CS403: Programming Languages

Fun with Grammars

Fun with grammars

Here are a set of abstract languages and grammars which can generate them.

Set of all strings over A,B ending in A

```
s : zA
z : Az | Bz | empty
```

Note that the non-terminal *z* generates all strings composed of `A`'s and `B`'s. The non-terminal *s* simply appends all strings generated by *z* with an `A` to give the desired result.

Set of all strings over A,B beginning and ending in A

```
s : AzA | A
z : Az | Bz | *empty*
```

Here, *z* plays the same role as before. In this case, *s* simply prepends an `A` as well to give the desired result.

Set of all strings over A,B with three consecutive A's

```
s : zAAAz
z : Az | Bz | *empty*
```

Now, *s* places a *z* at both ends to ensure that there are three consecutive A's somewhere in the sentence.

Set of all strings over A,B such that there is a pairof A's separated by 4i, i >= 0, characters

```
s : zAlAz
z : Az | Bz | *empty*
l : fl | *empty*
f : Ag | Bg
g : Ah | Bh
h : Ai | Bi
i : A | B
```

Similar to before, but note that *l* generates zero or more *f* strings. Note that an *i* string is composed of a single character. Therefore, an *h* string is composed of two characters, a *g* string is composed of three characters and and *f* string is composed of four characters. Since there are zero or more *f* strings between the two A's in an *s* string, we get the desired language.

Set of all strings over A,B such that at no two A's andno two B's are adjacent

```
s : a | b
b : Ba | *empty*
a : Ab | *empty*
```

The non-terminals *a* and *b* flip flop back and forth to ensure no two like characters are adjacent.

Palindromes over A,B

```
s : A | B | AsA | BsB | *empty*
```

If we place an `A` in the front, we must place one in the back. Likewise for `B`'s.

Balanced parentheses

```
s : (s) | ss | *empty*
```

or

```
s : (s)s | *empty*
```

It is a temptation to define *s* as `(s)`. This disallows such balanced strings as `(()())`.

Set of a strings over A,B so that the number of As equalsthe number of B's

```
s : Ab | Ba | empty
a : As | Baa
b : Bs | Abb
```

Note that b stands for strings with 1 more B than A and a stands for strings with 1 more A than B. These non-terminals arise naturally out of the observation that the strings must start with an A or B and the consequences of that starting character on the remainder of the string.