DNA#

Programming for life

WHO ARE WE?

GURUS

The project is KILLER. In all likelihood, you will fail if you don't assemble the Avengers to be your team. Make sure you put together a strong team which has time to commit to this project. The project is structured such that one person (the most responsible or the least busy) always ends up doing over 50% of the work. We've looked at other groups and previous teams and this is the case through out. I think Edwards should take a look at this and somehow fix the project so that it's more fair.

Motivations

- Scientists and geneticists are seeking to “engineer” DNA and develop complex computational tools
- Only tools to process genetic data are libraries within other languages (e.g. BioPython)
  - Large overhead
  - Low customizability
- DNA is rapidly being explored as an alternate form of data storage
  - “Capacity approaching DNA storage” - Yaniv Erlich (Columbia University) et al.
  - “Microsoft experiments with DNA storage: 1,000,000,000 TB in a gram” - Peter Bright

First...a little bit of biology

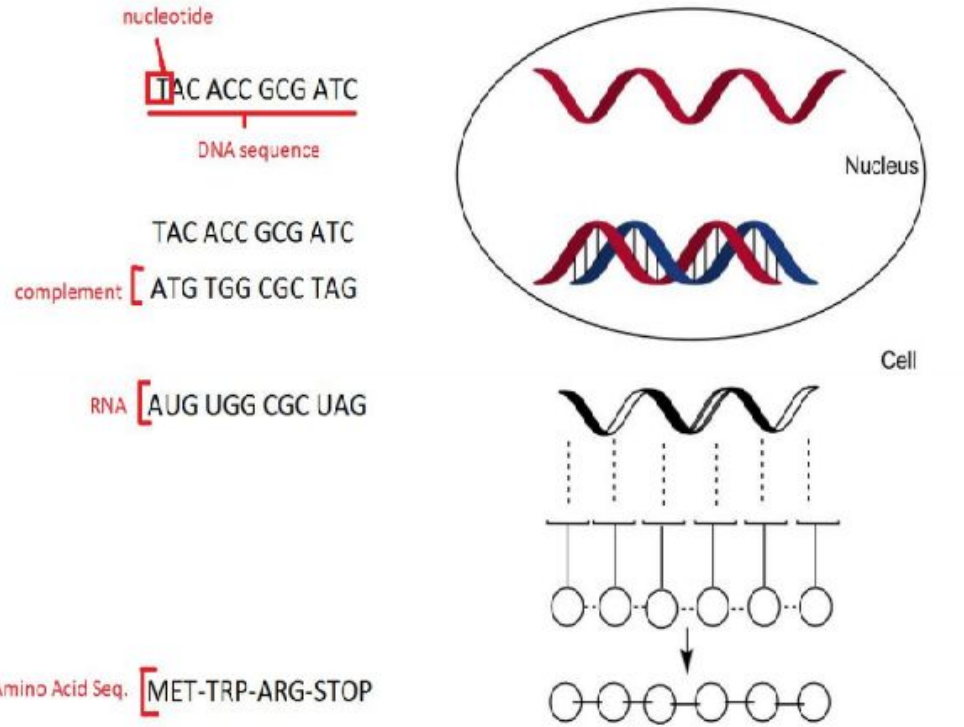
DNA

Transcription

RNA

Translation

Protein



DNA# In a slide

```
//This is our program!
DNA sample = readFASTA("sample.txt");
print(sample);
        // aggc
RNA sample2 = sample->;
int i = 0;
for i = 0; i< sample.length; i++
        then
                print(sample[i]);
        end
        //uccg
Pep protein = sample2+>;
```

```
DNA David = "atgc";
DNA Watkins = "cgta";
DNA CoolGuy = David + Watkins;
print(CoolGuy);
        //atgccgta
```

Data Types

- Native types from C
  - int, bool, char,
- Complex types
  - Strings, Arrays
- DNA specific types
  - DNA, RNA, Nuc, Pep, AA

Some friendly inbuilt operations

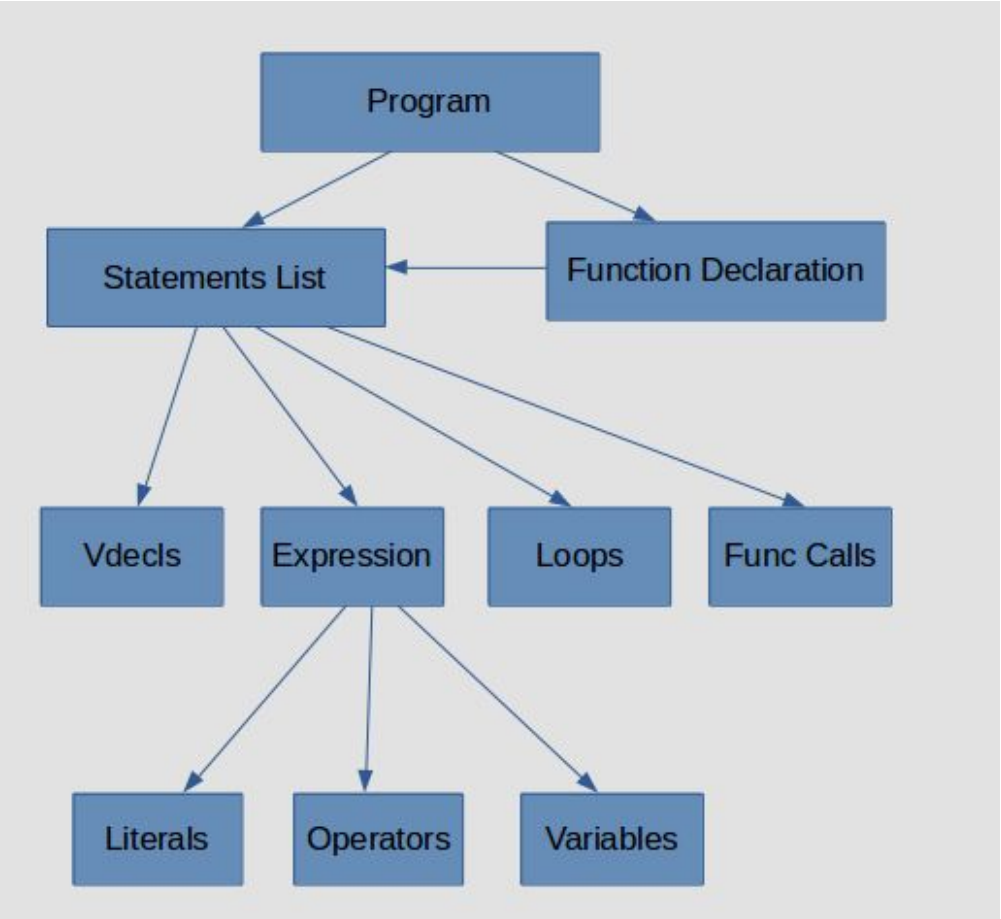
- DNA specific operators
  - DNA -> :transcribe
  - RNA +> : translate
- String/DNA friendly operations
  - Overloaded + operator for string types
  - .length function to get size of complex types and arrays
- Generalized print function
  - Can print any type!

