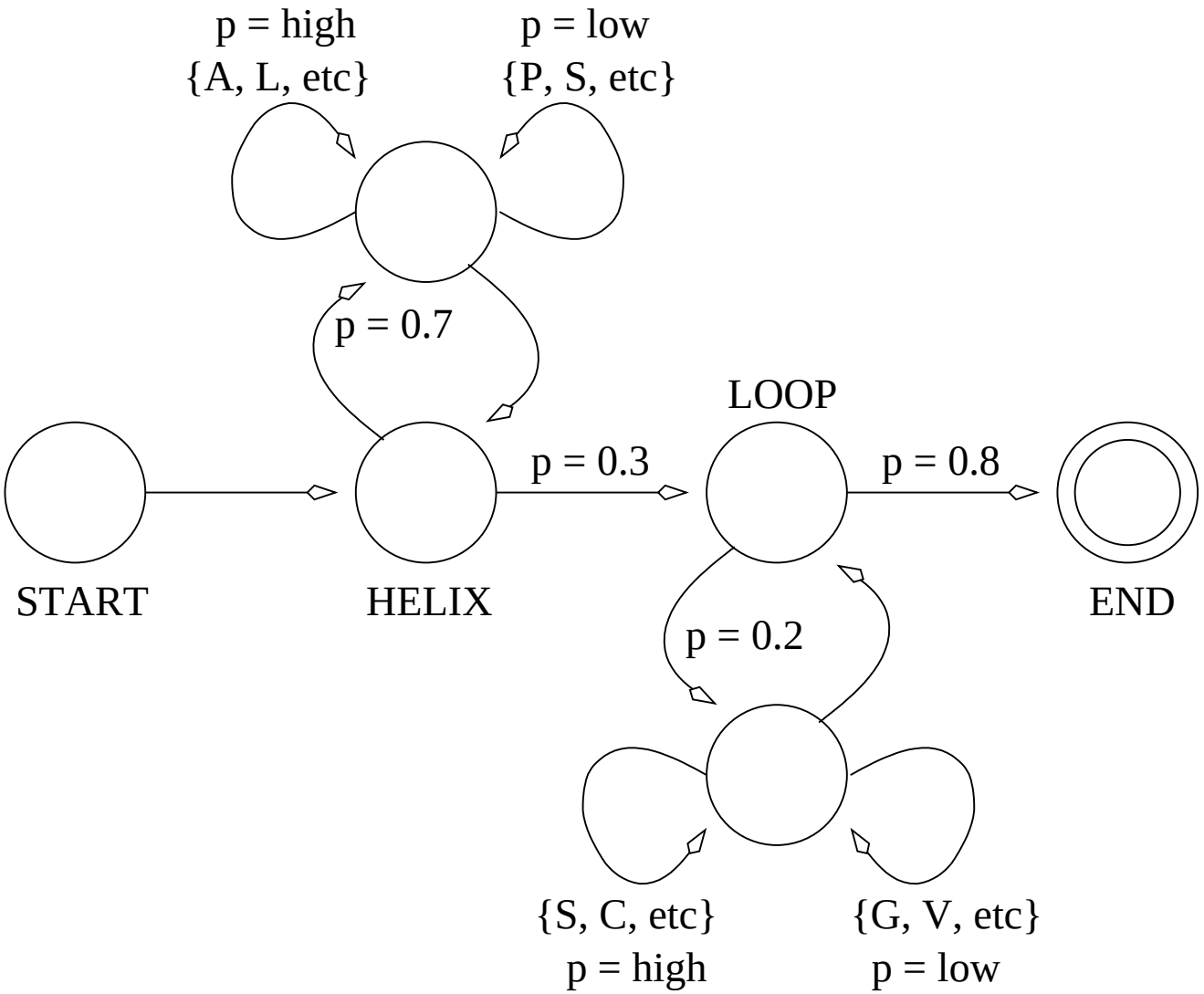
HMMs as Grammars

The equivalent stochastic regular grammar:

