CSE 531 - A CFG to generate \( u \# v \) such that 
\( u \) does not yield \( v \)

October 19, 2000
Kaustubh Deshmukh

In these notes we define a Context Free Grammar (CFG) for generating strings of the form \( u \# v \), such that \( u \) does not yield \( v \) because something goes wrong\(^1\). This CFG was used in the reduction of \( A_{TM} \) to the Everything Problem for CFGs.

Given a Turing Machine \( M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r) \), we want to generate all strings of the form \( u \# w \) such that \( u \) does not yield \( v \). For simplicity we make the assumption that \( M \) never tries to move its head of the left end of the tape. Let \( \Delta = \Gamma \cup Q \). All valid configurations are strings in \( \Delta \). Firstly we define a partial function \( F_M : \Delta^4 \rightarrow \Delta \) as defined in class, as follows:

\[
F_M(a, b, c, d) = e
\]

1. if \( a, b, c \in \Gamma \), then \( e = b \)
2. if \( a \in Q \), then
   (a) \( \delta(a, b) = (p, f, R) \Rightarrow e = p \)
   (b) \( \delta(a, b) = (p, f, L) \Rightarrow e = f \)
3. if \( b \in Q \), then
   (a) \( \delta(b, c) = (p, f, R) \Rightarrow e = f \)
   (b) \( \delta(b, c) = (p, f, L) \Rightarrow e = a \)
4. if \( c \in Q \), then
   (a) \( \delta(c, d) = (p, f, R) \Rightarrow e = b \)
   (b) \( \delta(c, d) = (p, f, L) \Rightarrow e = p \)

\(^1\)The “something goes wrong” stands for the fact that a string of the form \( u \# wx \) will not be generated, where \(|u| = |w|\) and \( u \) yields \( w \). Such strings are generated by “lengths wrong” CFG.
This function determines which character will occur in the place of $b$ in the yielded configuration, by looking at a window of four characters. We say that $a, b, c, d$ yields $e$.

We define the grammar $G_M = (V, \Delta', R, S)$ as follows:
$V$ consists of the non-terminals $S, C, F$ and $B^{(a,b,c,d)} \forall a, b, c, d \in \Delta$.
$\Delta' = \Delta \cup \{\#\}$
$S$ is the start symbol.

The rules are defined as follows:

1. $S \rightarrow B^{(a,b,c,d)} e C \quad \forall a, b, c, d, e \in \Delta$ such that $F(a, b, c, d) \neq e$
2. $S \rightarrow F$
3. $B^{(a,b,c,d)} \rightarrow x B^{(a,b,c,d)} y \quad \forall x, y \in \Delta$
4. $B^{(a,b,c,d)} \rightarrow a b c d C \neq x \quad \forall x \in \Delta$
5. $F \rightarrow b c d C \neq e C \quad \forall b, c, d, e \in \Delta$ such that $F(\sqcup, b, c, d) \neq e$
6. $C \rightarrow e \mid x C \quad \forall x \in \Delta$

The grammar works as follows:
The first rule first introduces an anomaly that cannot occur for a yield to work correctly. That is, if $e$ is the $(n + 1)^{\text{th}}$ symbol in $v$ then rule 1 will ensure that $a, b, c, d$, which will start at position $n$ in $u$, do not yield $e$, as $F(a, b, c, d) \neq e$. Rule 3 inserts $(n - 1)$ characters at the beginning of both $u$ and $v$. Rule 4 puts the characters $a, b, c, d$ in the $n^{\text{th}}$ position in $u$. The extra $x$ is to ensure that $e$ will occur in the $(n + 1)^{\text{th}}$ position in $v$, which is the same position in which $b$ occurs in $u$.

In the above description $e$ could never occur in the first position of $v$. That is, we could not generate strings where $u$ does not yield $v$ only because the first character of $v$ is wrong. To accommodate this, we have added rules 2 and 5. Rule 5 ensures that $b, c, d$ and $e$ are the initial characters of $u$ and $v$ respectively, and that $e$ is the wrong character as $F(\sqcup, b, c, d)^2 \neq e$.

Hence the grammar $G_M$ generates all strings of the form $u \# v$ where $u$ does not yield $w$ because something goes wrong.

---

2Any character in $\Gamma$ could be used as $a$ to capture the effect of $F$ in the case $bcd$ is the initial part of $u$. This can easily be seen from the definition of $F$. The blank is used just for convenience.