

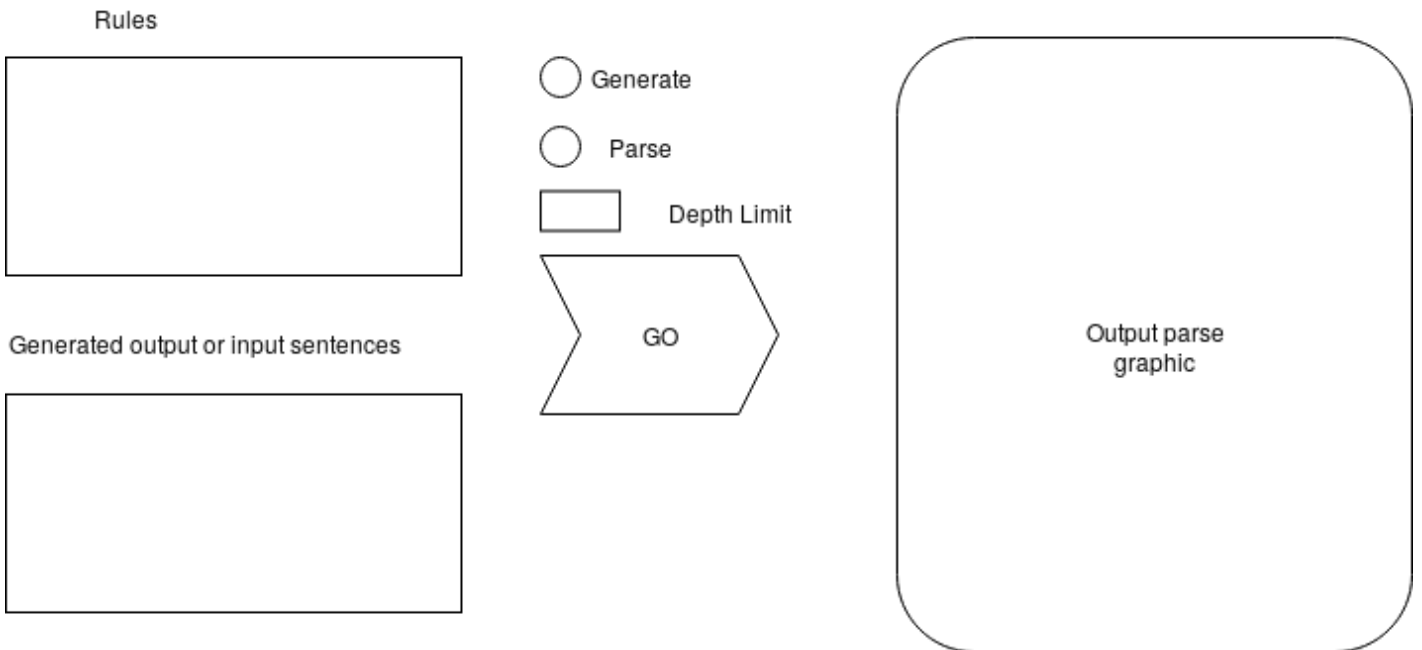
# Parser Utility

I'm developing a parser for the L203 classes that can be fed rules

- [Design specs](#)
- [Gameplan \(in order of priority\)](#)
- [Handling Optionality](#)
- [Hai's notes](#)
- [Genie tidbits](#)

# Design specs

## Original Page Layout



If generate is selected, the lexical entries and syntactic rules are used to generate a number of random sentences in the bottom left box. If parse is selected, the sentence/sentences in the bottom left box are parsed using the rules specified and the resulting svg is put in the box on the right. Depth limit is used to prevent recursion from causing problems in the parse generated.

There was also a timeout so that if it took too long to parse the sentence, it was just aborted.

## Rule format

The rules input could be of two types, lexical rules and syntactic rules.

# Lexical rules

These rules consist of a lexical category label, a colon and then a thing that is part of that category.

for example `N : dog`

in addition, a set can be specified

`N : {dog, cat}`

this means that both `dog` and `cat` are `N`'s.

# Syntactic rules

These rules consist of a syntactic category label, an arrow and then component constituents.

```
NP -> Det N
```

# Scope of rules

Should be able to handle

- Optionality using parenthesis
- Repetition using \*
- Right hand sides should be able to be non-binary

# Gameplan (in order of priority)

## Utilities

- Intake of rules
- Transformation of non-binary rules into binary rules
- Generation of new rules using optionality
- Generation of new rules using \*

## Sentence parsing

- Earley parser
- Transformation of binarized tree into non-binarized tree

## Tree generation

- Take in a tree object and generate svg using `Luxor.jl`
- Create a tree creation website using svg generation
  - This will provide practice building MVC in Genie.jl and generation of trees

## Sentence generation

- Generate random sentences using phrase structure rules

- Implement length cutoff in case of recursive rules

# Handling Optionality

The issue is that the optional pieces are occurring in rules of any length. E.g. we could have NP -> (D)(Adj)(Mod)N

The possible options for this are:

- NP -> D Adj Mod N
- NP -> D Adj N
- NP -> D Mod N
- NP -> D N
- NP -> Adj Mod N
- NP -> Adj N
- NP -> Mod N
- NP -> N

This can be seen as using a negated mask over the optional elements with the non-optional elements interpolated

- NP -> D Adj Mod N (000)
- NP -> D Adj N (001)
- NP -> D Mod N (010)
- NP -> D N (011)
- NP -> Adj Mod N (100)
- NP -> Adj N (101)
- NP -> Mod N (110)
- NP -> N (111)

if the bit at position n in the list of optionals is 1, then that position is removed.

# Hai's notes

1. upper limit for num of generations:

S -> NP VP NP:{John, Mary} VP:{eat, sleep}

when I ask it to generate 20 sentences, it generates many duplicates. Of course this simple grammar can only have 4 distinctive sentences. Do we want to allow duplicates? Or maybe set a parameter whether duplicates are allowed.

2. it doesn't seem to handle Chinese:

same small grammar but in Chinese:

S -> NP VP NP:{[], []} VP:{[], []}

will results in:

image.png

3. empty lines are not allowed? Any grammar with empty lines will give an error. I think we may want to allow empty lines?

I have fixed 1 & 2 mostly. Some small changes need to be made to CFG.jl to use a font that allows for more non-ascii characters like chinese characters.

3 is not resolved at the moment.

# Genie tidbits

css and other frontend assets should be modified in the public folder not the assets/css folder.