$$
\begin{aligned}
\text{START} &\xrightarrow{1} \alpha\text{-HELIX} \\
\alpha\text{-HELIX} &\xrightarrow{0.7} \alpha\text{-AA } \alpha\text{-HELIX} \\
\alpha\text{-AA} &\xrightarrow{\text{high}} \{\text{alanine, leucine, . . . , etc.}\} \\
\alpha\text{-AA} &\xrightarrow{\text{low}} \{\text{proline, serine, . . . , etc.}\} \\
\alpha\text{-HELIX} &\xrightarrow{0.3} \text{LOOP} \\
\text{LOOP} &\xrightarrow{0.2} \text{L-AA LOOP} \\
\text{L-AA} &\xrightarrow{\text{high}} \{\text{serine, cysteine, . . . , etc.}\} \\
\text{L-AA} &\xrightarrow{\text{low}} \{\text{glycine, valine, . . . , etc.}\} \\
\text{LOOP} &\xrightarrow{0.8} \text{END}
\end{aligned}
$$

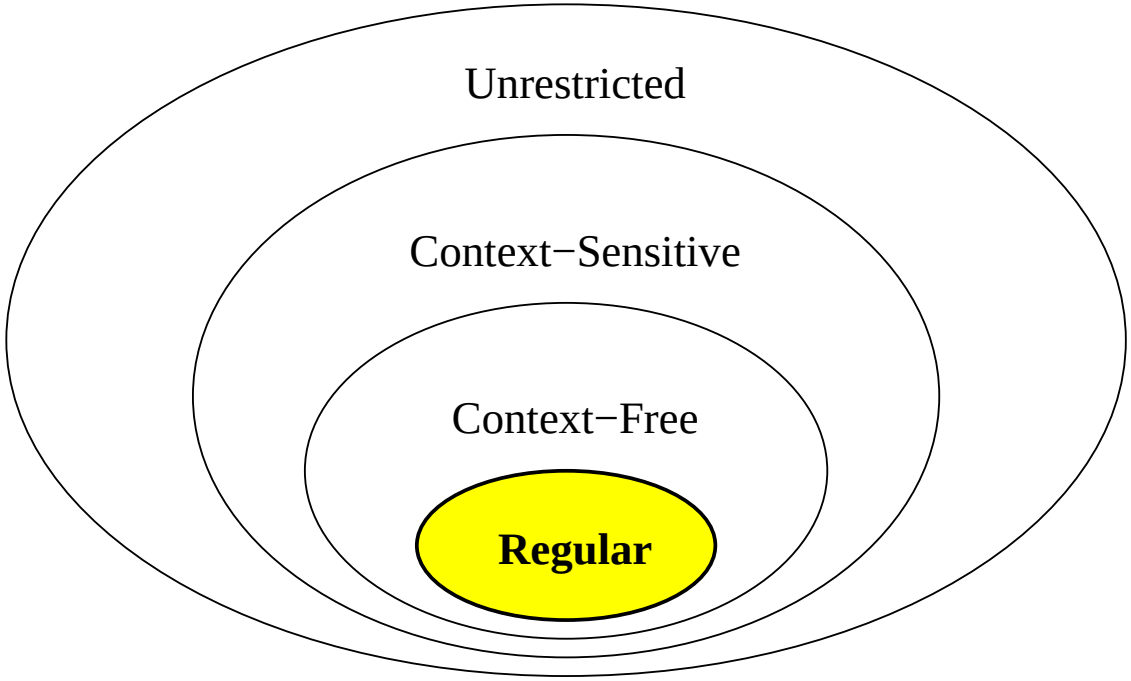
HMMs as Grammars

The equivalent stochastic finite state automaton:



HMMs as Grammars

The problem space defined for HMMs:



