Refresher: Context-Free Grammars

- Context-free grammar $G = (N, T, P, S)$, where
  - N is a finite set of non-terminals, *with no useless non-terminals*;
  - T is a finite set of terminals (aka tokens);
  - P is a finite set of productions (rewrite rules) of the form $A \to \alpha$, where $A \in N$, and $\alpha \in (N \cup T)^*$ is a sentential form;
  - $S \in N$ is the start symbol, *which does not appear of the RHS of any production*.

Refresher: Context-Free Grammars

- Context-free grammar $G = (N, T, P, S)$, where
  - N is a finite set of non-terminals, *with no useless non-terminals*;
  - T is a finite set of terminals (aka tokens);
  - P is a finite set of productions (rewrite rules) of the form $A \to \alpha$, where $A \in N$, and $\alpha \in (N \cup T)^*$ is a sentential form;
  - $S \in N$ is the start symbol, *which does not appear of the RHS of any production*.
- For convenience, we will augment an input string s with a distinct end-of-string symbol \$, and change all productions of the form $S \to \alpha$ to $S \to \alpha\$$ (call this change \$-augmentation).

Refresher: Context-Free Grammars

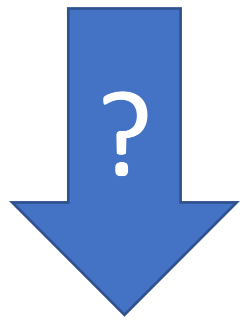
- Context-free grammar $G = (N, T, P, S)$, where
  - N is a finite set of non-terminals, *with no useless non-terminals*;
  - T is a finite set of terminals (aka tokens);
  - P is a finite set of productions (rewrite rules) of the form $A \rightarrow \alpha$, where $A \in N$, and $\alpha \in (N \cup T)^*$ is a sentential form;
  - $S \in N$ is the start symbol, *which does not appear of the RHS of any production*.
- For convenience, we will augment an input string s with a distinct end-of-string symbol \$, and change all productions of the form $S \rightarrow \alpha$ to $S \rightarrow \alpha\$$ (call this change \$-augmentation).
- Notational shortcuts: $\Sigma = N \cup T$, $\mathrm{F} = T \cup \{\varepsilon\}$, $T' = T \cup \{\$\}$.

Refresher: Context-Free Grammars

- Context-free grammar $G = (N, T, P, S)$, where
  - N is a finite set of non-terminals, *with no useless non-terminals*;
  - T is a finite set of terminals (aka tokens);
  - P is a finite set of productions (rewrite rules) of the form $A \rightarrow \alpha$, where $A \in N$, and $\alpha \in (N \cup T)^*$ is a sentential form;
  - $S \in N$ is the start symbol, *which does not appear of the RHS of any production*.
- For convenience, we will augment an input string s with a distinct end-of-string symbol $, and change all productions of the form $S \rightarrow \alpha$ to $S \rightarrow \alpha\$$ (call this change $-augmentation).
- Notational shortcuts: $\Sigma = N \cup T$, $\mathrm{F} = T \cup \{\varepsilon\}$, $T' = T \cup \{\$\}$.
- A derivation $\alpha \stackrel{*}{\Rightarrow} \beta$, with $\alpha, \beta \in \Sigma^*$, is a finite sequence of (valid) applications of productions that rewrites α to β.

TODO: Constructing Parsing Tables

S → ES'
S' → ε
S' → +S
E → num
E → (S)



	num	+	(	)	$
S	→ ES'		→ ES'		
S'		→ +S		→ ε	→ ε
E	→ **num**		→ (S)		

From Grammar To Parsing Table

- Problem
  - Decide whether the production $A \rightarrow \alpha$ of the context-free grammar $G = (N, T, P, S)$ is a candidate for rewriting A when the input token is t, i.e., whether $A \rightarrow \alpha \in ParsingTable[A, t]$.

From Grammar To Parsing Table

- Problem
  - Decide whether the production $A \rightarrow \alpha$ of the context-free grammar $G = (N, T, P, S)$ is a candidate for rewriting A when the input token is t, i.e., whether $A \rightarrow \alpha \in ParsingTable[A, t]$.
- Key insight
  - Suppose we know the first symbols of all sentences derivable from the sentential form α. Call this set $FIRST(\alpha) \subseteq F$.
  - Then $A \rightarrow \alpha$ is a candidate if $t \in FIRST(\alpha)$.

From Grammar To Parsing Table

- Problem
  - Decide whether the production $A \to \alpha$ of the context-free grammar $G = (N, T, P, S)$ is a candidate for rewriting A when the input token is t, i.e., whether $A \to \alpha \in ParsingTable[A, t]$.
- Key insight
  - Suppose we know the first symbols of all sentences derivable from the sentential form α. Call this set $FIRST(\alpha) \subseteq F$.
  - Then $A \to \alpha$ is a candidate if $t \in FIRST(\alpha)$.
  - Almost!
  - The only complication is if $\alpha \overset{*}{\Rightarrow} \varepsilon$. Call this predicate $NULLABLE(\alpha)$.

