
BIOINFORMATICS
2019

# The Composition of Dense Neural Networks and Formal Grammars for Secondary Structure Analysis

Polina Lunina, **Semyon Grigorev**

JetBrains Research, Programming Languages and Tools Lab
Saint Petersburg University

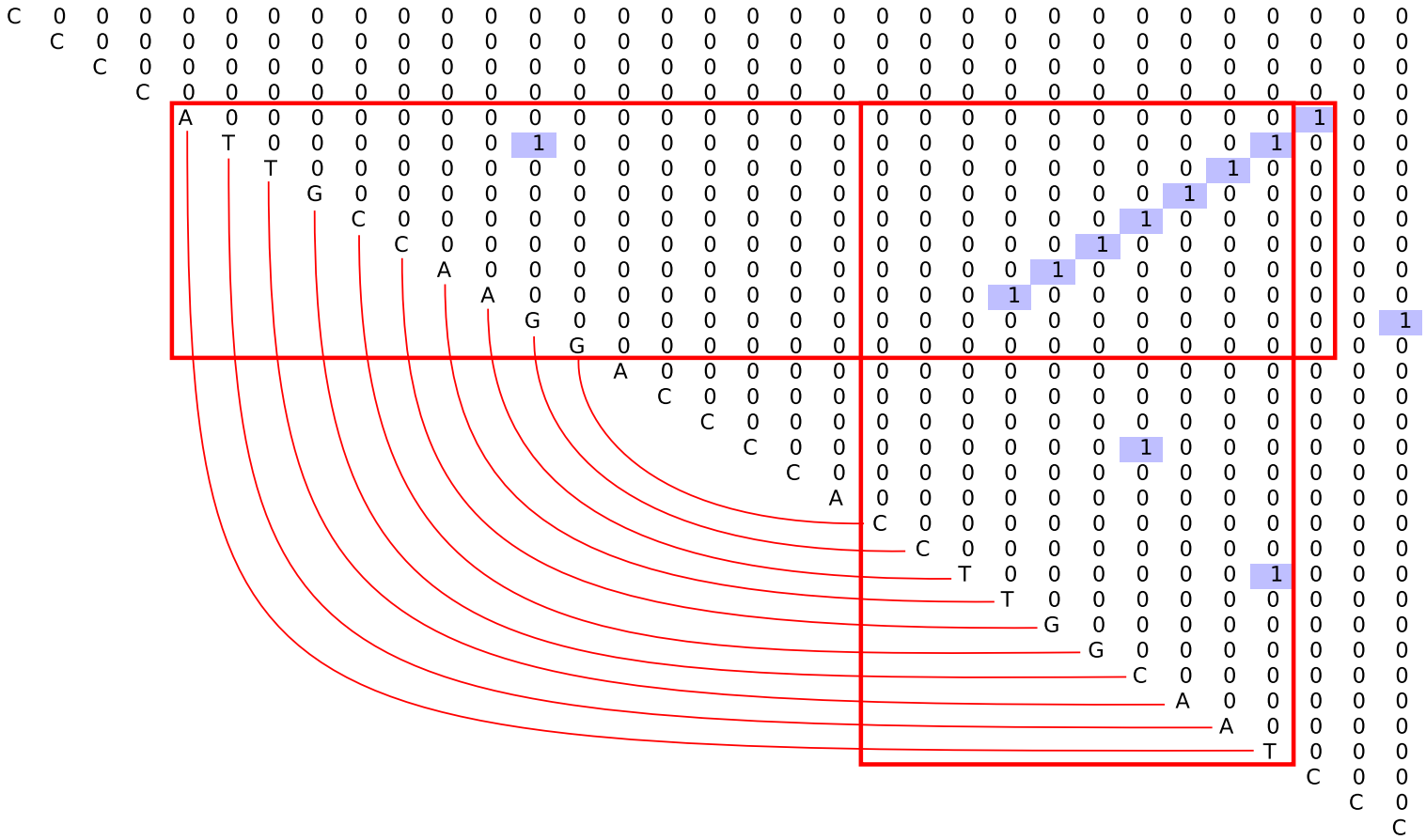
Febrary !!!, 2019

# Grammar

```
s1: stem<s0>
any_str : any_smb*[2..10]
s0: any_str | any_str stem<s0> s0
any_smb: A | T | C | G
stem1<s>:                    \\ stem of height exactly 1
      A s T | G s C | T s A | C s G
stem2<s>:                    \\ stem of height exactly 2
      stem1< stem1<s> >
stem<s>:                     \\ stem of height 3 or more
      A stem<s> T
    | T stem<s> A
    | C stem<s> G
    | G stem<s> C
    | stem1< stem2<s> >
```