RNA Secondary Structures

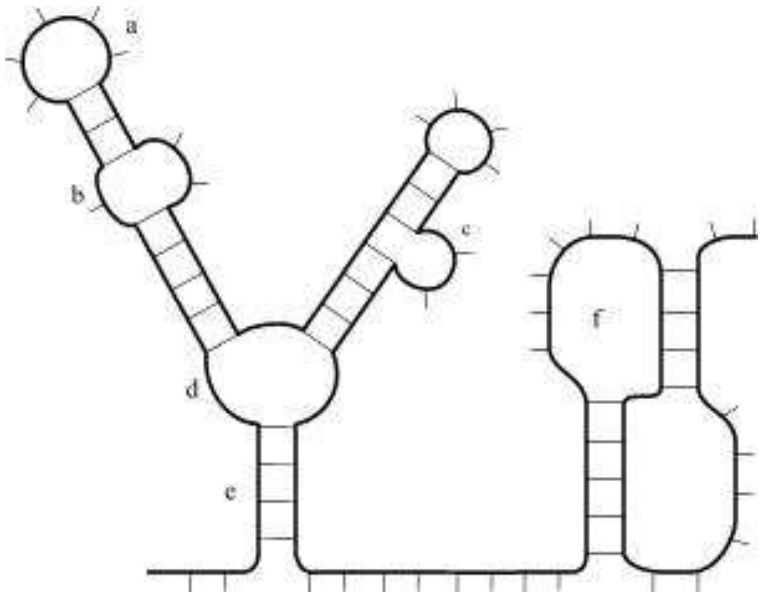


Illustration stolen from Peter De Rijk, http://rrna.uia.ac.be/ peter/doctoraat/struct.html

a: Stem/Loop - Hairpin

b: Inernal Loop

c: Bulge

RNA Secondary Structures

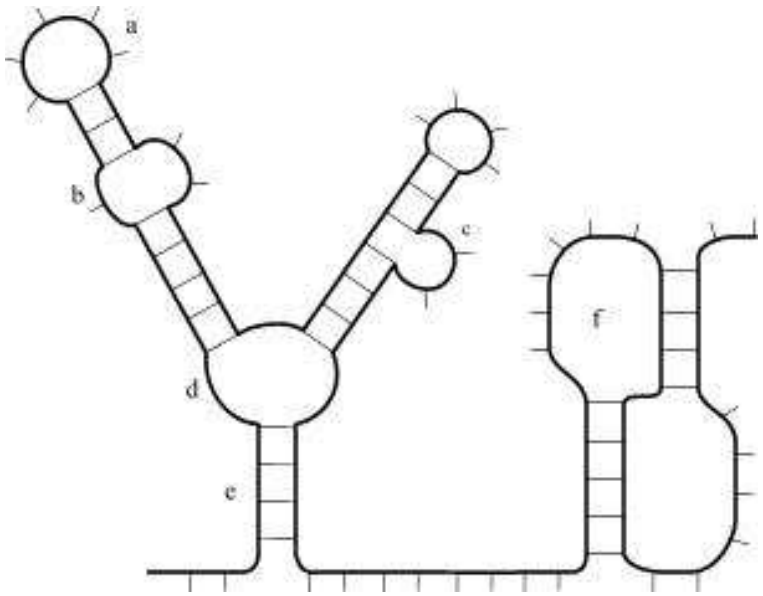


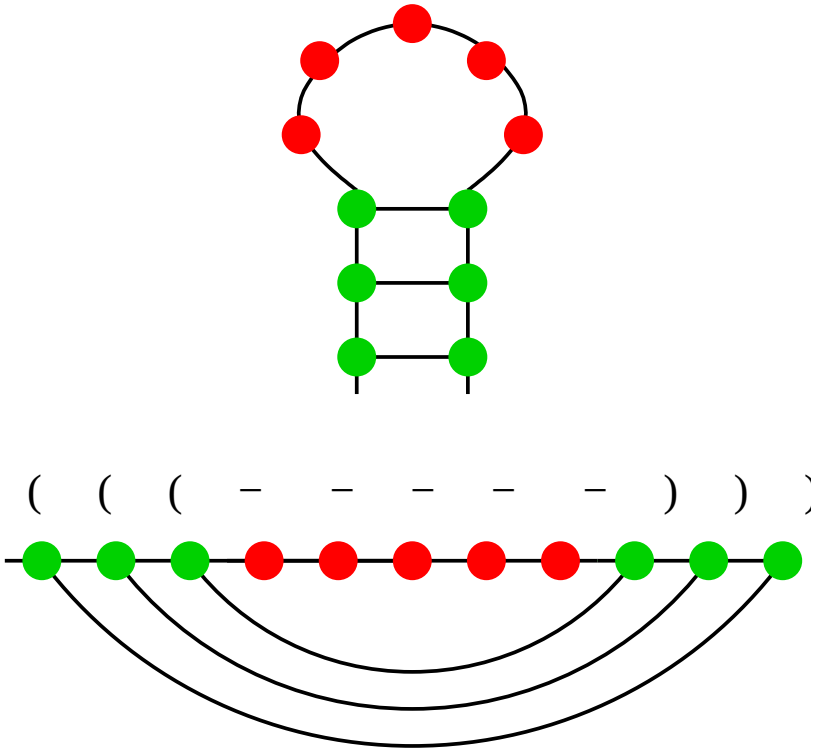
Illustration stolen from Peter De Rijk, http://rrna.uia.ac.be/ peter/doctoraat/struct.html