From Grammar To Parsing Table

- Problem
  - Decide whether the production $A \rightarrow \alpha$ of the context-free grammar $G = (N, T, P, S)$ is a candidate for rewriting A when the input token is t, i.e., whether $A \rightarrow \alpha \in ParsingTable[A, t]$.
- Key insight
  - Suppose we know the first symbols of all sentences derivable from the sentential form α. Call this set $FIRST(\alpha) \subseteq F$.
  - Then $A \rightarrow \alpha$ is a candidate if $t \in FIRST(\alpha)$.
  - Almost!
  - The only complication is if $\alpha \overset{*}{\Rightarrow} \varepsilon$. Call this predicate $NULLABLE(\alpha)$.
  - In this case, the production is still a candidate if $S \overset{*}{\Rightarrow} \eta A t \xi$.

From Grammar To Parsing Table

- Problem
  - Decide whether the production $A \rightarrow \alpha$ of the context-free grammar $G = (N, T, P, S)$ is a candidate for rewriting A when the input token is t, i.e., whether $A \rightarrow \alpha \in ParsingTable[A, t]$.
- Key insight
  - Suppose we know the first symbols of all sentences derivable from the sentential form α. Call this set $FIRST(\alpha) \subseteq F$.
  - Then $A \rightarrow \alpha$ is a candidate if $t \in FIRST(\alpha)$.
  - Almost!
  - The only complication is if $\alpha \overset{*}{\Rightarrow} \varepsilon$. Call this predicate $NULLABLE(\alpha)$.
  - In this case, the production is still a candidate if $S \overset{*}{\Rightarrow} \eta A t \xi$.
  - So we also need to know the set of terminals that can immediately follow A in some sentential form derivable from S. Call this set $FOLLOW(A) \subseteq T'$.
  - That is, if $NULLABLE(\alpha)$, then the production $A \rightarrow \alpha$ is also a candidate if $t \in FOLLOW(A)$.

Definition: $NULLABLE$

- $NULLABLE(\alpha) = true$ iff $\alpha \overset{*}{\Rightarrow} \varepsilon$.

- $NULLABLE: \Sigma^* \to \{\text{true}, \text{false}\}$ is defined inductively as follows:
  - $NULLABLE(\varepsilon) = \text{true}$.
  - $\forall t \in T, NULLABLE(t) = \text{false}$.
  - $NULLABLE(Y_1 \cdots Y_k) = NULLABLE(Y_1) \wedge \cdots \wedge NULLABLE(Y_k)$.
  - $(A \to \alpha \in P) \wedge NULLABLE(\alpha) \Longrightarrow NULLABLE(A)$.

Definition: $FIRST$

- $FIRST(\alpha)$ is the set of the first symbols of all sentences derivable from α.

- $FIRST: \Sigma^* \rightarrow F$ is defined inductively as follow:
  - $FIRST(\varepsilon) = \{\varepsilon\}.$
  - $\forall t \in T, FIRST(t) = \{t\}.$
  - $FIRST(Y_1 \cdots Y_k) = FIRST(Y_1) +_1 \cdots +_1 FIRST(Y_k).$
  - $(A \rightarrow Y_1 \cdots Y_k \in P) \wedge \big(f \in FIRST(Y_1 \cdots Y_k)\big) \Longrightarrow f \in FIRST(A).$

- $+_1: 2^F \times 2^F \rightarrow 2^F$ is a binary helper function on subsets of F defined thus:
  - Concatenate each element of the first set with each element of the second set, and truncate to the first symbol of the resulting string.
  - Thus, $\{\varepsilon, a, b\} +_1 \{\varepsilon, c\} = \{\varepsilon, a, b, c\}$, but $\{\varepsilon, a, b\} +_1 \{c\} = \{a, b, c\}$.

Definition: $FOLLOW$

- $FOLLOW(A)$ is the set of terminals that can appear immediately following A in some sentential form derivable from S.

- $FOLLOW: N \rightarrow T'$ is defined inductively as follows:
  - $\$ \in FOLLOW(S)$.
  - $(A \rightarrow X_1 \cdots X_k B Y_1 \cdots Y_m \in P) \wedge$
    $\big(t \in FIRST(Y_1 \cdots Y_m) +_1 FOLLOW(A)\big) \Longrightarrow t \in FOLLOW(B).$

Algorithm for Computing Sets

```
for each A do {
    NULLABLE[A] = false;
    FIRST[A] = FOLLOW[A] = { };
}
FOLLOW[S] = {$};
repeat {
  for each production A → Y_1 ⋯ Y_n do {
      if NULLABLE(Y_1 ⋯ Y_n) then NULLABLE[A] = true;

      FIRST[A] = FIRST[A]  U  FIRST(Y_1 ⋯ Y_n);

      for each Y_i do
         FOLLOW[Y_i] = FOLLOW[Y_i] U (FIRST(Y_{i+1} ⋯ Y_k) +_1 FOLLOW[A]);
  }
} until sets do not change
```

Example

S → A$

A → BC | x

B → t | ε

C → v | ε

NULLABLE = {A,B,C}
FIRST(A)={x,t,v,ε}
FIRST(B)={t,ε}
FIRST(C)={v,ε}
FIRST(S)={x,t,v,$}
FOLLOW(A)={$}
FOLLOW(B)={v,$}
FOLLOW(C)={$}