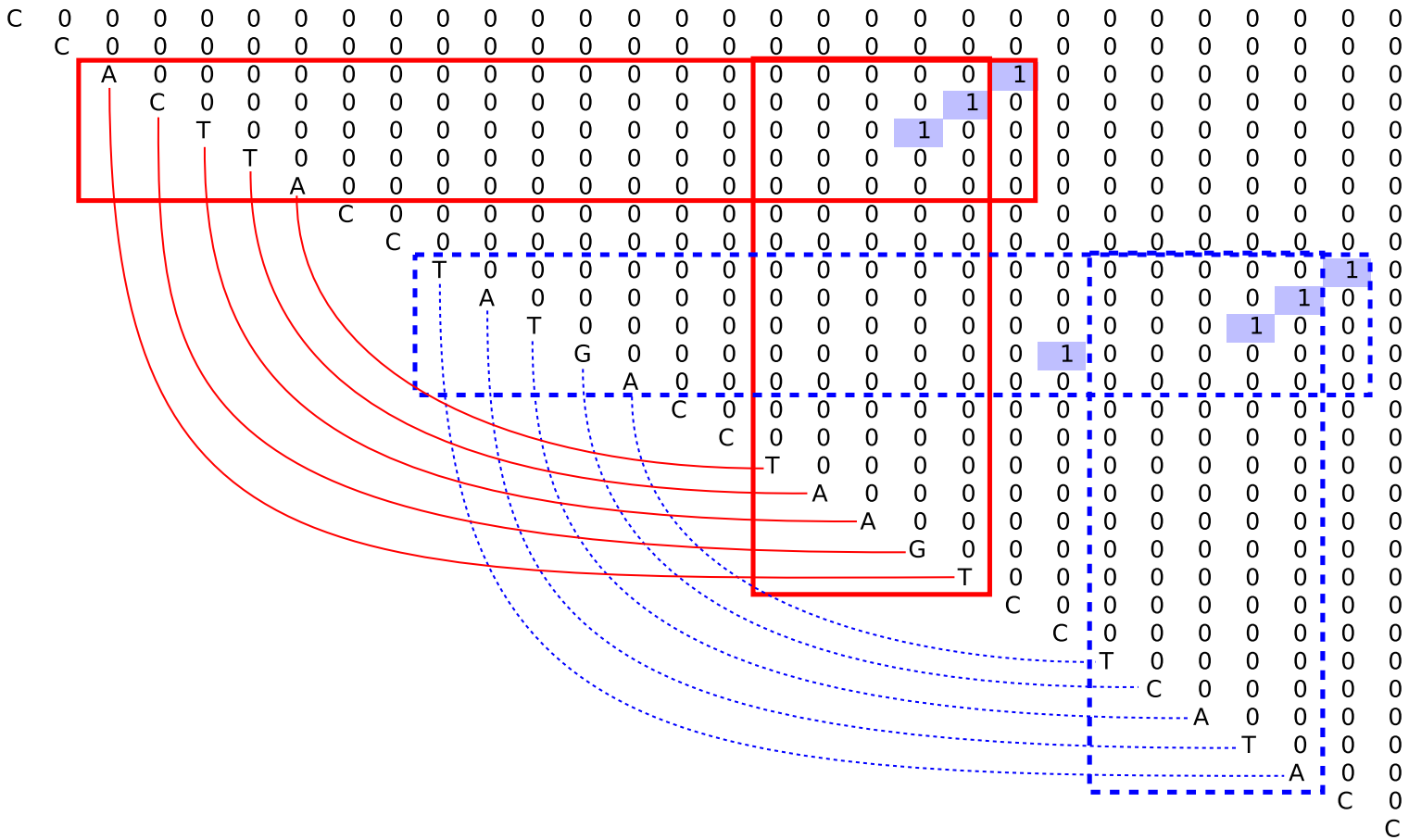
# Example 1: Stem

$$\omega_1 = \texttt{CCCCATTGCCAAGGACCCCACCTTGGCAATCCC}$$



# Example 2: Pseudoknot

$$\omega_2 = \texttt{CCACTTACCTATGACCTAAGTCCTCATACC}$$

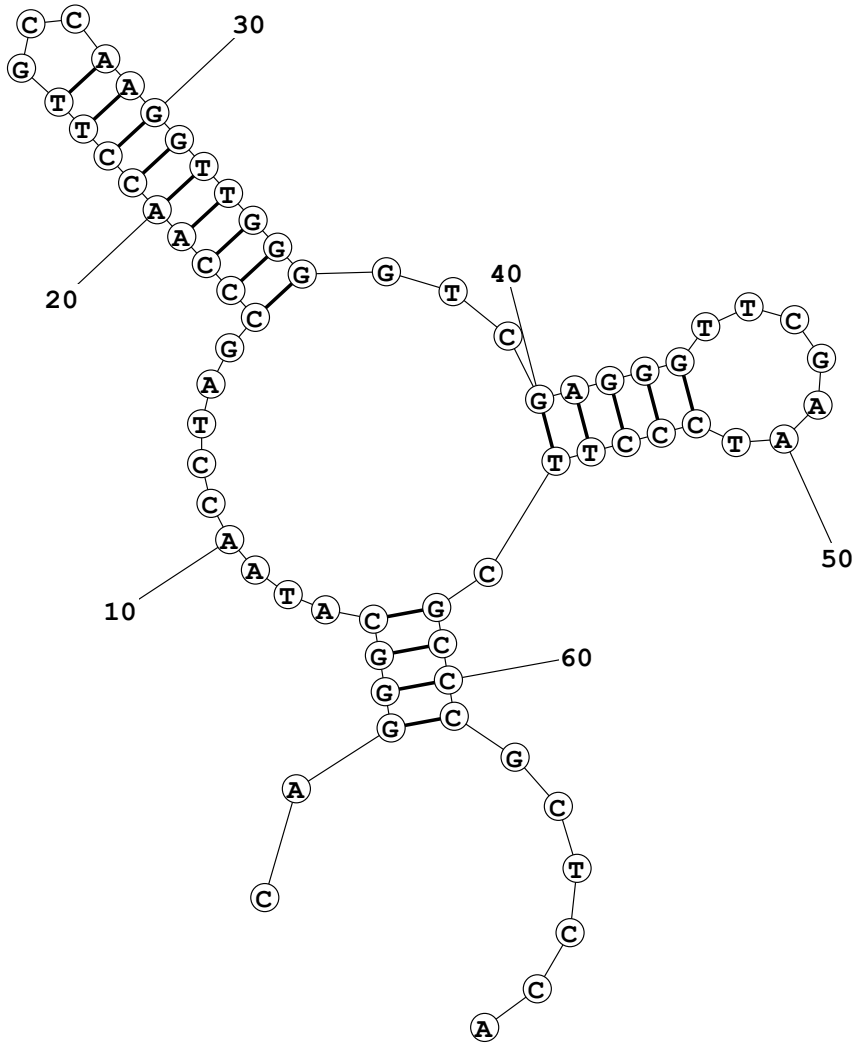


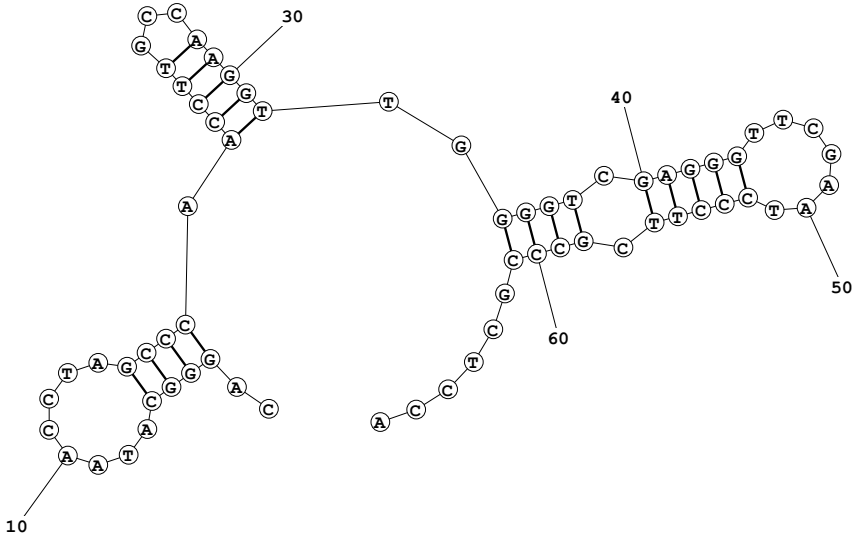
# Example 2: Pseudoknot

$$\omega_3 =$$

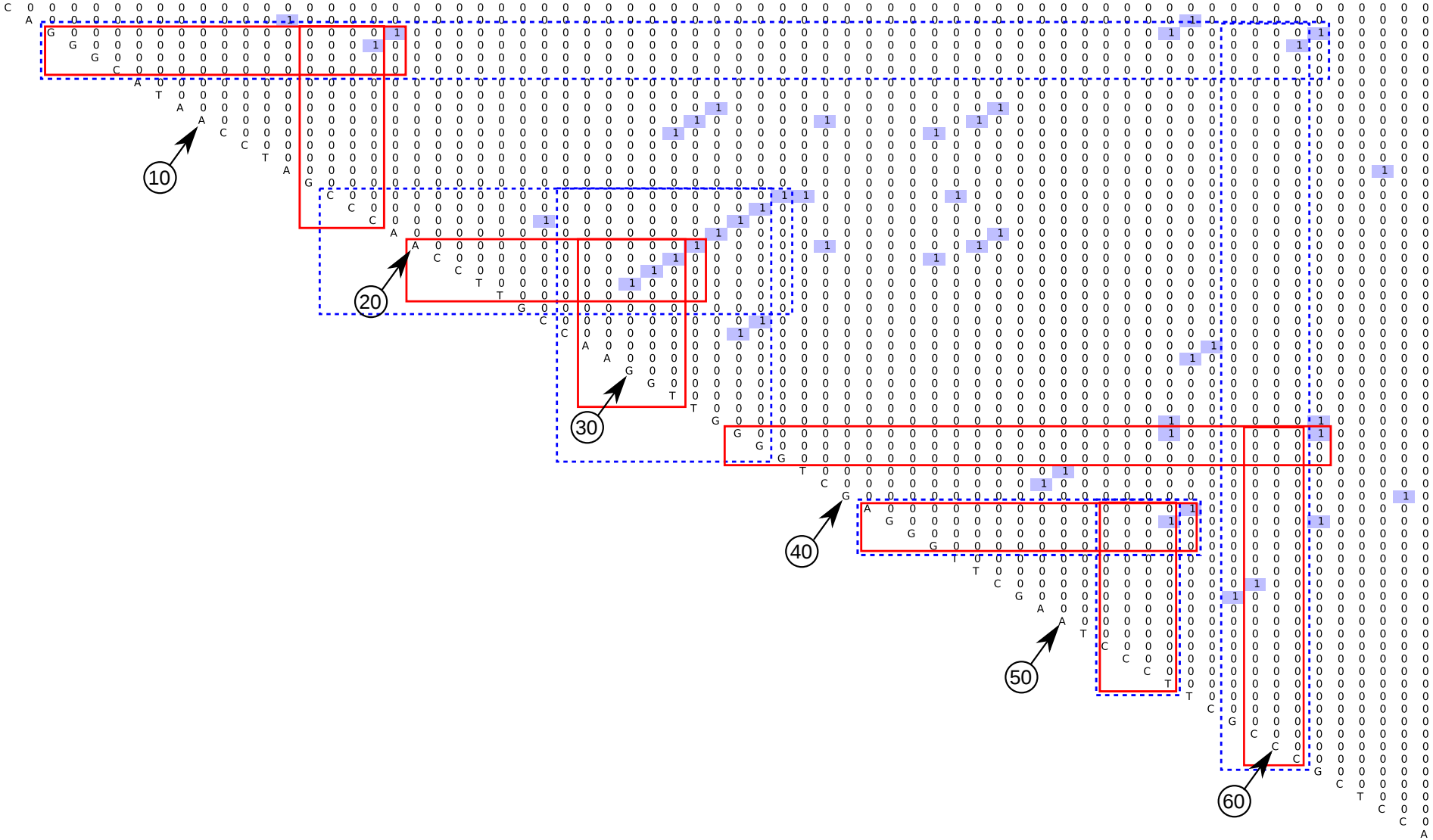
CAGGGCATAACCTAGCCCAACCTTGCCAAGG

TTGGGGTCGAGGGTTCGAATCCCTTCGCCCGCTCCA



# Example 2: Pseudoknot



# F# type providers

- Compile-time metaprogramming for types creation
  - Type provider is a **function which constructs type**
- Design-time features in IDE