Key Features

- Statically typed
- Statically scoped
- Fluid data type conversion (e.g. DNA -> RNA -> peptides)
- Natively supported string functions (string1 + string2)
- No global variables
- All memory stored on stack

Third Party Software

Abstract Syntax Tree



DNA# Architecture

- **Built-in C lib & Elegant ext_func_lst**

Our language has one built-in C-lib, and a series of helper functions. It is very easy to use C-library. There are only three steps to add one C-function.

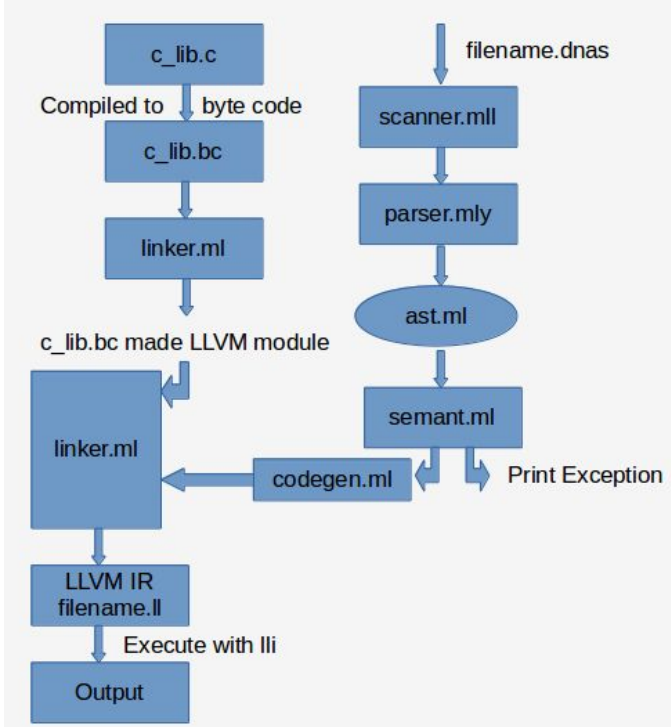
(1) Add your function in c_lib.c.

(2) Register the new function in ext_func_lst table.

(3) Make project, then magic happens.

- **Pseudo-Main**

Since DNA# is a script style language, it starts at the first line of *.dnas file. In ‘codegen.ml’, we build a pseudo-main function to collect all stmts outside other defined functions and make it the main func in LLVM.




```
let ext_func_lst=[
  (*NOTICE : the sequence of arg list has to be reverse order of origin C-fu
  {name="test"          ;ret=i32_t      ;arg=[|i32_t;i32_t|]          };
  {name="printf"        ;ret=i32_t      ;arg=[|L.pointer_type i8_t |]  };
  {name="complement"    ;ret=str_t      ;arg=[|str_t|]                };
  {name="transcribe"    ;ret=str_t      ;arg=[|str_t|]                };
  {name="translate"     ;ret=str_t      ;arg=[|str_t|]                };
  {name="translate2"    ;ret=str_t      ;arg=[|str_t|]                };
  {name="concat"        ;ret=str_t      ;arg=[|str_t; str_t|]         };
  {name="strlength"     ;ret=i32_t      ;arg=[|str_t|]                };
  {name="readFASTAFile" ;ret=str_t      ;arg=[|str_t|]                };
```

Testing Suite

- Unit Testing
  - Identifiers (if, for, while)
  - Standard, primitive, and complex data types (dna, rna)
  - Control flow
  - Functions
  - Literals (Nuc, AA, Integer, Double, Bool, Character, String)
- Integration Testing
-
- System testing

DEMO

- Find longest subsequence amongst two DNA sequences and print protein that would be generated
  - Mutations
  - DNA alignment and sequencing

Applications

- DNA encoding (Huffman encoding, DNA fountain, etc.)
- Yaniv Erlich/NY Genome Center
- Still using biopython and hacked together tools with large overhead (personal experience)
- iGEM and personal experience with that

Future Directions

- Optimizing the transcribe/translate using encoding schemes (e.g. DNA Fountain, Huffman)
- Supporting variable nucleotides and file types
- Supporting addition of libraries (e.g. a file i/o library for different file formats)
- Incorporating type associated global constants, such as weight, to make computation easier

Questions

References

Funk Programming Language

Dice Programming Language

OCaml Documentation