d: Junction

e: Stem (long range)

f: Pseudoknot

Hairpin



While it might be possible to model some hairpin configurations with a Markov model, we can not model all of them. We must use a context free grammar.

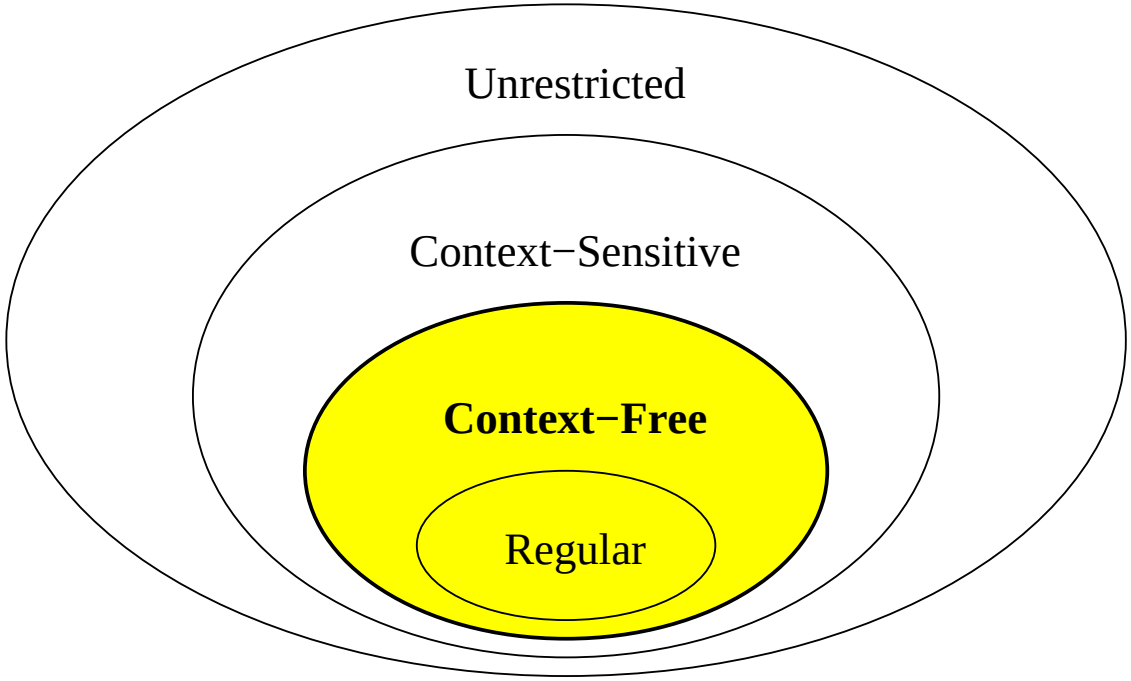
A Context-Free Grammar

A simple context-free grammar for hairpins:

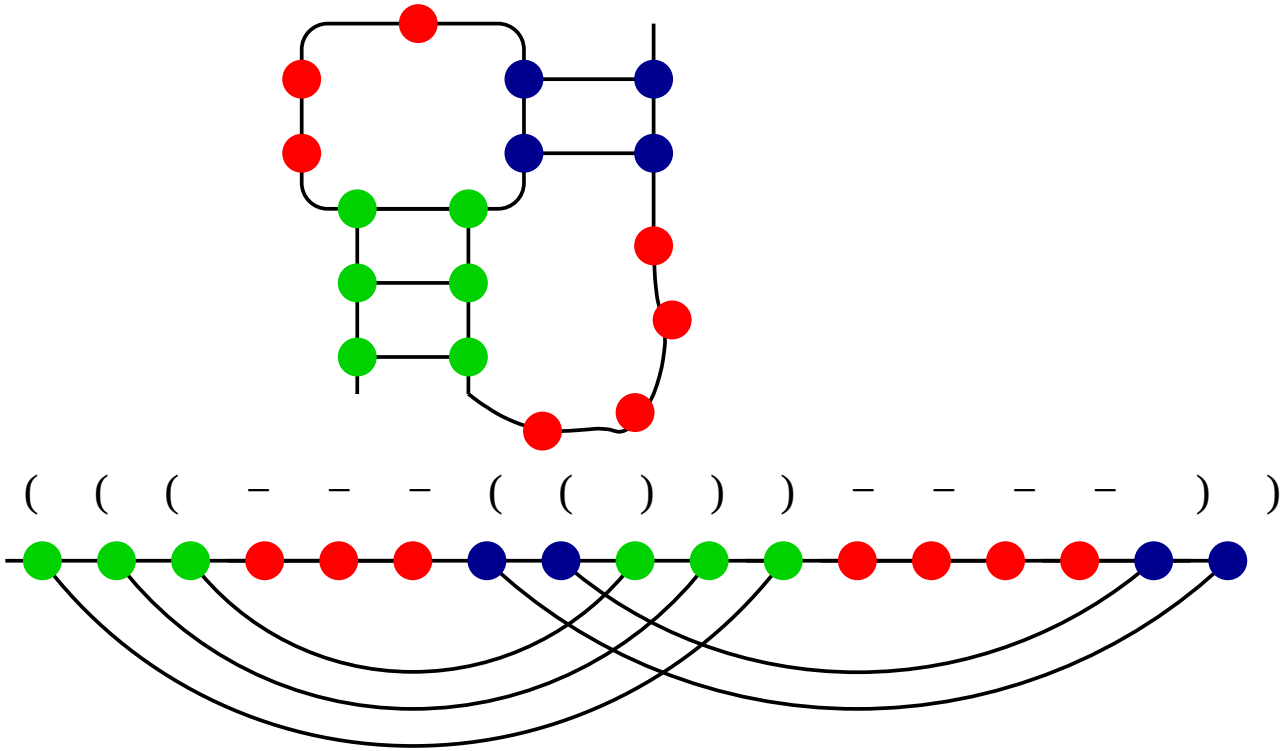
$$
\begin{aligned}
\text{RNA} &\rightarrow \text{BASE RNA} \\
\text{BASE} &\rightarrow \{a, c, g, u\} \\
\text{RNA} &\rightarrow \text{STEM RNA} \\
\text{STEM} &\rightarrow a\ \text{STEM}\ u \\
\text{STEM} &\rightarrow u\ \text{STEM}\ a \\
\text{STEM} &\rightarrow c\ \text{STEM}\ g \\
\text{STEM} &\rightarrow g\ \text{STEM}\ c \\
\text{STEM} &\rightarrow \text{RNA} \\
\text{RNA} &\rightarrow \text{END}
\end{aligned}
$$

Context-Free Grammars

The problem space defined for Context-Free Grammars:

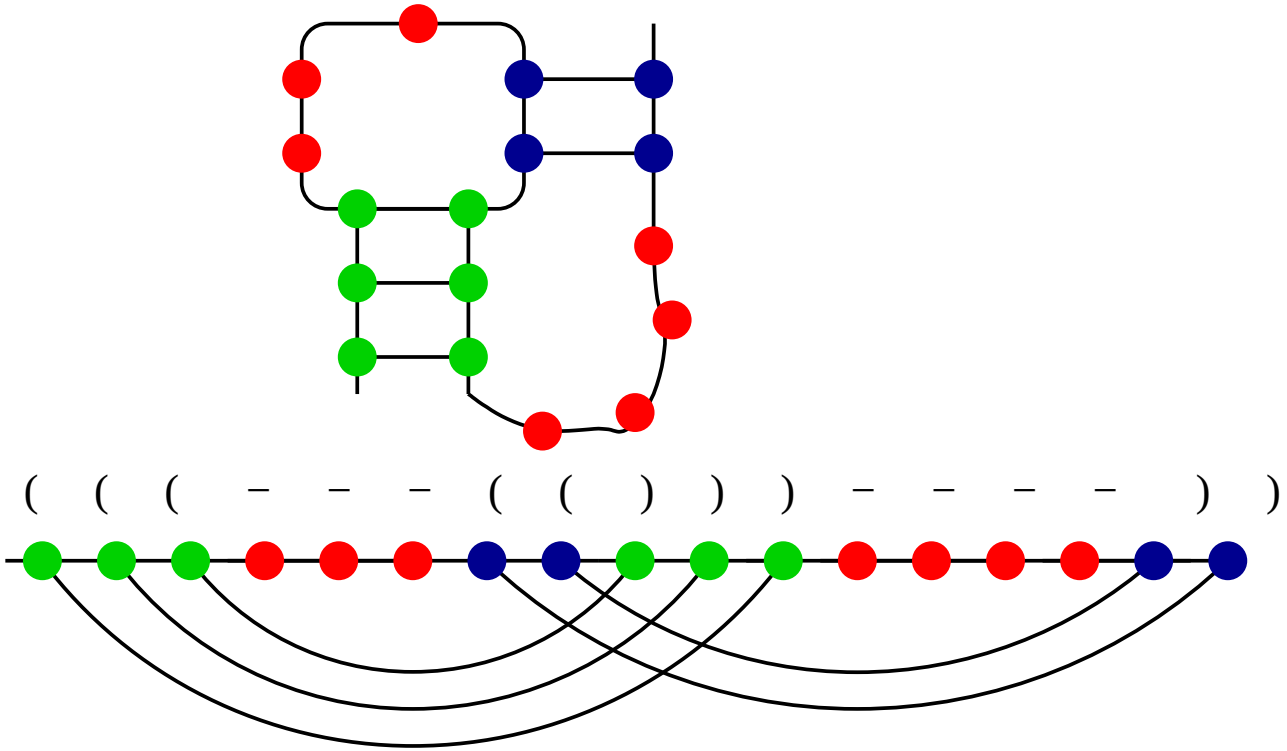


Pseudoknot



It is not possible to model all possible pseudoknot configurations with a context free grammer. However, it can be done with a context sensitive grammar.

Pseudoknot



The problem is that context free grammars take too long to parse (the program might not finish in our lifetimes). The algorithm in the paper chooses to model only a subset of possible configurations.

Results

- **tRNAs:** 25 were selected at random from the Sprinzl tRNA database. The authors emphasize that their algorithm did not find any spurious pseudonots in any of the predicted structures (no false positives).

  Of 24 with cloverleaf foldings, 24 cloverleaf folds were found. 15 of them were completely consistent with their proposed structures. MFOLD only found 19 cloverleaf foldings with only 14 being completely consistent with their proposed structures.

  The authors emphasized that their algorithm is at least comparable with MFOLD's.

Results

- **HIV-1-RT-ligand RNA pseudoknots:** They tested 63 SELEX-selected pseudoknotted sequences (between 34 and 47 bases in length). The program found 54 foldings that agreed exactly with structures derived by comparative analysis.

Results

- **Viral RNAs:** Here they tested their program on a grab bag of viral RNAs including TMV, TYMV (turnip yellow mosaic virus) and Ribozymes from the Hepatitis Delta virus (HDV). The program did pretty well (sorry for the sketchy details here.)

NOTES:

- The authors mention that their method is sensitive to the thermodynamic paramters that you give it. (Increasing the accuracy of these paramters should increase the accuracy of the method)
- There is no decent thermodynamic information for pseudoknot configurations.
- The algorithm can only predict structures for RNAs with less than 140 bases (4 years ago - considering that computers are probably several times faster today, it could probably handle sequences with up to 190 bases).

NOTES: (continued)

- The aproximation made on the bottom of page 337, which cause this algorithm to only recognize a subset of the context-sensitive grammars, prevents it from handleing parallel β-sheet configurations.

Rivas, E. and Eddy, S.R. (1999) A dynamic programming algorithm for RNA structure prediction including pseudoknots. *J. Mol. Biol*., **285**, 2053-2068.

Main Ideas

- Grammars can give us insight about the underlying structure of a surprisingly large range of natural phenomena.
- Grammars can be used to help categorize problems to give you an idea of how difficult it is to solve with a computer.
- Grammars can be used to define the limitations of a given method.