  - Completion
  - Type information
- Used for type-safe integration of external data with fixed schema
  - Type providers for XML, JSON, INI
  - R, SQL

# Example of INI type provider

```
[Section1]
intSetting = 2
stringSetting = stringValue
[Section2]
floatSetting = 1.23
boolSetting = true
anotherBoolSetting = False
emptySetting =
stringWithSemiColonValue = DataSource=foo@bar;UserName=blah
```

# OpenCL C type provider

- We want to construct type-safe wrapper for existing library
- OpenCL standard declares source-level distribution with in place compilation
  - \+ We can work with source code, not with binaries
  - – Existing library is a set of files includes *.h files
- It is enough to process functions signatures

# OpenCL C type provider: architecture

# OpenCL C type provider: architecture

Yes, it is typical type provider

# Limitations

- Only (small) subset of OpenCL C
  - *.h files are not supported
  - preprocessor is not supported
  - only small subset of syntax is supported
- Very simple C to F# type mapping

# Examples

# Future work

- Improve OpenCL C support
  - Lexer and parser
  - Translator
  - Types mapping
  - Headers files processing
  - ...
- Unify kernels on client side
  - Currently native Brahma.FSharp's kernel and kernel loaded by type provider are different types
- Improve mechanism of kernels composition

# Summary

- F# OpenCL C type provider
  - Type-safe integration of existing OpenCL C code in F# applications
  - Proof of concept

- Source code on GitHub:
  `https://github.com/YaccConstructor/Brahma.FSharp`
- Package on NuGet:
  `https://www.nuget.org/packages/Brahma.FSharp/`

# Contact Information


- Semyon Grigorev: s.v.grigoriev@spbu.ru
- Kirill Smirenko: k.smirenko@gmail.com


- Brahma.FSharp:
  https://github.com/YaccConstructor/Brahma.FSharp

Thanks!